AI Agents

1. **Định nghĩa**

   AI Agent (Trí tuệ nhân tạo tự chủ) là một hệ thống máy tính có khả năng tự động quan sát môi trường, đưa ra quyết định và hành động để đạt được mục tiêu cụ thể.

2. **Đặc điểm cốt lõi**

   - **Tự chủ(Autonomy) :** AI Agent có thể hoạt động mà không cần sự can thiệp liên tục của con người, dựa trên dữ liệu đầu vào và mục tiêu đã định

   - **Nhận thức môi trường (Perception)**: Thu thập thông tin từ môi trường thông qua cảm biến (sensor), dữ liệu (text, hình ảnh, âm thanh) hoặc API.

   - **Hành động (Action):** Tác động lại môi trường bằng cách đưa ra phản hồi (output)**,** điều khiển thiết bị, hoặc giao tiếp với hệ thống khác.

   - **Mục tiêu (Goal-directed)**: Hành động hướng đến một kết quả cụ thể

3. **AI Agent và Large Language Model (LLM)**

   AI Agent và LLM có điểm chung là đều sử dụng trí tuệ nhân tạo, nhưng chúng khác nhau về mục đich, khả năng và cách hoạt động.

Khác biệt chính:

Tiêu chí	AI Agent	LLM (Large Language Model)
Định nghĩa	Hệ thống tự động nhận thức môi trường, ra quyết định và hành động.	Mô hình AI chuyên xử lý và tạo văn bản dựa trên dữ liệu ngôn ngữ.
Tự chủ	Có thể hoạt động độc lập không cần con người.	Chỉ phản hồi khi có đầu vào (prompt).
Hành động	Tác động vật lý/số (gửi email, điều khiển robot).	Chỉ tạo văn bản, không thực thi hành động bên ngoài.
Công nghệ nền	Kết hợp nhiều AI: Machine Learning, NLP, Robotics, API.	Tập trung vào xử lý ngôn ngữ (NLP).
Ví dụ	Alexa, Tesla Autopilot, chatbot đặt lịch tự động.	ChatGPT, Gemini, Claude.

Mối quan hệ giữa AI Agent và LLM:

- **LLM có thể là một phần của AI Agent:**
  Một AI Agent phức tạp có thể sử dụng LLM như "bộ não ngôn ngữ" để giao tiếp với con người hoặc phân tích văn bản.
  Ví dụ:
    - Một agent đặt lịch họp dùng LLM để hiểu yêu cầu bằng giọng nói, sau đó kết nối với Google Calender để thực thi.

- **LLM đơn thuần không phải Agent:**
  LLM chỉ xử lí ngôn ngữ, không thể tự động hành động hoặc tương tác với thế giới thực nếu không được tích hợp vào hệ thống lớn hơn.

4. Thành phần và cấu trúc hoạt động

a. Cảm biến(Sensors)

i. Định nghĩa

Là "giác quan" của AI Agent, thu thập dữ liệu từ môi trường vật lí hoặc kỹ thuật số để làm đầu vào cho quá trình xử lí.

ii. Chi tiết

- **Loại cảm biến**
    - Vật lí: Camera (hình ảnh), microphone (ghi âm), cảm biến nhiệt độ,...
    - Kỹ thuật số: API thời tiết, dữ liệu từ Database, tín hiệu từ máy chủ.

- **Ví dụ:**
    - Robot hút bụi: Dùng cảm biến hồng ngoại để tránh vật cản
    - Chatbot hỗ trợ khách hàng: Nhận input từ tin nhắn người dùng.

b. Bộ xử lí (Processor)

i. Định nghĩa:

Là "bộ não" của Agent, sử dụng các thuật toán AI để phân tích dữ liệu từ cảm biến và đưa ra quyết định.

ii. Chi tiết:

- **Công nghệ sử dụng:**
    - Machine Learning: Phân loại dữ liệu, dự đoán
    - NLP(Xử lí ngôn ngữ tự nhiên): Hiểu và tạo văn bản
    - Computer Vision: Nhận diện hình ảnh
    - Rule-Based Systems: Quy tắc cứng (Vd : thermostat bật điều hòa khi nhiệt độ trên 30°C)

- **Ví dụ:**
    - Xe tự lái có bộ xử lí kết hợp hình ảnh từ camera + dữ liệu lidar để quyết định rẽ trái/rẽ phải.

c. Bộ tác động (Actuors)

i. Định nghĩa:

Là “cánh tay” của Agent, thực thi hành động dựa trên quyết định từ bộ xử lý.

ii. Chi tiết:

- **Hình thức tác động:**
  - Vật lí: Động cơ robot, màn hình hiển thị, loa phát thanh,...
  - Kĩ thuật số: Gửi email, cập nhật database, gọi API thanh toán,...

- **Ví dụ:**
  - Robot phẫu thuật DaVinci: Actuator điều khiển dụng cụ y tế.
  - Chatbot đặt vé máy bay : Tự động gọi API airline để book vé.

d. Học tập (Learning)

i. Định nghĩa

Khả năng tự cải thiện hiệu suất qua kinh nghiệm (dữ liệu mới hoặc tương tác)

ii. Chi tiết

- **Phương pháp học:**
  - Reinforcement Learning (RL): Học từ thử sai + phần thưởng/phạt(VD : AlphaGo)
  - Online Learning: Cập nhật model liên tục(Vd: chatbot học từ feedback người dùng)
  - Transfer Learning: Tận dụng kiến thức từ tác vụ cũ sang tác vụ mới.

- **Ví dụ:**

- Netflix Recommendation: Tự điều chỉnh gợi ý phim dựa trên lịch sử xem.
- Tesla Autopilot: Cải thiện khả năng lái qua dữ liệu từ hàng triệu xe.

e. Ví dụ thực tế đầy đủ
Agent Smart Home(Google Nest):

1. **Cảm biến:** Microphone nghe lệnh “Tăng nhiệt độ lên 25°C”
2. **Bộ xử lí** : NLP hiểu lệnh -> ML kiểm tra thói quen người dùng
3. **Bộ tác động:** Gửi tín hiệu đến điều hòa.
4. **Học tập:** Ghi nhớ người dùng thích 25°C vào buổi tối

5. Phân loại AI Agent

1. Simple Reflex Agent (Agent phản xạ đơn giản)
   a. Đặc điểm:
      i. Hoạt động dựa trên quy tắc “if - then” cứng nhắc.
      ii. Chỉ phản ứng với tín hiệu đầu vào hiện tại, không quan tâm đến lịch sử hoặc tương lai.
      iii. Không có mô hình về thế giới (World model).

   b. Ví dụ:

      Thermostat:
      - Quy tắc: "Nếu nhiệt độ > 30°C → Bật điều hòa. Nếu < 20°C → Tắt điều hòa."
      - Không "nhớ" trạng thái trước đó hoặc dự đoán nhu cầu người dùng.

2. Model-Based Reflex Agent (Agent dựa trên mô hình)
   a. Đặc điểm :

i. Duy trì mô hình nội bộ về thế giới (world model) để theo dõi trạng thái môi trường.
ii. Ra quyết định dựa trên lịch sử dữ liệu và dự đoán thay đổi.

b. Ví dụ:
i. Xe tự lái:
- Mô hình theo dõi : Vị trí các xe khác, biển báo , tốc độ hiện tại.
- Quyết định: Nếu xe phía trước giảm tốc -> Ước lượng khoảng cách an toàn -> Phanh hoặc đổi làn

3. Goal-Based Agent (Agent hướng mục tiêu)

a. Đặc điểm
i. Hành động để đạt mục tiêu cụ thể (goal)
ii. Sử dụng kế hoạch hành động (planning) và đánh đổi giữa các phương án

b. Ví dụ:
i. Agent lập trình:
1. Mục tiêu: “Tìm đường ngắn nhất từ A đến B, tránh kẹt xe”
2. Hành động: Phân tích giao thông thời gian thực -> So sánh các tuyến -> Chọn đường tối ưu (Google Maps)

4. Utility-Based Agent (Agent dựa trên độ hữu ích)

a. Đặc điểm:
i. Không chỉ hướng dẫn đến mục tiêu, mà còn tối ưu hóa “độ hữu ích” (utility function)
ii. Đánh giá kết quả qua hàm tiện ích (Ví dụ : lợi nhuận, thời gian, độ chính xác)

b. Ví dụ:
    i. Hệ thống gợi ý sản phẩm:
        1. Mục tiêu: Tăng doanh thu
        2. Utility function: “Ưu tiên sản phẩm có tỷ lệ chuyển đổi cao + lợi nhuận lớn”
        3. Hành động: Hiển thị sản phẩm A thay vì B dù là cả hai đều phù hợp với người dùng.

5. Learning Agent (Agent học tập)

a. Đặc điểm:
    i. Tự cải thiện hiệu suất qua kinh nghiệm (dữ liệu mới hoặc tương tác)
    ii. Gồm 4 thành phần:
        1. Performance Element : Phần tử thực thi nhiệm vụ (Vd : chatbot trả lời câu hỏi).
        2. Critic: Đánh giá hiệu suất (Vd: Tỉ lệ hài lòng người dùng)
        3. Learning Element: Điều chỉnh model dựa trên feedback
        4. Program Generator: Tạo tình huống mới để học
b. Ví dụ:
    i. Chatbot hỗ trợ khách hàng:
        1. Ban đầu trả lời sai câu hỏi về chính sách đổi trả.
        2. Critic nhận feedback tiêu cực -> Learning Element cập nhật kiến thức -> Lần sau trả lời chính xác

Bảng tổng hợp so sánh:

Loại Agent	Ưu điểm	Nhược điểm	Ứng dụng phù hợp
Simple Reflex	Tốc độ nhanh, dễ triển khai.	Không xử lý được bối cảnh phức tạp.	Thiết bị IoT đơn giản (đèn tự bật).
Model-Based	Theo dõi được trạng thái động.	Tốn tài nguyên tính toán.	Xe tự lái, hệ thống giám sát.
Goal-Based	Giải quyết bài toán có mục tiêu rõ ràng.	Khó áp dụng khi mục tiêu mơ hồ.	Lập kế hoạch, logistics.
Utility-Based	Tối ưu hóa đa tiêu chí.	Thiết kế hàm utility phức tạp.	Gợi ý sản phẩm, quản lý tài nguyên.
Learning Agent	Tự thích nghi, linh hoạt.	Rủi ro học sai/học lệch.	Chatbot, hệ thống recommendation.

6. Kiến trúc Multi-Agent Systems (MAS)

Multi-Agent Systems là hệ thống bao gồm nhiều AI Agent tương tác với nhau để giải quyết các vấn đề phức tạp:

Đặc điểm:

- **Phân phối tính toán**: Chia nhỏ tác vụ lớn thành các tác vụ con
- **Giao tiếp giữa các Agent**: Chia sẻ thông tin, phối hợp hành động
- **Đàm phán và hợp tác**: Các agent có thể thương lượng để đạt mục tiêu chung
- **Khả năng chịu lỗi**: Nếu một agent gặp sự cố, các agent khác vẫn có thể hoạt động

Ví dụ thực tế:

- **Hệ thống giao dịch tài chính**: Nhiều agent theo dõi thị trường, phân tích rủi ro, thực hiện giao dịch
- **Quản lý chuỗi cung ứng**: Agent theo dõi kho hàng, dự báo nhu cầu, tối ưu hóa vận chuyển

Các thành phần cốt lõi

1. Individual Agents (Các Agent riêng lẻ)

Cấu trúc mỗi Agent:

- **Knowledge Base**: Cơ sở tri thức riêng của agent
- **Reasoning Engine**: Bộ máy suy luận để đưa ra quyết định
- **Communication Module**: Module giao tiếp với các agent khác
- **Action Executor**: Bộ thực thi hành động
- **Learning Component**: Thành phần học tập và cải thiện

Đặc điểm:

- Tự chủ trong việc đưa ra quyết định
- Có mục tiêu riêng (có thể trùng hoặc khác với mục tiêu hệ thống)
- Khả năng học tập và thích nghi

2. Communication Infrastructure (Hạ tầng giao tiếp)

Message Passing System:

- **Message Queue**: Hàng đợi tin nhắn giữa các agent
- **Routing Mechanism**: Cơ chế định tuyến tin nhắn
- **Protocol Standards**: Các chuẩn giao thức (như FIPA-ACL, KQML)

Communication Patterns:

- **Point-to-Point**: Giao tiếp trực tiếp giữa hai agent
- **Broadcast**: Một agent gửi tin nhắn đến tất cả agent khác
- **Multicast**: Gửi tin nhắn đến một nhóm agent cụ thể
- **Publish-Subscribe**: Mô hình đăng ký nhận thông tin

3. Environment (Môi trường)

Shared Environment:

- **Global State**: Trạng thái chung của hệ thống
- **Resources**: Tài nguyên chia sẻ giữa các agent
- **Constraints**: Các ràng buộc môi trường

Environment Types:

- **Static Environment**: Môi trường không thay đổi
- **Dynamic Environment**: Môi trường thay đổi theo thời gian
- **Partially Observable**: Agent chỉ thấy một phần môi trường

4. Coordination Mechanisms (Cơ chế phối hợp)

Coordination Strategies:

- **Contract Net Protocol**: Đấu thầu công việc giữa các agent
- **Auction Mechanisms**: Cơ chế đấu giá tài nguyên
- **Voting Systems**: Hệ thống bỏ phiếu để đưa ra quyết định chung
- **Consensus Algorithms**: Thuật toán đồng thuận

Negotiation Protocols:

- **Bilateral Negotiation**: Đàm phán giữa hai agent

- **Multi-party Negotiation**: Đàm phán nhiều bên
- **Mediated Negotiation**: Đàm phán có trung gian

5. Organization Structure (Cấu trúc tổ chức)

Hierarchical Structure:

- **Manager Agents**: Agent quản lý, điều phối
- **Worker Agents**: Agent thực hiện công việc cụ thể
- **Broker Agents**: Agent trung gian, môi giới

Network Structure:

- **Peer-to-Peer**: Tất cả agent có quyền bình đẳng
- **Hub-and-Spoke**: Một agent trung tâm kết nối với các agent khác
- **Mesh Network**: Mỗi agent có thể kết nối với nhiều agent khác

Kiến trúc phân lớp

1. Application Layer (Lớp ứng dụng)

- **Domain-Specific Agents**: Agent chuyên biệt cho từng lĩnh vực
- **User Interface Agents**: Agent giao diện người dùng
- **Integration Agents**: Agent tích hợp với hệ thống bên ngoài

2. Coordination Layer (Lớp phối hợp)

- **Coordination Protocols**: Các giao thức phối hợp
- **Task Allocation**: Phân bổ nhiệm vụ
- **Resource Management**: Quản lý tài nguyên

3. Communication Layer (Lớp giao tiếp)

- **Message Routing**: Định tuyến tin nhắn
- **Protocol Translation**: Dịch giao thức
- **Security & Authentication**: Bảo mật và xác thực

4. Infrastructure Layer (Lớp hạ tầng)

- **Runtime Environment**: Môi trường thực thi
- **Resource Pool**: Pool tài nguyên hệ thống

- **Monitoring & Logging**: Giám sát và ghi log

Model Context Protocol (MCP)

1. Định nghĩa

- Model Context Protocol (MCP) là một giao thức hoặc framework được thiết kế để quản lí ngữ cảnh mô hình.
- MCP có thể hình dung như một cầu nối giao tiếp 2 chiều giữa mô hình AI và các công cụ, dịch vụ và nguồn dữ liệu bên thứ 3.
- MCP giúp mô hình AI truy cập dữ liệu , hiểu rõ hơn về ngữ cảnh, từ đó phản hồi yêu cầu của người dùng chính xác và thông minh hơn.

2. Tại sao MCP lại quan trọng đối với mô hình AI

Nếu không có MCP thì các mô hình AI chỉ có thể sử dụng những gì chúng học được qua quá trình huấn luyện. Điều này có nghĩa là chúng không thể:

- Truy cập những nguồn thông tin hiện tại từ internet
- Lấy dữ liệu từ cơ sở dữ liệu để trả lời câu hỏi cụ thể
- Sử dụng các dịch vụ chuyên biệt chẳng hạn như xử lý video
- Lưu thông tin vào các tệp
- Kết nối với các công cụ bên ngoài để mở rộng khả năng của chúng

MCP sẽ giúp AI khắc phục các hạn chế này bằng cách kết nối mô hình AI với các công cụ và dịch vụ bên thứ 3, biến AI từ một hệ thống tách biệt thành ứng dụng mang tính kết nối , có thể giải quyết nhiều vấn đề phức tạp của người dùng
Ví dụ:

- Mô hình AI nhận được một URL Youtube

- MCP sẽ kết nối với dịch vụ lấy transcript từ Youtube
- Dịch vụ trả lại transcript cho mô hình AI
- Mô hình AI tóm tắt nội dung transcript
- Thông qua một dịch vụ MCP khác, mô hình AI lưu bản tóm tắt vào 1 tệp

Có thể thấy nếu không có MCP thì mỗi bước trong quy trình trên sẽ cần có code tùy chỉnh và tích hợp riêng sẽ rất phức tạp và tốn tài nguyên.

3. Cách MCP hoạt động

MCP vận hành dựa trên mô hình client-host-server, cho phép các ứng dụng kết nối cùng lúc với nhiều tài nguyên khác nhau.Cấu trúc này bao gồm 3 phần chính:

- MCP Host: Thường là chatbot, IDE hoặc công cụ AI khác, đóng vai trò như bộ điều phối trung tâm trong ứng dụng. Host chịu trách nhiệm quản lý từng phiên Client, kiểm soát quyền truy cập và chính sách bảo mật

- MCP Client: Có thể là ứng dụng di động hoặc web, được khởi tạo bởi MCP Host. Client kết nối với một Server duy nhất và xử lí giao tiếp hai chiều giữa Host và Server.

- MCP Server: Kết nối với các nguồn dữ liệu hoặc công cụ và cung cấp các khả năng cụ thể.Cung cấp dữ liệu cho mô hình AI thông qua 3 phương thức cơ bản
    - Prompt : Các mẫu lệnh được xác định trước cho mô hình ngôn ngữ lớn (LLM) có thể sử dụng dễ dàng thông qua lệnh gạch chéo.
    - Source: Dữ liệu có cấu trúc như tệp, cơ sở dữ liệu hoặc lịch sử hoạt động, giúp bổ sung ngữ cảnh cho LLM
    - Tool: Các hàm (functions) cho phép mô hình thực hiện hành động, giúp bổ sung ngữ cảnh cho LLM.

4. MCP SDK - Bộ công cụ phát triển

4.1. Các SDK chính thức

MCP hiện có nhiều SDK chính thức cho các ngôn ngữ khác nhau:

- **Python SDK**: SDK Python chính thức triển khai đầy đủ đặc tả MCP
- **TypeScript SDK**: Hỗ trợ phát triển client và server
- **Java SDK**: Cung cấp triển khai Java của MCP với các pattern giao tiếp đồng bộ và bất đồng bộ
- **C# SDK**: Cho phép developers dễ dàng xây dựng cả server và client sử dụng giao thức này

4.2 Tính năng của SDK

SDK cung cấp khả năng xử lý tin nhắn JSON-RPC hai chiều qua các luồng input/output tiêu chuẩn với xử lý tin nhắn không chặn, serialization/deserialization, và hỗ trợ tắt ứng dụng một cách mượt mà.

5. Cấu trúc JSON-RPC 2.0 trong MCP

5.1 JSON-RPC 2.0 cơ bản

JSON-RPC là một giao thức gọi thủ tục từ xa (RPC) nhẹ, không trạng thái. Giao thức này chủ yếu định nghĩa một số cấu trúc dữ liệu và các quy tắc xung quanh việc xử lý chúng.

Tất cả tin nhắn giữa MCP client và server phải tuân theo JSON-RPC 2.0.

5.2 Các loại tin nhắn trong MCP

Giao thức định nghĩa ba loại tin nhắn: Tin nhắn hai chiều có thể được gửi từ client đến server hoặc ngược lại

1. Requests (Yêu cầu) Requests được gửi từ client đến server hoặc ngược lại để khởi tạo một thao tác. Requests PHẢI bao gồm một ID kiểu string hoặc integer.

Cấu trúc request:

```
{
"jsonrpc": "2.0",
"id": "string | number",
"method": "method_name",
"params": { // Tham số của phương thức }
 }
```

2. Responses (Phản hồi) Cấu trúc response bao gồm trường result hoặc error:

```
{
 "jsonrpc": "2.0",
 "id": "string | number",
"result": { // Dữ liệu kết quả }
}
```

Hoặc trong trường hợp lỗi:

```
{
"jsonrpc": "2.0",
"id": "string | number",
 "error": {
```

```
"code": "number", "message": "string","data": "optional"  }
}
```

3. Notifications (Thông báo) Notifications không yêu cầu phản hồi và không có trường id:

```
{
 "jsonrpc": "2.0",
 "method": "notification_method",
"params": { // Tham số thông báo }
}
```

5.3 Ứng dụng và lợi ích

- **Tích hợp dễ dàng**: Giao thức định nghĩa định dạng có cấu trúc cho tin nhắn, dựa trên JSON-RPC 2.0, xử lý các vấn đề như tương quan request/response, báo cáo lỗi và gọi tool.
- **Hệ sinh thái phong phú**: Anthropic đã "dogfood" giao thức này và phát hành với một bộ ban đầu toàn diện bao gồm Claude Desktop làm client và nhiều server tham chiếu.
- **Chuẩn hóa**: Cho phép các ứng dụng AI cung cấp context theo cách chuẩn hóa, tách biệt mối quan tâm về việc cung cấp context khỏi tương tác LLM thực tế.

6. Kiến trúc MCP Server

6.1 Cấu trúc thư mục chuẩn:

```
mcp-server/
├── src/
│   ├── handlers/        # Xử lý các request
│   ├── tools/           # Định nghĩa các công cụ
│   ├── resources/       # Quản lý tài nguyên
│   └── main.py          # Entry point
├── config/
│   └── server.json      # Cấu hình server
```

6.2 Lifecycle Methods:

- **initialize()**: Khởi tạo server và thiết lập kết nối
- **list_tools()**: Trả về danh sách các công cụ có sẵn
- **call_tool()**: Thực thi công cụ được yêu cầu
- **cleanup()**: Dọn dẹp tài nguyên khi đóng kết nối

7.FastMCP SDK

FastMCP SDK là một framework phát triển bởi Anthropic giúp tạo ra các MCP (Model Context Protocol) servers một cách nhanh chóng và dễ dàng. Đây là công cụ quan trọng để mở rộng khả năng của Claude thông qua việc tích hợp với các hệ thống và dịch vụ bên ngoài.

FastMCP SDK cung cấp các công cụ và thư viện để:

1. Tạo MCP Servers nhanh chóng

- Templates có sẵn cho các use cases phổ biến
- Boilerplate code tự động sinh
- Hot reload trong quá trình phát triển

2. Định nghĩa Tools và Resources

- Tools: Các function mà Claude có thể gọi (như API calls, database queries)
- Resources: Các nguồn dữ liệu mà Claude có thể đọc (files, documents, web content)

3. Xử lý Authentication và Security

- Tích hợp OAuth, API keys
- Rate limiting và error handling
- Sandbox môi trường thực thi

Lợi ích chính

- **Nhanh chóng**: Giảm thời gian phát triển từ tuần xuống giờ
- **Tiêu chuẩn hóa**: Tuân thủ MCP protocol specifications
- **Bảo mật**: Built-in security best practices
- **Linh hoạt**: Hỗ trợ nhiều ngôn ngữ lập trình

- **Scalable**: Có thể xử lý từ prototype đến production

FastMCP SDK giúp developers dễ dàng mở rộng khả năng của Claude để làm việc với dữ liệu và hệ thống thực tế của họ, tạo ra những AI assistants thực sự hữu ích trong công việc hàng ngày.

Nguồn:
https://base.vn/blog/model-context-protocol-mcp-la-gi/
https://200lab.io/blog/model-context-protocol-mcp-la-gi?srsltid=AfmBOoo q26uwOmGACojaF5PFYAGgSBzTPUQGCGRVkHaiGOcNoNanDyWA
https://a16z.com/a-deep-dive-into-mcp-and-the-future-of-ai-tooling/
https://www.anthropic.com/news/model-context-protocol
https://claude.ai/