

Optimiser les Tokens

Philippe PRADOS

12.10.2026

Philippe PRADOS

carbon

- Contributeur Python
- Contributeur Langchain
- +100 articles dans GNU Linux Mag
- Des dizaines de conférences
- Plusieurs projets Open Sources

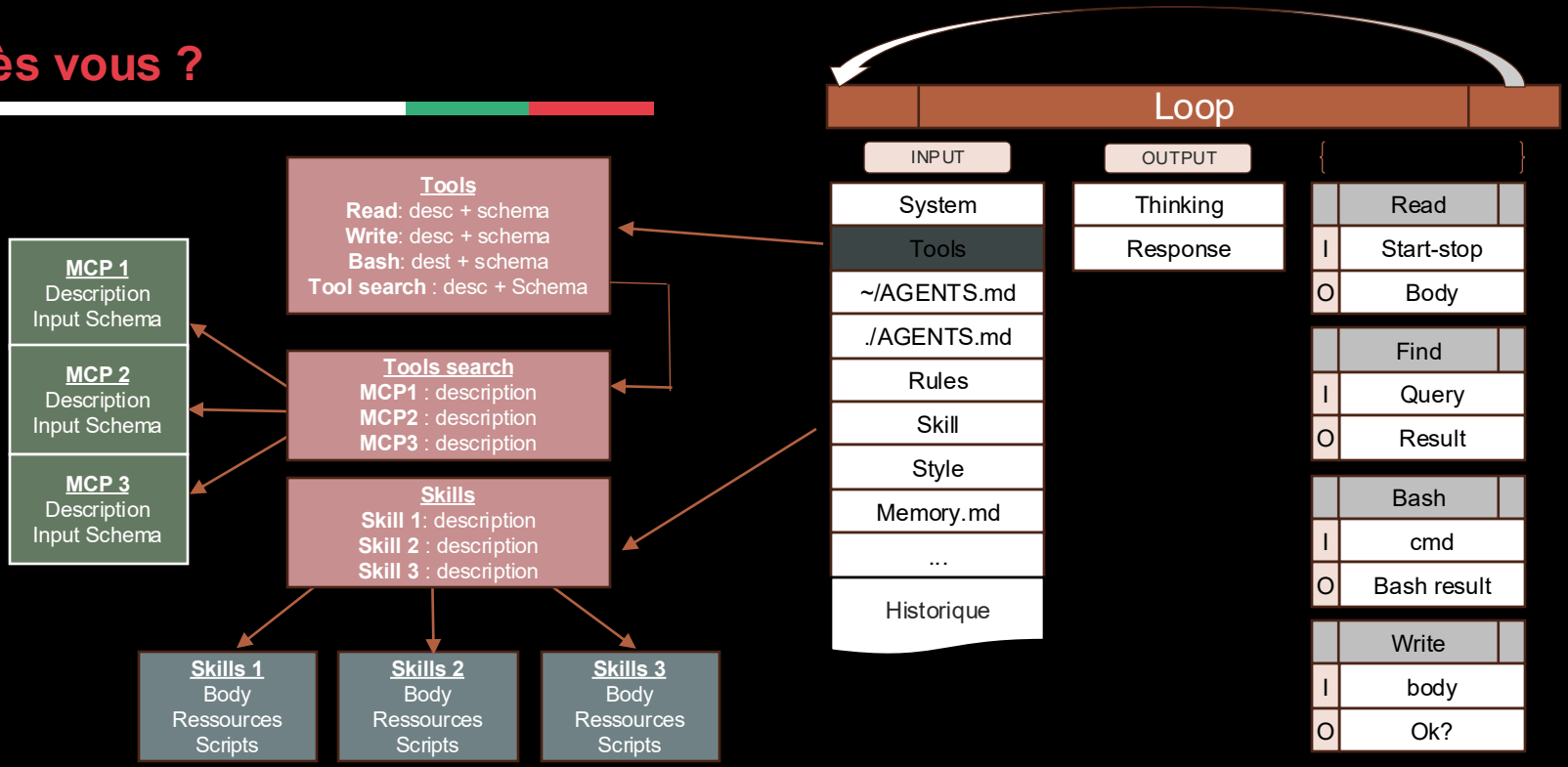




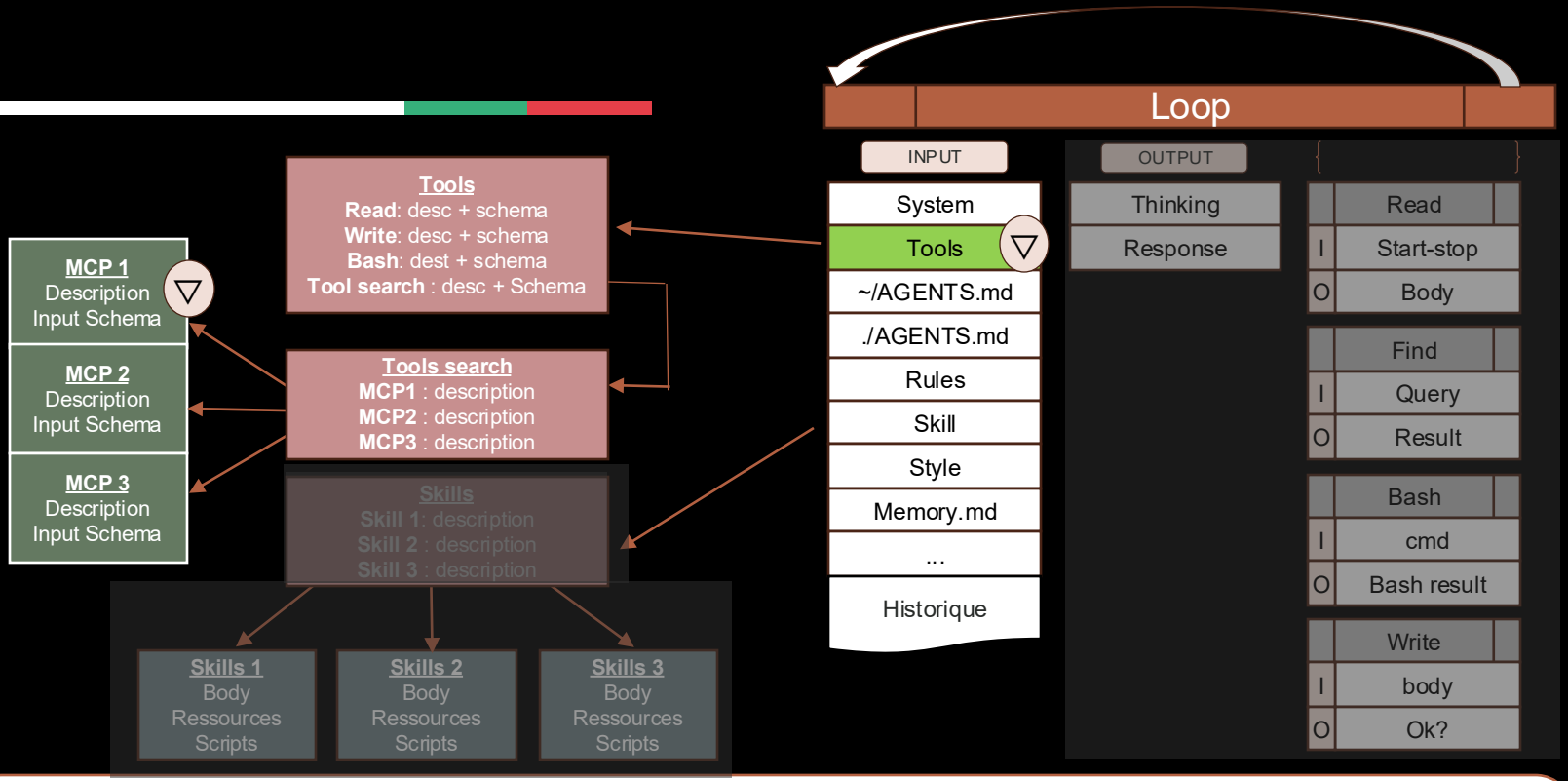
Où et comment optimiser ?



D'après vous ?

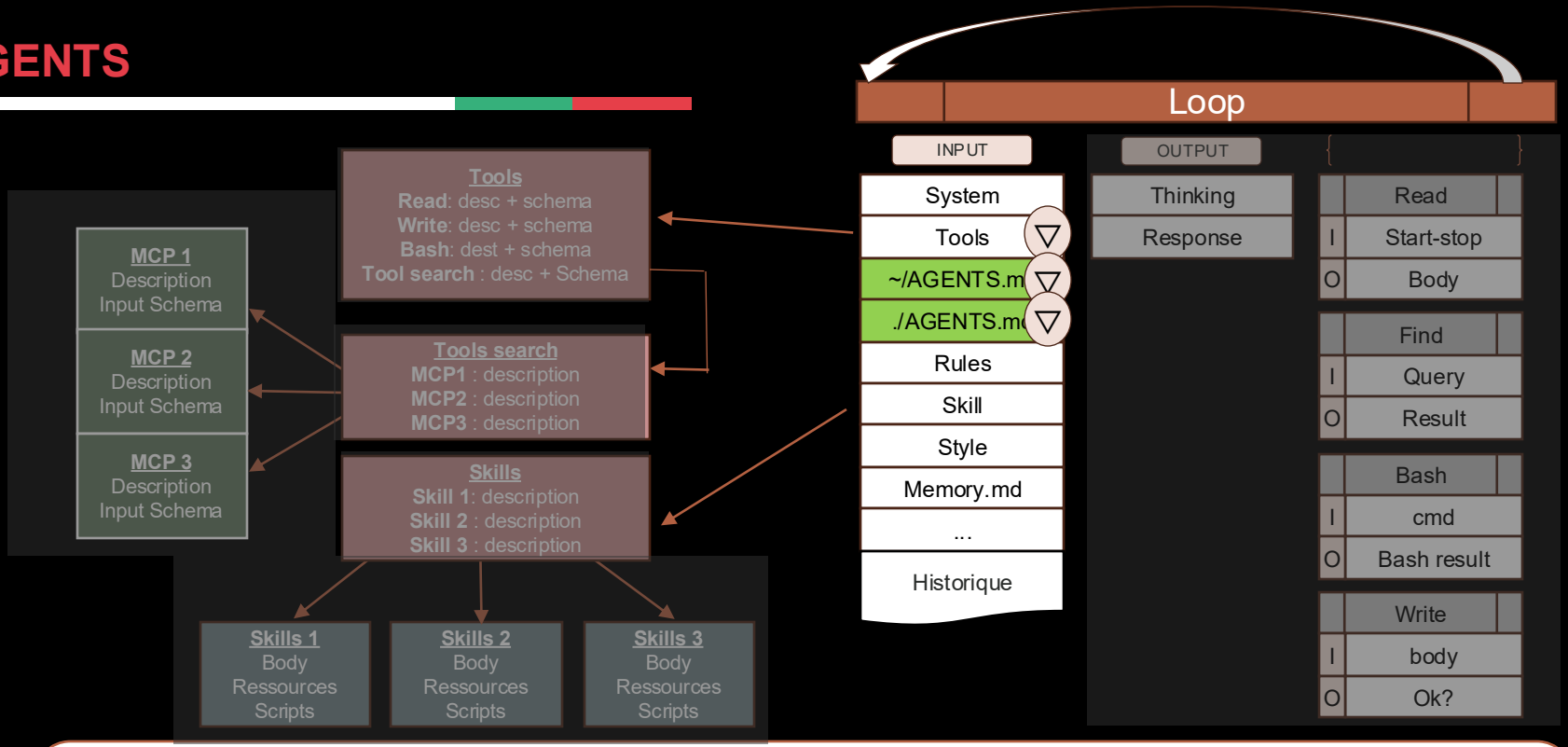

Loop
INPUT
OUTPUT
System
Tools
~/AGENTS.md
./AGENTS.md
Rules
Skill
Style
Memory.md
...
Historique
Thinking
Response
Read
I Start-stop
O Body
Find
I Query
O Result
Bash
I cmd
O Bash result
Write
I body
O Ok?
Tools
Read: desc + schema
Write: desc + schema
Bash: dest + schema
Tool search : desc + Schema
Tools search
MCP1 : description
MCP2 : description
MCP3 : description
Skills
Skill 1: description
Skill 2 : description
Skill 3 : description
MCP 1
Description
Input Schema
MCP 2
Description
Input Schema
MCP 3
Description
Input Schema
Skills 1
Body
Ressources
Scripts
Skills 2
Body
Ressources
Scripts
Skills 3
Body
Ressources
Scripts

MCP



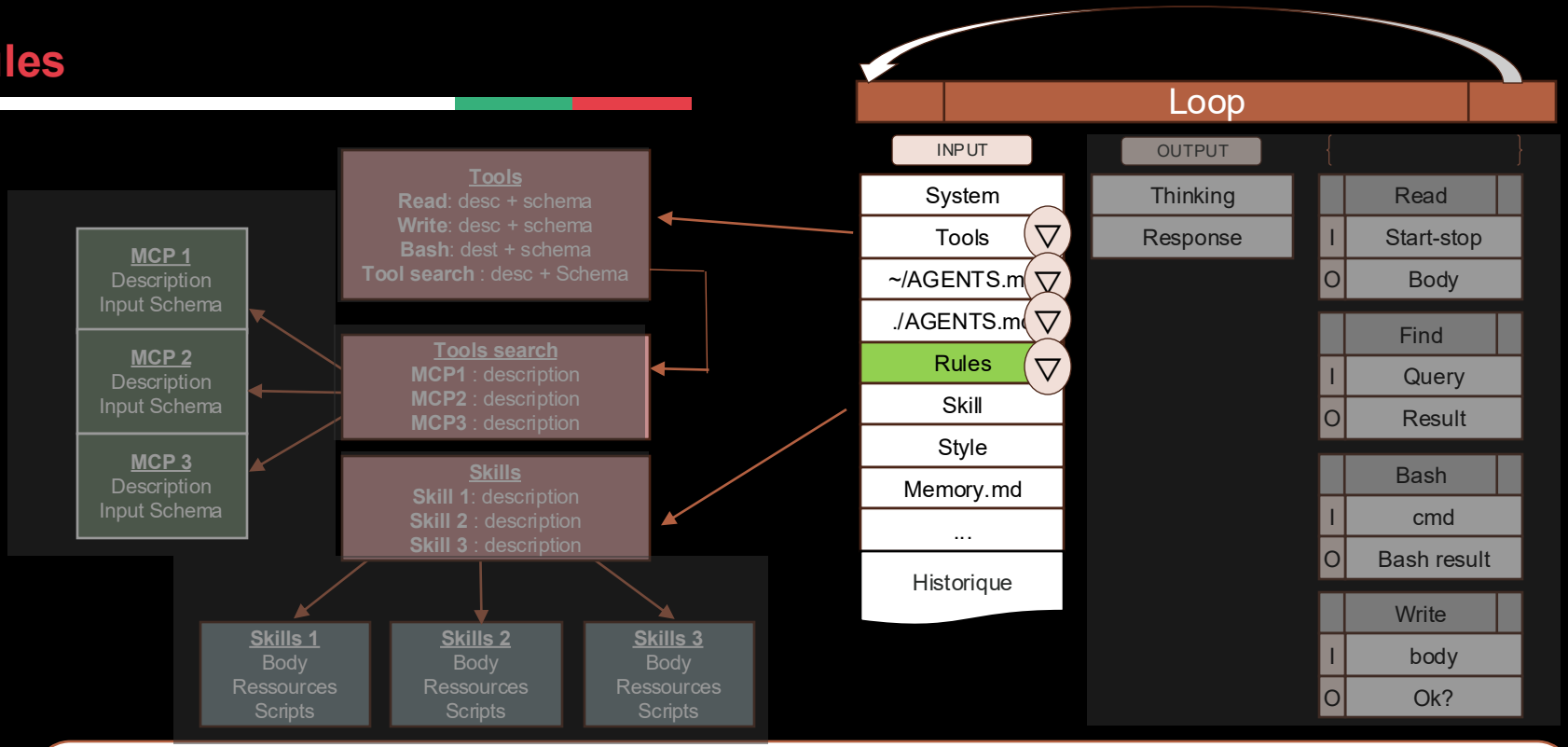
- Limiter la liste à ceux réellement utiles
- Choisir des MCP qui accomplissent la tâche en **une seule invocation**
- Privilégier un **résultat court**
- **Choisir des outils qui sont plus qu'un simple relais d'API (wrapper API) !**

AGENTS



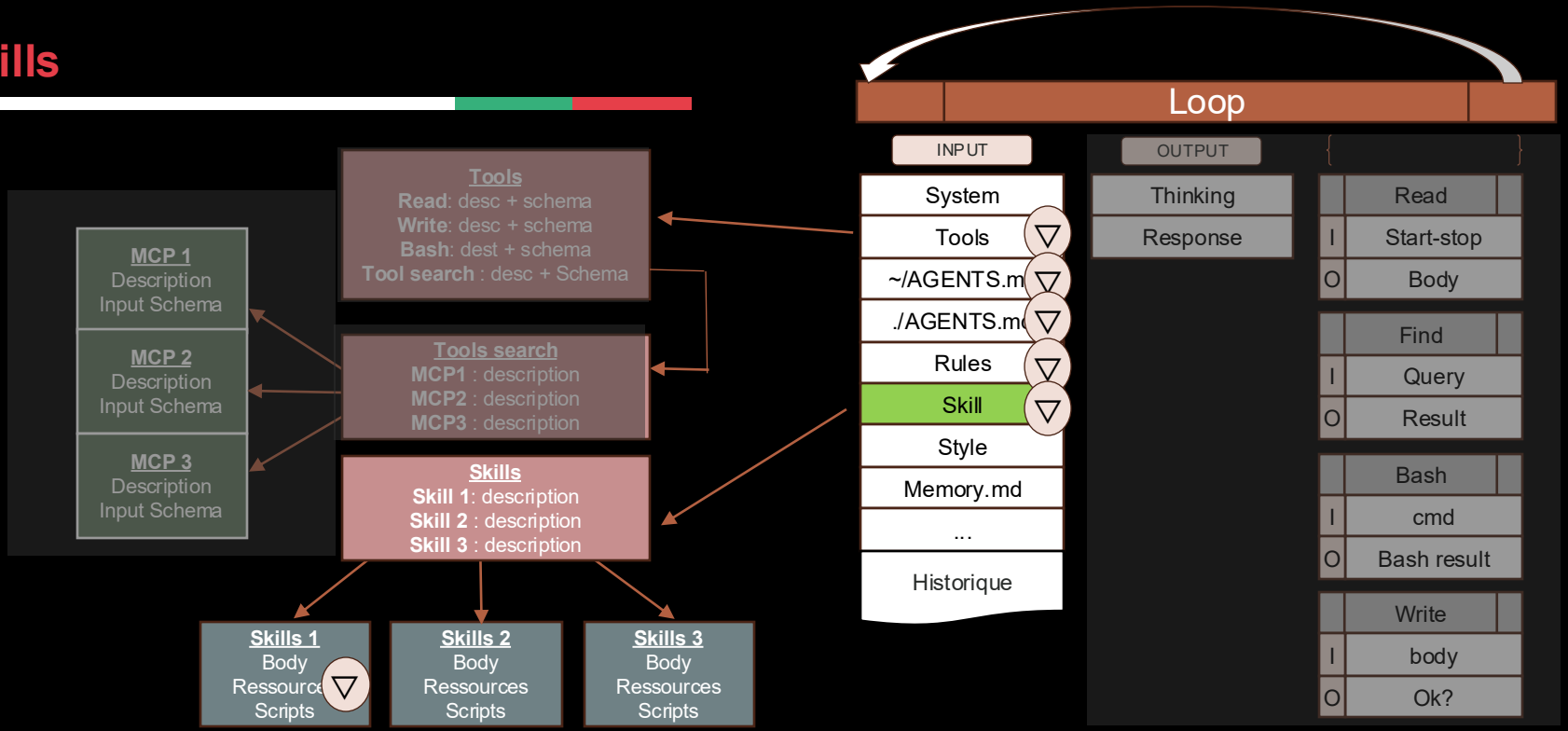
- `/doctor` analyse et propose des améliorations
- **Utilisateur :** les préférences personnelles
  Projet : uniquement le spécifique projet
- Pas de considérations générique (style pour les langages de dev, etc)
- Des prompts d'amélioration (Prompt Karpathy, ponytail, code-simplifier...)

Rules



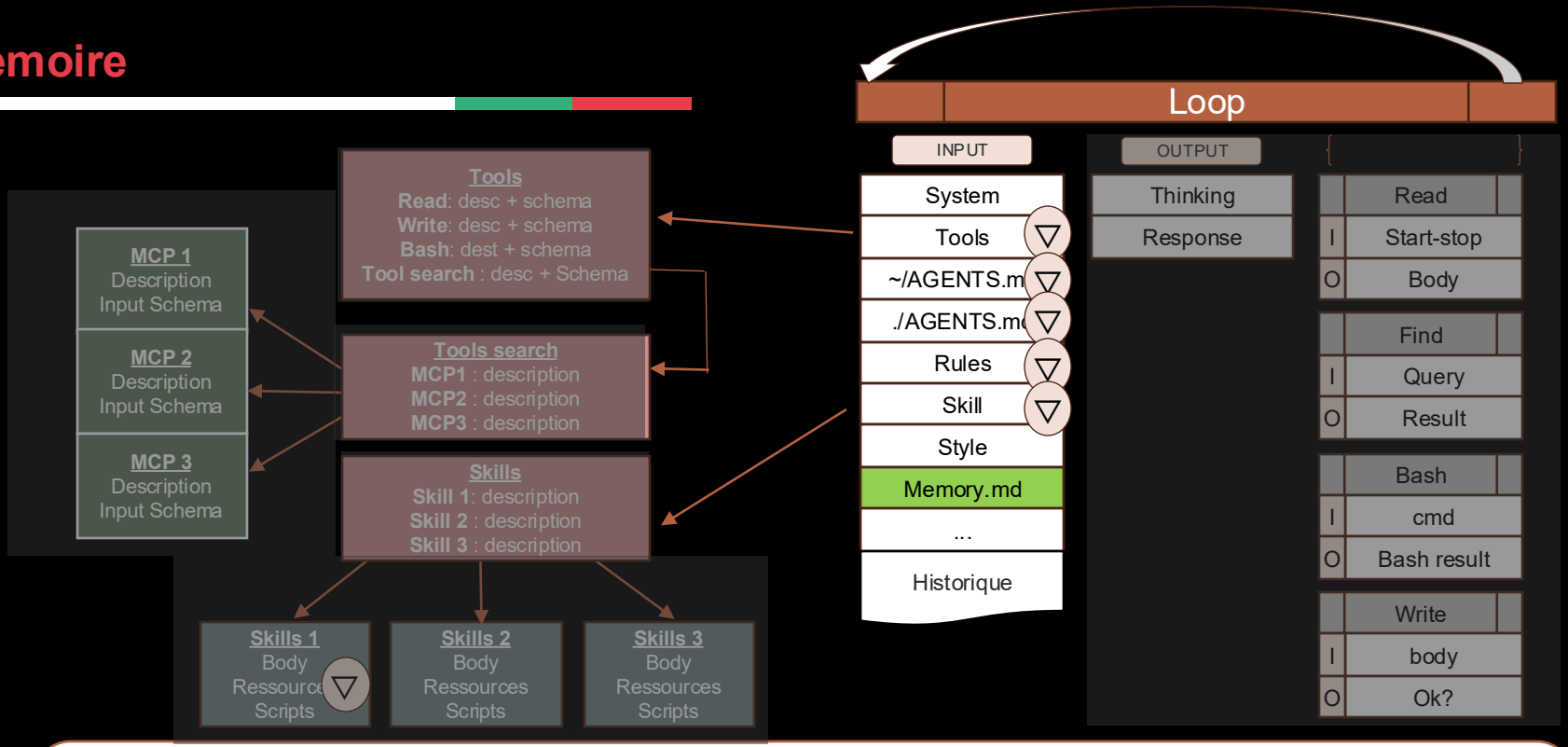
- Fragment de prompt injecté **seulement si une condition** est réunie
- Des règles filtrées par **types de fichiers**
- Les spécificités de votre utilisation du langage (Les LLM savent déjà coder correctement)

Skills



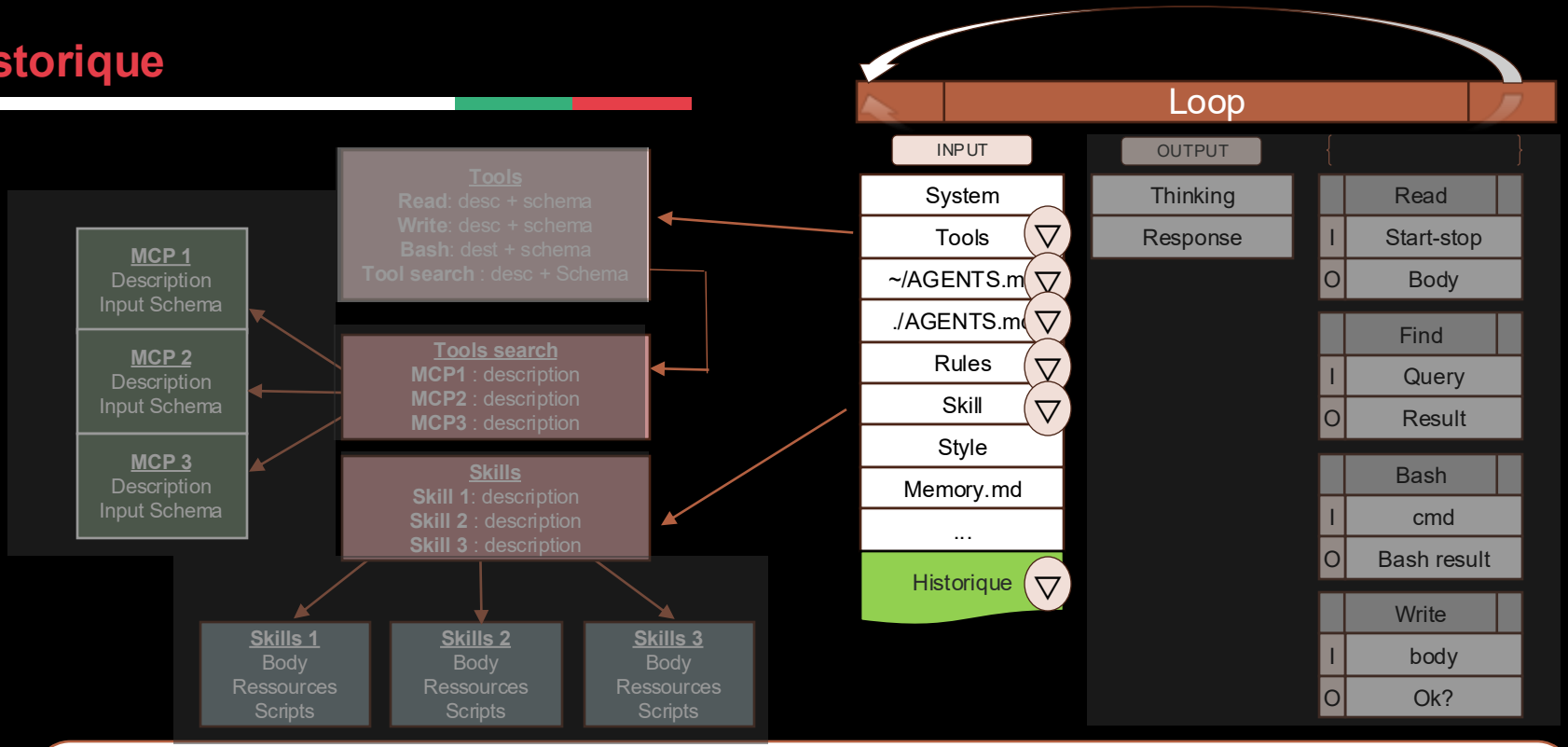
- Limiter la liste à ceux réellement utiles
- Réduire la description au minimum (**à quoi il sert et quand l'utiliser**)
- Body réduit à l'objectif, aux ressources, aux paramètres des scripts
- Supprimer les listes d'étapes précises

Mémoire



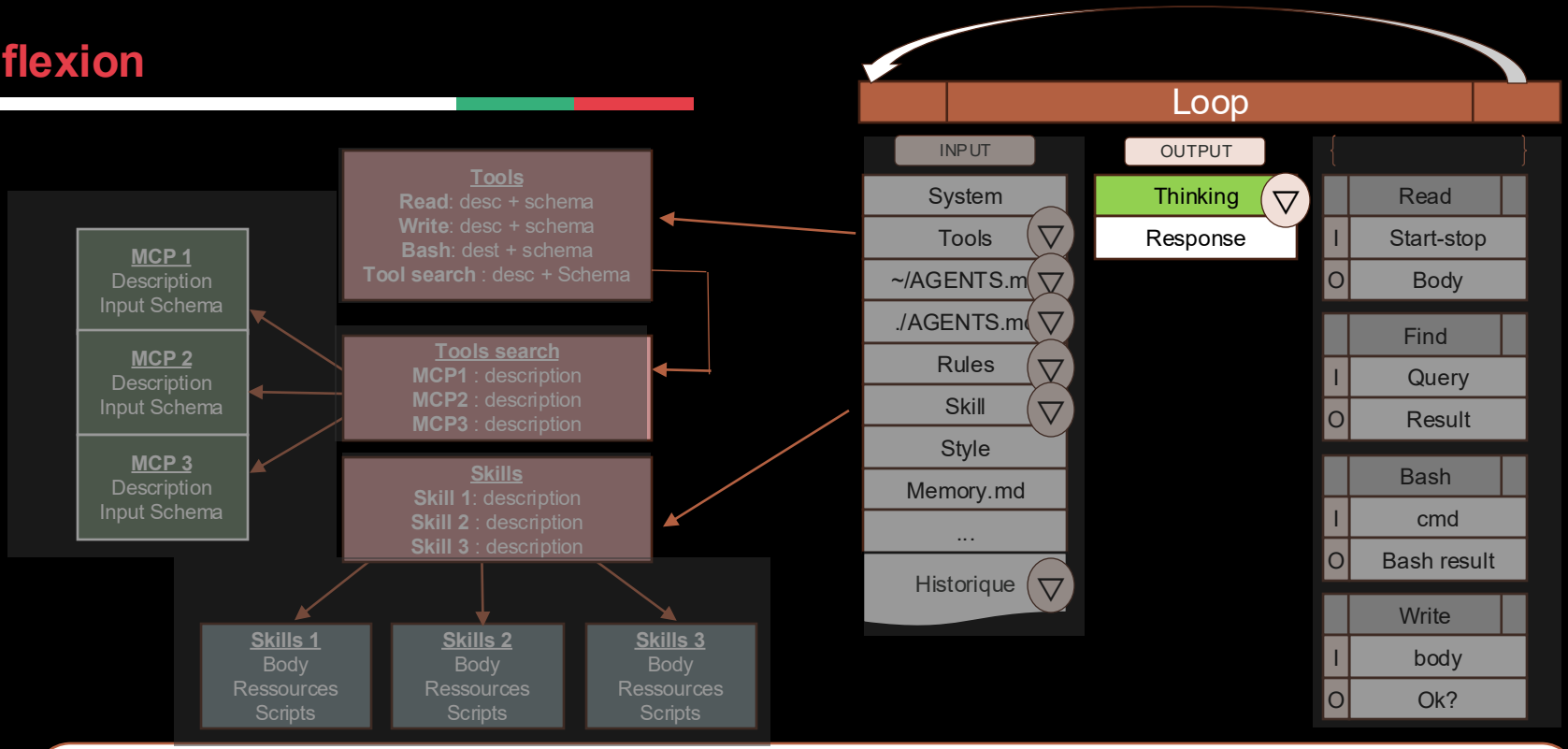
- La mémoire consomme des tokens…
- Améliore la recherche dans le projet
- Améliore la pertinence des réponses IA
- RAG, couches, « savoir oublier », extension Git
- Mutualiser sur le projet amortit le coût
- `/memory`, `/dream`, Memsearch, mempalace, Serena, Hindsight, Entire

Historique



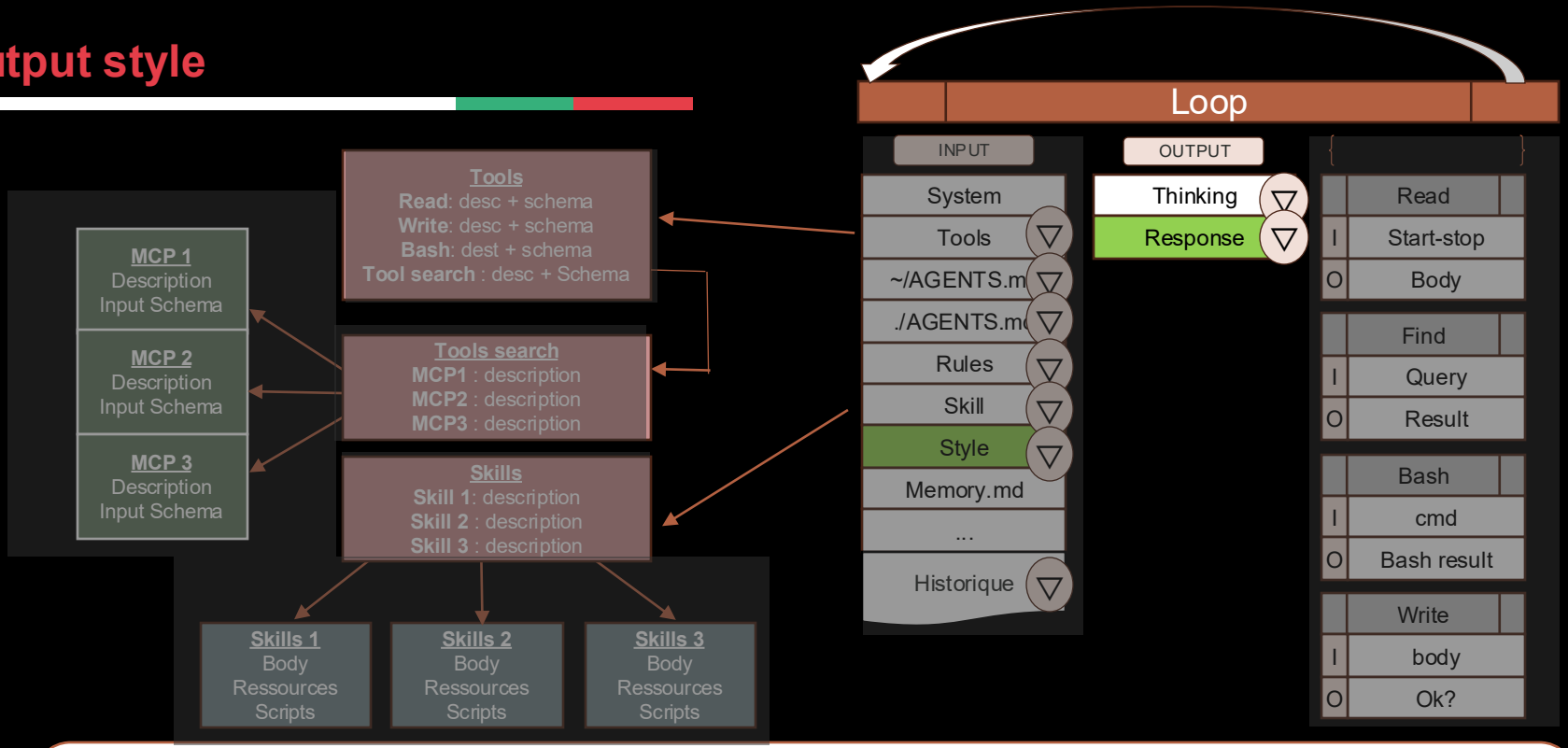
- L'historique s'accumule au fil de la conversation
- Cache activé = moins d'input
- Rester clair, garder le contexte en dessous de 50%
- `/clear`
- `/compact` (Attention à bien le rentabiliser)

Réflexion



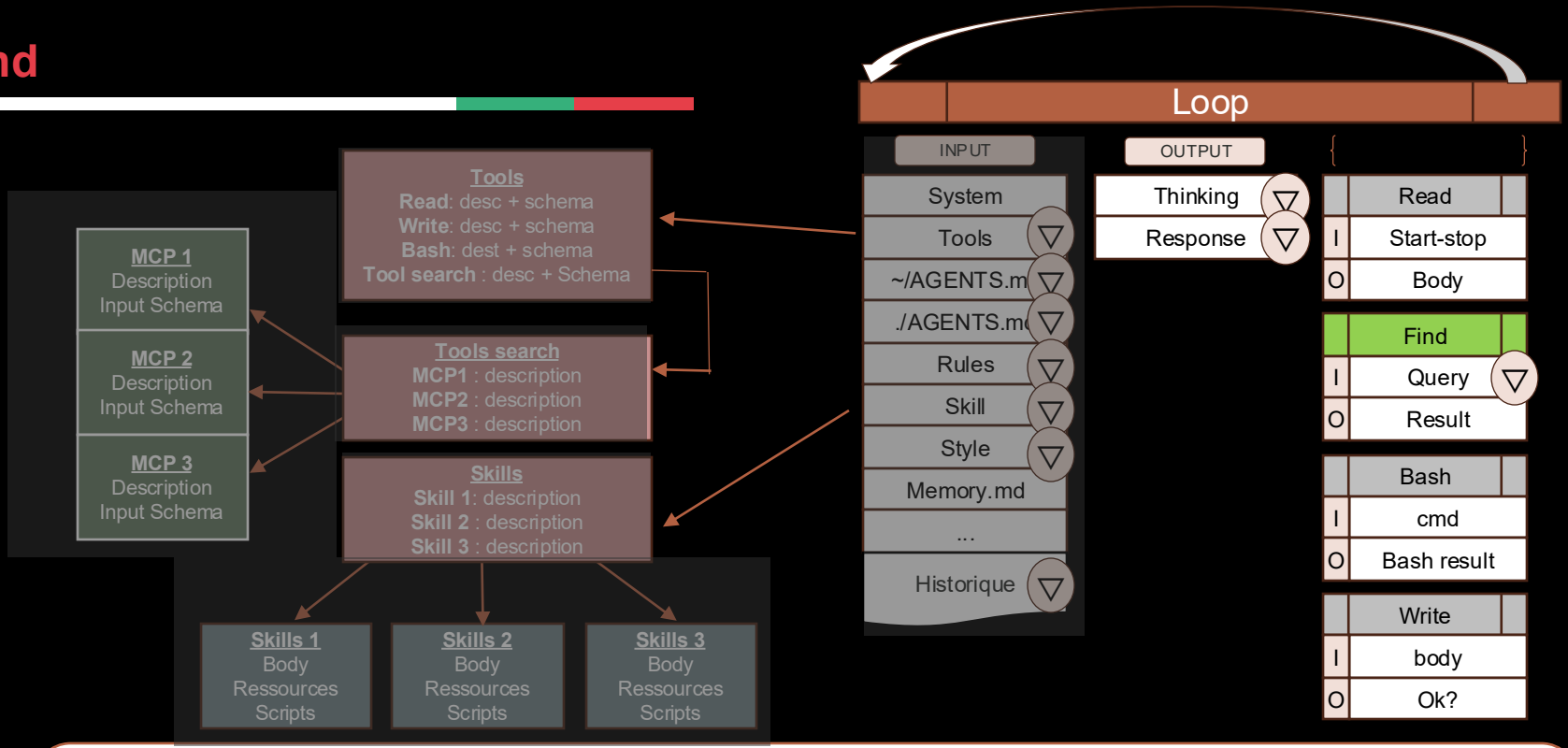
- Adapter le `/model` et `/effort` à la tâche
- **Gros** modèle : planification, architecture (Opus, Sol, Gemini Pro)
- Modèle **moyen** : implémentation (Sonnet, Luna, Gemini Flash)
- **Petit** modèle : tâches simples et répétitives (Haiku, Terra, Gemini Flash-Lite)
- Effort : faible pour les tâches simples, élevé pour les problèmes complexes.
- Dans le doute `/model opusplan` (Mode spécial qui utilise opus pendant le mode plan, puis bascule vers sonnet pour l'exécution)
- Les dernières générations cherchent à répondre aussi bien, en réfléchissant moins

Output style



- Choisir le style avec `/config output style`
- Caveman communication condensée « homme des cavernes »
  - Compresse AGENTS.md, Skills, descriptions (alt. kompress-tokens)
  - Permet également de "compresser" les `AGENTS.md`, `SKILLS`, descriptions, etc. (Alternative : Kompress-tokens)
- Pas d'impact sur le `<thinking>`
- Headroom, compression juste avant l'invocation des APIs
- Claw Compactor

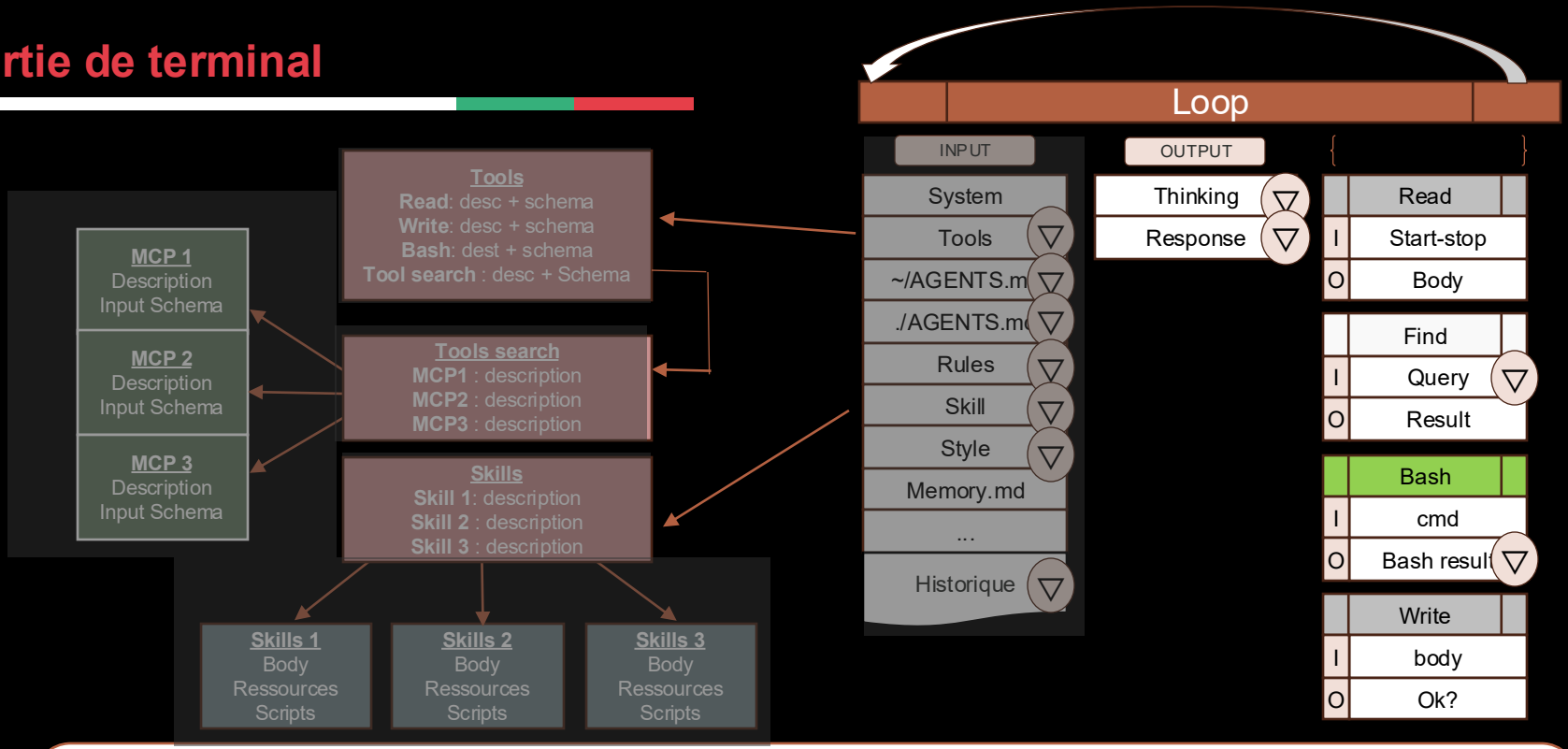
Find



- Outils par défaut `find` + `grep`
- Un MCP d'indexation améliore les résultats
  - `/ide`, ast-grep, `*-lsp`
  - Recherche floue (`ug --fuzzy=<n> <pattern>`)

- Renommer les symboles ambigües
- Injecter des commentaires
- Consomme des tokens à amortir en équipe
  - Création de base graphe
  - LLM Wiki
- Code Review Graph, Graphify, codebase-memory-mcp, Serena, Semble, Socraticode, ...

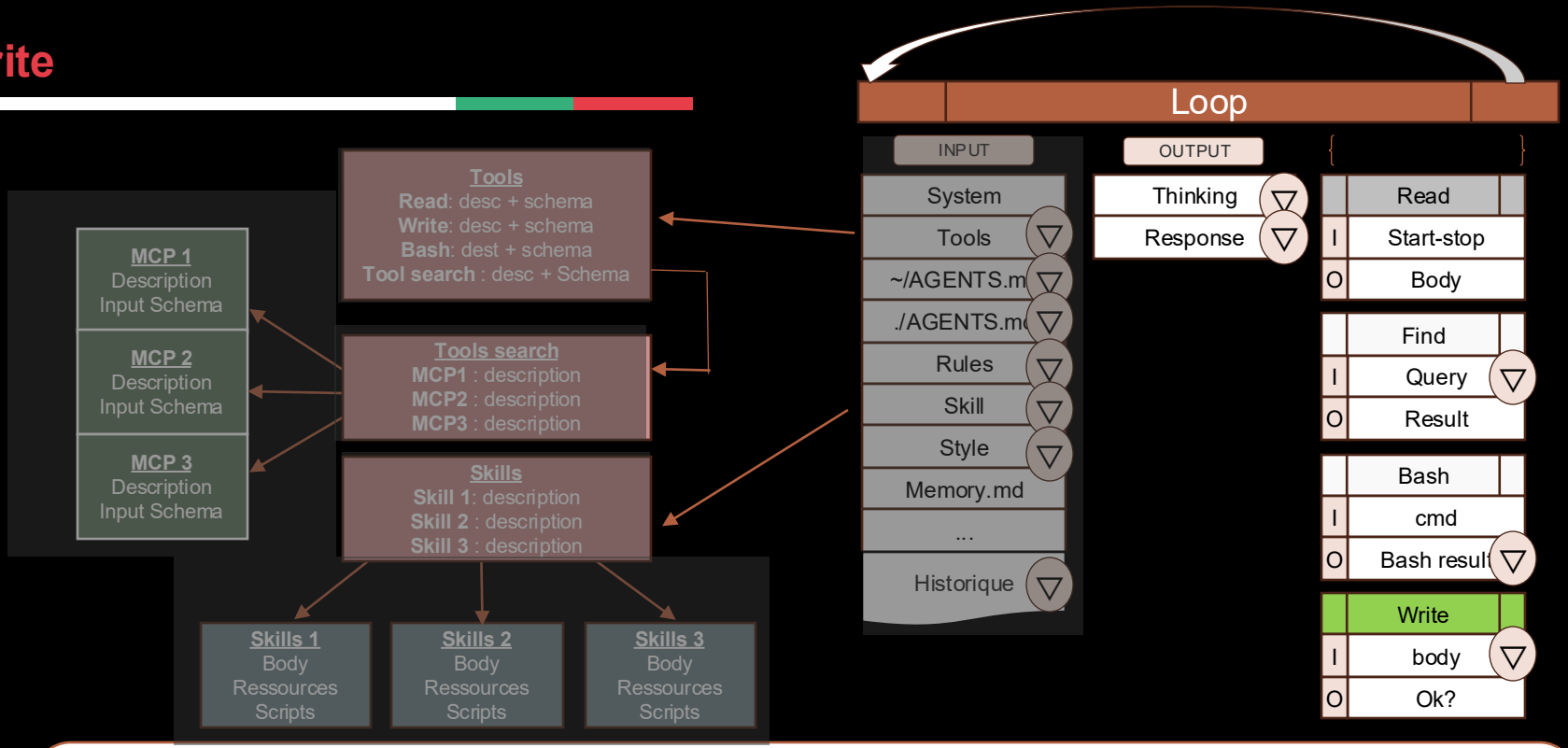
Sortie de terminal



- Compresser la sortie
  - Réduire la taille
  - Synthétiser la réponse en proposant de venir chercher ce qui manque
  - Ne pas laisser le LLM calculer
    (96 tokens pour 5 chiffre, 510 pour 10ch - croît comme n^2)

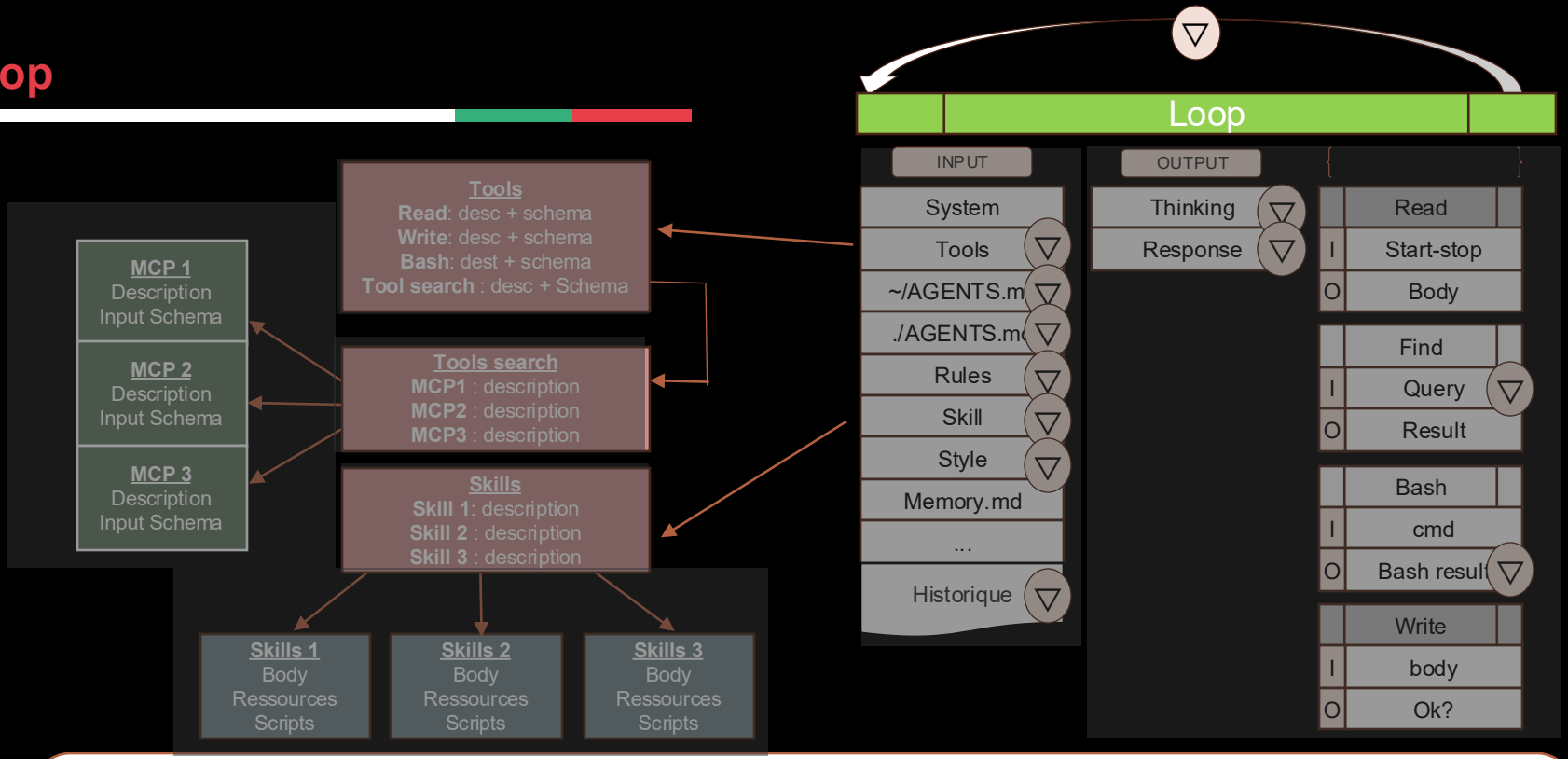
- RTK (Détail à 17h20) / Snip / Token-saver
- `curl.md/<url>`

Write



- Privilégier l'édition multi-fichiers.
- La modification par fichier n'est pas idéale (rename symbole par exemple)
- Connecter l'IDE pour accéder à ses fonctionnalités (/ide, MCP).
- En CI/CD, utiliser un CLI et/ou MCP (Serena)

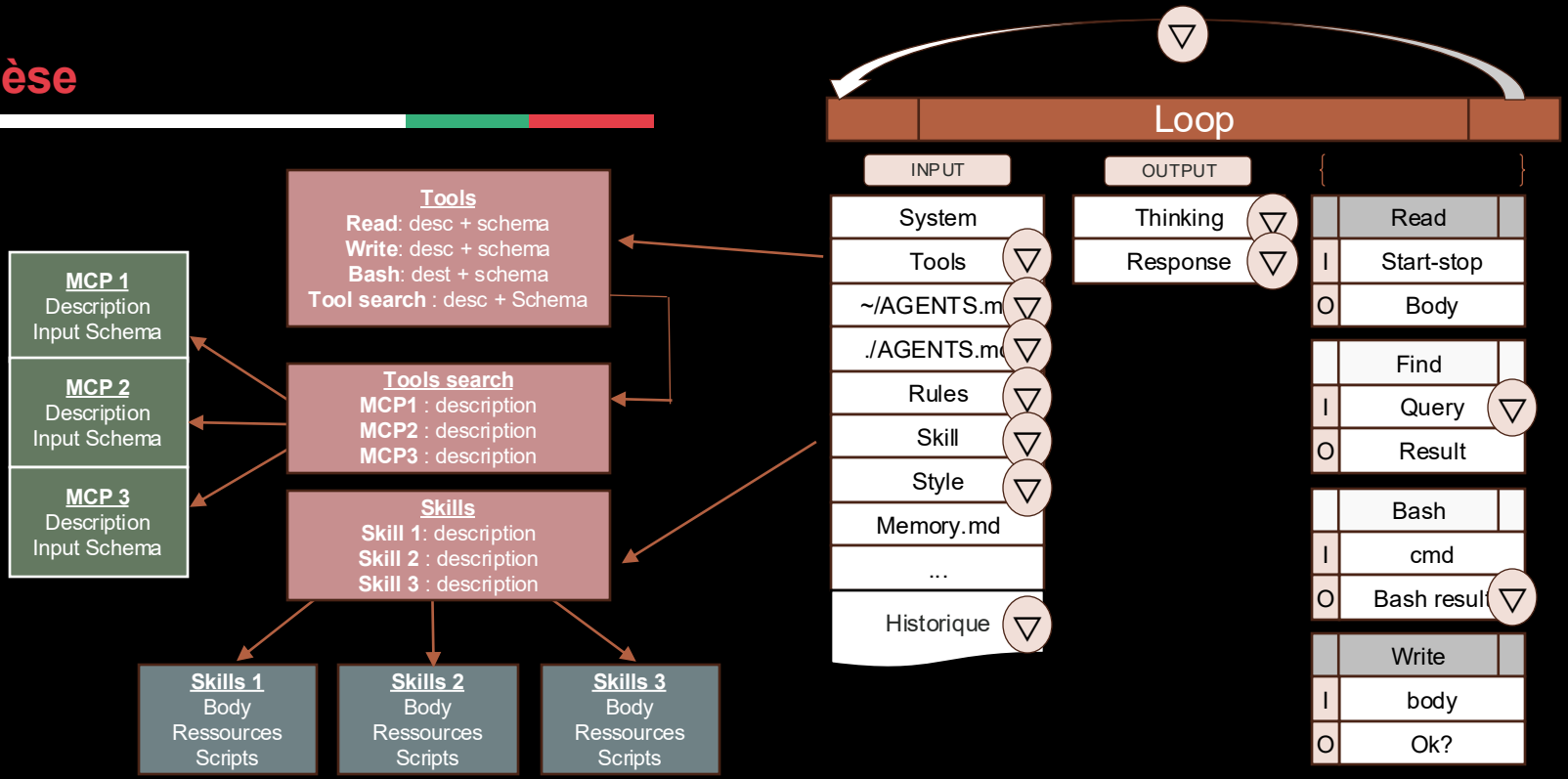
Loop



- Moins de cycles : combiner les commandes dans un script (context-mode)
- Suivre en direct et **couper dès que cela dérive**
- @file et non simplement le nom du fichier
- **No shell tool → ask `! <cmd>`. No workaround.**
- Installer tous les CLI que le LLM peut avoir besoin

- Hooks pour formater le code généré
- Au lieu de râler et demander des corrections, revenir en arrière et ajuster le prompt, pour repartir sur de bonnes bases.

Synthèse



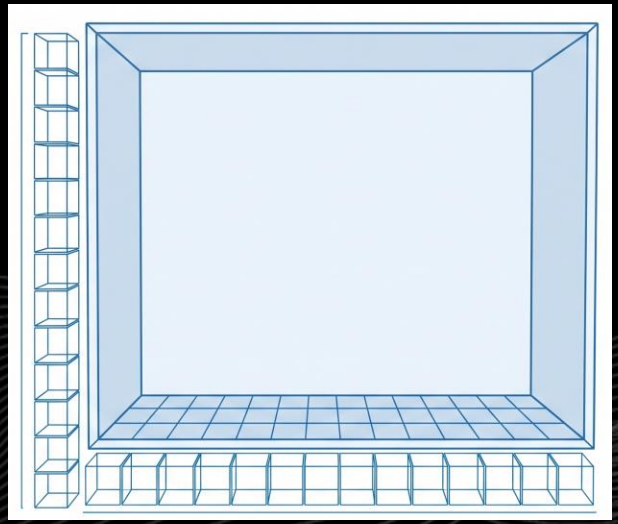
- De nouvelles idées chaque jours
- Très difficile de se tenir au courant
- Le glissement du *how to do* vers le *how to think*



/compress



K/V



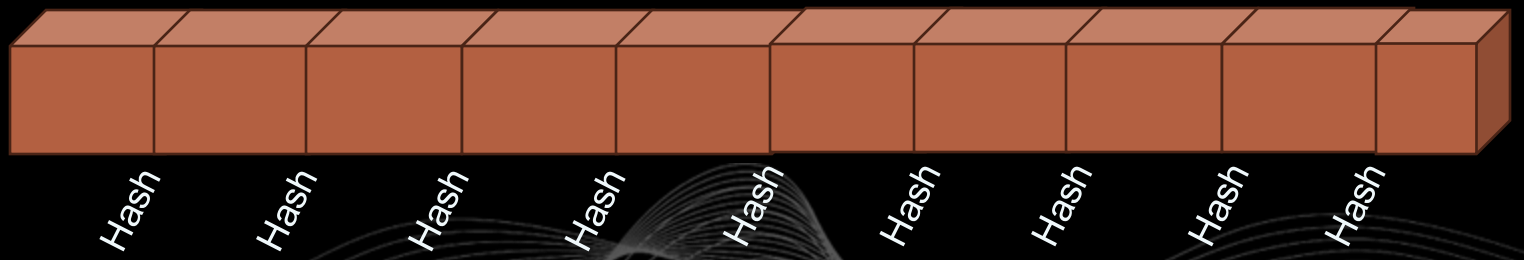
Un LLM maintient sont état dans un dictionnaire Clé/valeur

- Clé : un vecteur de "sujet" (de quoi cela parle)
- Valeur : Un vecteur de la connaissance actuelle sur le sujet

Calculé pour chaque token

Pour produire le projet, il faut tous reprendre pour retrouver l'état de réflexion juste avant le token à produire

L'idée d'un cache



Prendre un snapshot régulièrement, indexé par le hash du prompt

Lorsque le prompt arrive, chercher le point de départ le plus proche de la fin et partir de là

Revenir en arrière est bien plus efficace que se battre pour corriger une erreur.

De même que faire un faire une fois le contexte constitué

Simulateur

https://tinyurl.com/4fesmnnt





Organiser cela



Projet dédié aux paramètres

carbon

Plusieurs cibles: Claude, Opex, etc.

Décision : Utiliser vs Projet vs Local

Script d'installation des outils sélectionnées

Chaque outils à son installation, avec ajout de hook, de skills, de configuration, etc.

Template ajustable

```
# AGENT.template.md
{% include 'karpathy.md' %}
{% include 'compress.md' %}
{% include 'workflow.md' %}
{% include 'lesson.md' %}
{% include 'PPR.md' %}
{% include 'curl_md.md' %}
{% include '4lines.md' %}
{% include 'RTK.md' %}
```

Outil maison : https://github.com/pprados/kompress_tokens



Pour conclure...

Les grands principes

carbon

Moins de cycles

Des cycles plus efficaces

Avec des informations sélectionnées, les plus utiles à la tâche

Ne pas laisser le LLM chercher des contournements. Régler le problème une fois pour toute.



Merci

