# Introduction aux agents et à MCP

EMSI - Université Côte d'Azur
Richard Grin
Version 0.9 - 11/10/25

1

## Plan du support





2

## Introduction aux agents



3

## Présentation

- L'IA agentique est le sujet à la mode en IA
- Un agent IA est un système capable de
  - percevoir un environnement
  - prendre des décisions
  - agir de façon autonome pour atteindre un objectif



4

## Ce que peut faire un agent

- Il peut analyser un problème et concevoir un plan en plusieurs étapes pour accomplir une tâche
- Il peut analyser ses propres résultats, ou les faire analyser par un autre agent, pour les améliorer
- Le plus souvent il utilise des outils (voir cours LM), d'autres agents et des ressources (bases de données, APIs, applications externes) pour obtenir les informations nécessaires pour accomplir une tâche
- Il peut utiliser une mémoire persistante pour apprendre de ses actions passées et s'améliorer



5

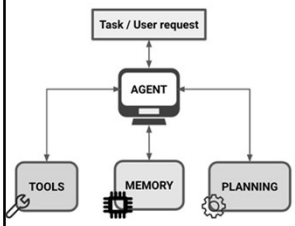
## Cas d'utilisation

- Augmentation de la productivité en automatisant des tâches répétitives
- Organisation complète d'un voyage, avec analyse des coûts, comparaison des différents choix, réservation des billets et du logement
- Réponses automatiques à des emails, avec transmission à des humains pour ceux qui nécessitent une décision qu'il ne peut pas prendre



6

## Composants




- **Planning** : analyser un problème et trouver les étapes pour le résoudre
- **Tools** : utiliser les outils et les ressources dont il a besoin
- **Memory** : mémoire persistante pour apprendre de ses actions passées et s'améliorer



7

## Développer des agents

- Il est possible de développer soi-même des agents (en Java ou autre langage) en utilisant en particulier des outils (MCP, `@Tool`, `@Agent` de LangChain4j par exemple)
- Ou d'utiliser un framework ou une plateforme dédiée ; certains de ces produits sont « no-code » et peuvent être utiles pour faire des prototypes
- Quelques outils qui peuvent aider : Autogen, LangGraph, Superagent (no-code), CrewAI, OpenAgents (payant), LangChain, LangGraph, (adaptation LangGraph4j), Semantic Kernel, Haystack



8

## MCP (Model Context Protocol)

Présentation
MCP et Java
MCP et LangChain4j



9

## Présentation MCP

- Protocole de communication standard qui permet aux applications/agents IA d'accéder à des outils ou ressources externes
- Par exemple pour accéder aux magasins de données (Postgres, …), aux espaces de développement (GitHub, …) et diverses APIs (Gmail, Google Drive, Puppeeter, …)
- Créé par Anthropic (Claude) et adopté par de nombreux autres acteurs de l'IA : OpenAI, Codeium, Replit, Google (Gemini), Apollo, Cursor, …



10

## Acteurs pour MCP

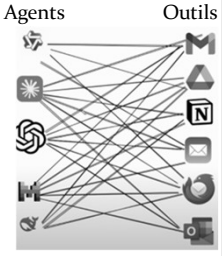
- **Application/agent IA**, par exemple chatGPT ou Claude Desktop
- **LM** utilisé par l'agent IA
- **Outils ou ressources externes** qui pourraient être utiles au LM pour prendre des décisions ou lancer des actions, par exemple site pour la météo, serveur d'emails
- Le LM est le « cerveau » de l'application ; pour donner une réponse ou prendre une décision, il peut avoir besoin de faire exécuter des outils ou de consulter des ressources externes



11

## Problème avec les outils

- De nombreux LMs pouvaient déjà communiquer avec des outils externes pour consulter le Web, envoyer des emails, se connecter à GitHub, …
- Le problème était que chaque LM avait sa propre façon de faire (par exemple, fonctions OpenAI) ; pas de standard pour accéder à un outil
- Pour chaque outil, les développeurs d'un agent IA devaient donc écrire du code particulier






12

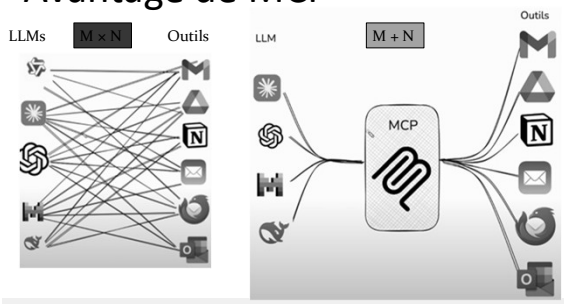
## Pourquoi MCP ?

- Il standardise le protocole de communication entre un agent IA et un outil externe, principalement
  - types et formats des messages échangés
  - métadonnées pour décrire les capacités des outils
  - possibilité pour un agent d'utiliser automatiquement les capacités d'un outil





## Avantage de MCP




Source images : https://www.youtube.com/watch?v=9RV5gttT6rA





## La solution MCP (1/2)

- MCP fonctionne sur le mode client - serveur
- Chaque outil est associé à un serveur MCP (souvent écrit par le fournisseur de l'outil) et l'accès à ce serveur par un client MCP est standardisé par le protocole
- Le serveur peut fonctionner en local ou à distance
- Il permet de décrire les capacités de l'outil et de les utiliser
- Pour utiliser un outil, un agent IA n'a qu'à implémenter un client MCP pour se connecter au serveur MCP de l'outil





## La solution MCP (2/2)

- Le plus souvent un serveur MCP est déjà écrit par le fournisseur de l'outil (version « officielle »), ou bien est déjà disponible sur Internet en version non officielle
- Il existe des librairies (ou frameworks) simples à utiliser pour écrire un client MCP
- Pour utiliser l'outil, le développeur d'un agent IA n'a qu'à
  - déclarer le serveur MCP (souvent un fichier de configuration ou quelques lignes de code)
  - et décider quand utiliser l'outil (pseudo-code « si la question porte sur ***, utiliser les outils **** ») et comment traiter la réponse du LM





## Exemple avec LangChain4j (1/2)

```
McpTransport transport =
    new StreamableHttpMcpTransport.Builder()
        .url("https://api.githubcopilot.com/mcp/")
        .customHeaders(Map.of("Authorization", "Bearer "
                        + githubToken))
        .build();

try (McpClient githubMcpClient =
         new DefaultMcpClient.Builder()
            .transport(transport)
            .build()) {
  ChatModel model = GoogleAiGeminiChatModel.builder()
      .apiKey(System.getenv("GEMINI_KEY"))
      .modelName("gemini-2.5-flash")
      .build();
```





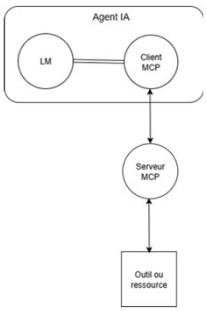
## Exemple avec LangChain4j (2/2)

```
  ToolProvider toolProvider =
      McpToolProvider.builder()
          .mcpClients(List.of(githubMcpClient))
          .build();

  Assistant assistant =
      AiServices.builder(Assistant.class)
          .chatModel(model)
          .chatMemory(chatMemory)
          .toolProvider(toolProvider)
          .build();

    conversationAvec(assistant);
}
```

## En résumé




- Agent IA implémente un client MCP
- qui communique par le protocole MCP avec un serveur MCP
- pour utiliser un outil ou une ressource externe
- qui sera disponible pour le LM



19

## Processus

1. Dans une phase d'initialisation les outils et ressources qui pourraient être utiles à un LM sont demandées au serveur MCP ; l'information est transmise au LM
2. Si le LM a besoin du résultat de l'exécution d'un outil, ou d'une ressource, il demande à l'agent IA ce dont il a besoin
3. L'agent, client MCP, fait appel au serveur MCP qui exécute l'outil et lui fournit le résultat qu'il demande
4. L'agent transmet au LM qui peut répondre en tenant compte de l'exécution de l'outil



20

## Sécurité

- Un serveur MCP peut lancer des exécutions et exposer des données (ressources) ; il faut donc vérifier que le client MCP a bien les autorisations pour cela
- OAuth 2.1 permet à un utilisateur de déléguer au serveur MCP l'accès à ses ressources, sans lui communiquer ses identifiants
- Par exemple, si on veut utiliser un serveur MCP pour Gmail pour faire gérer des emails par une IA, il faudra tout d'abord aller sur le site de Google pour configurer OAuth pour l'utilisation de Gmail



21

## Listes de serveurs MCP

- https://mcpservers.org
- https://glama.ai/mcp/servers
- https://smithery.ai
- https://mcp.so
- https://www.pulsemcp.com
- https://portkey.ai/mcp-servers



22

## Références



23

## Agents (1/2)

- Introduction : https://www.youtube.com/watch?v=F8NKVhkZZWI
- RAG avec agent : https://www.youtube.com/watch?v=0z9_MhcYvcY
- https://towardsdatascience.com/?s=autonomous+agents ; en particulier un agent qui parcourt le Web pour y extraire des informations : https://towardsdatascience.com/building-visual-agents-that-can-navigate-the-web-autonomously-1184efbfe895/



24

## Agents (2/2)

- Agents avec LangChain4J : https://docs.langchain4j.dev/tutorials/agents



25

## MCP

- Protocole MCP : https://modelcontextprotocol.io/
  - Get started : https://modelcontextprotocol.io/docs/getting-started/intro
  - Les SDKs (TypeScript, Python, Java, …): https://modelcontextprotocol.io/docs/sdk
  - MCP Inspector (pour tester et mettre au point les serveurs MCP) : https://modelcontextprotocol.io/docs/tools/inspector
- SDK Java pour client MCP : https://github.com/modelcontextprotocol/java-sdk



26