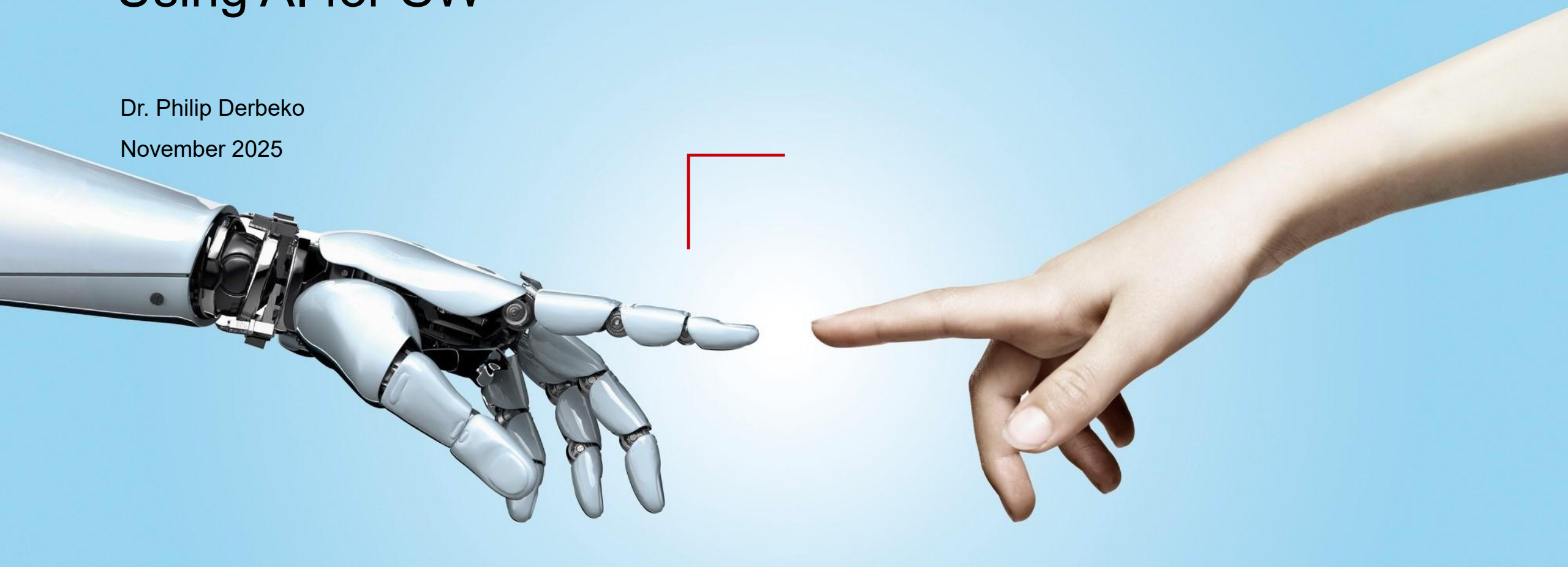
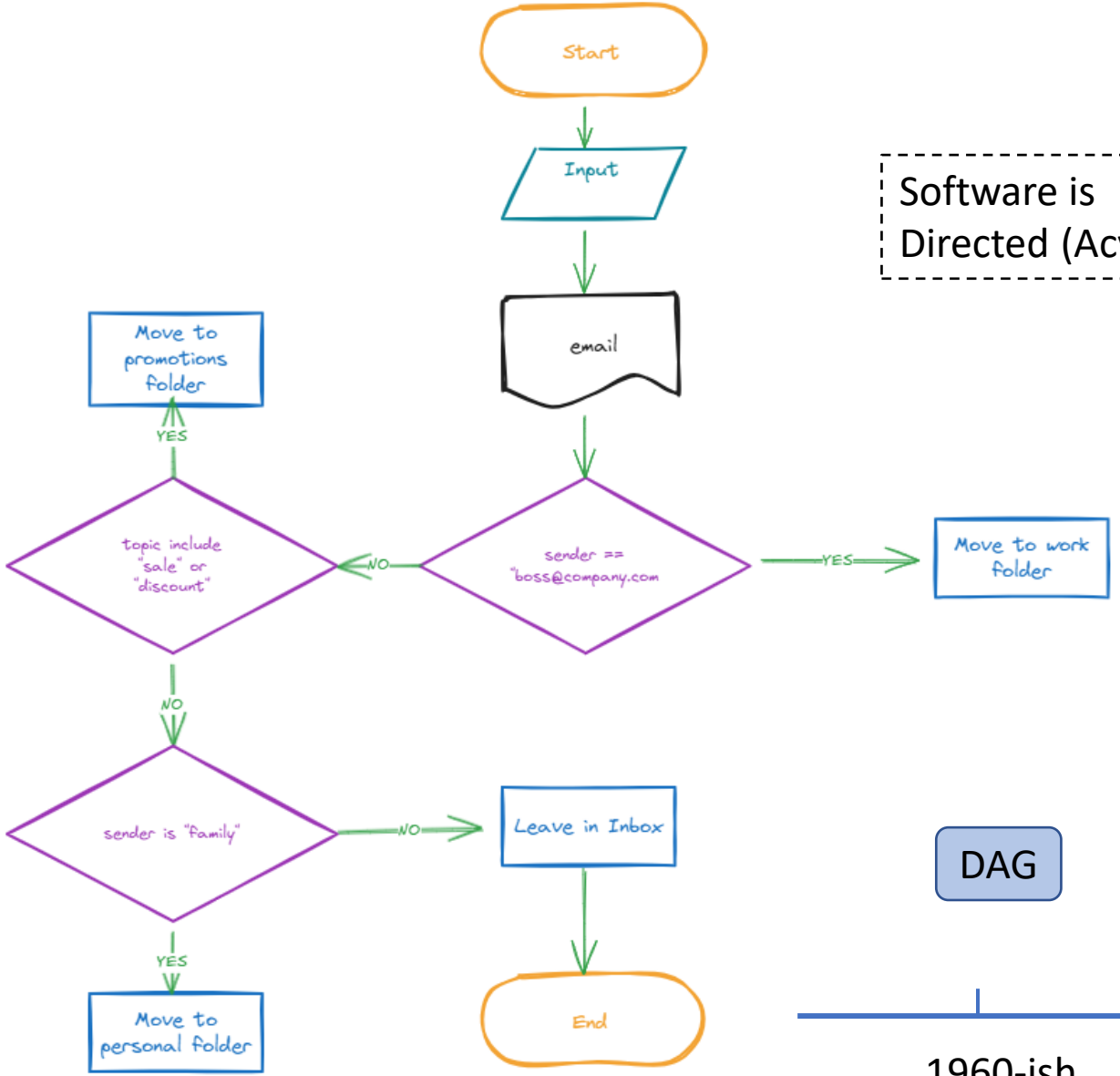
Using AI for SW

Dr. Philip Derbeko

November 2025



Flow Charts



Software is
Directed (Acyclic) Graph - DAG

DAG

1960-ish

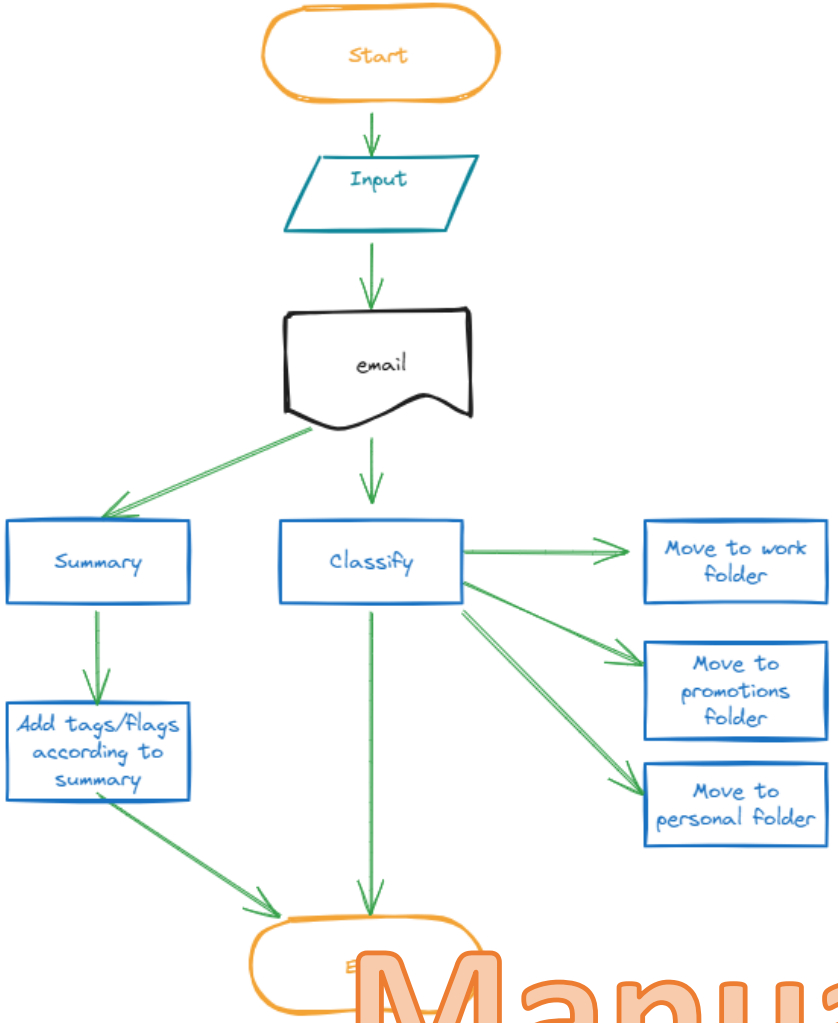
Flow Chart – back to email example

Limitations:

- **Rigid and Static** – new family member, new boss, and other changes will require tweaking of the rules.
- **Scalability** – managing hundreds of rules is hard and rules begin to contradict one another.
- **Extensibility** – extending the flow with new actions or more information is hard.
- **Lack of Context** – the system only looks at a specific characteristic of the message. An important email from Sales department might end up in “promotions” folder.
- **No Learning** – The system does not learn from its mistakes or from its results.

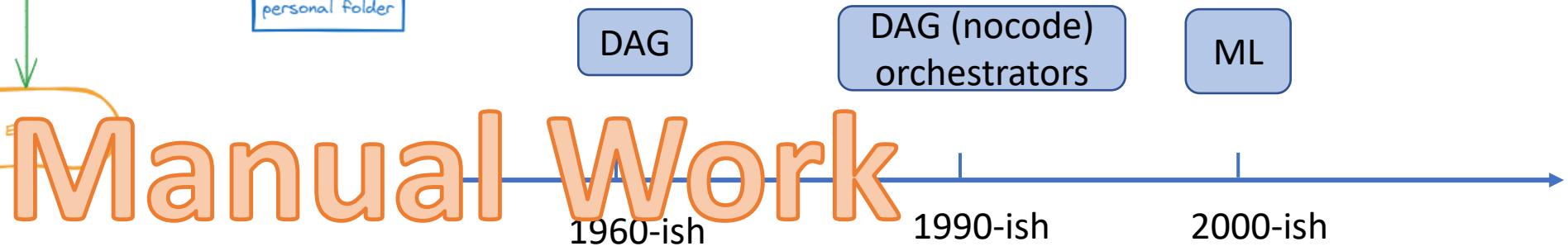
Manual Work

Machine Learning

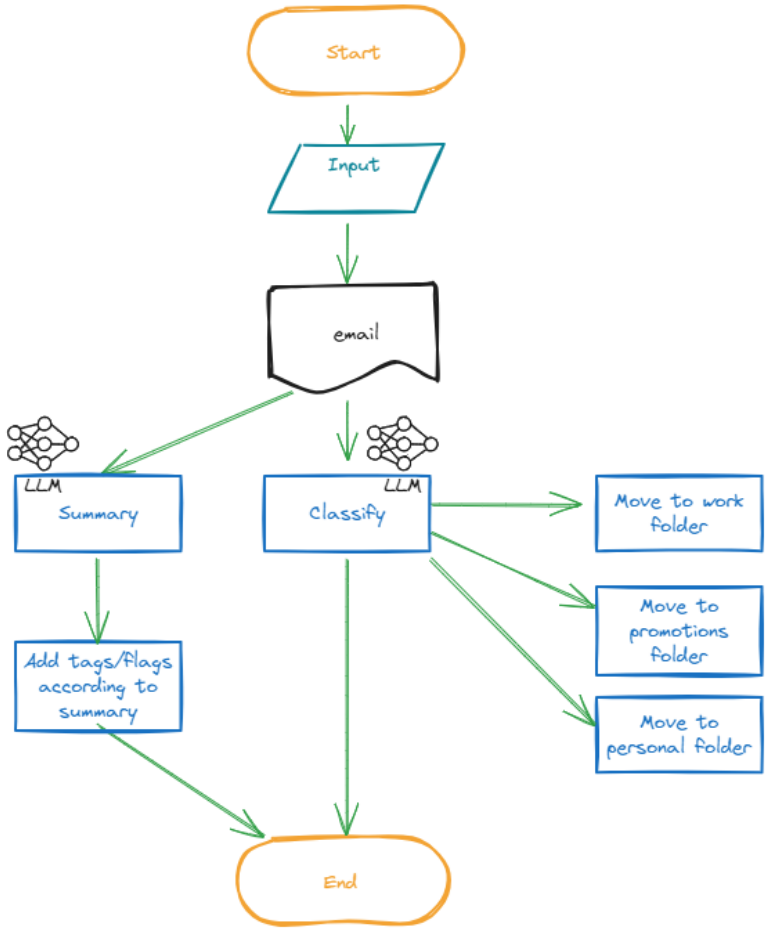


ML enables smarter blocks.
But:

- The flow is still manually defined.
- Reliance on Explicit Features – manually defined
- Lack of semantic understanding
- Susceptible for evasion – senders can trick the system
- Limited scope – each additional function, like summary, suggest actions, etc., has to be manually implemented.
- Changes in system usually require re-training of the ML components.
- Hard to personalize – have to be retrained.

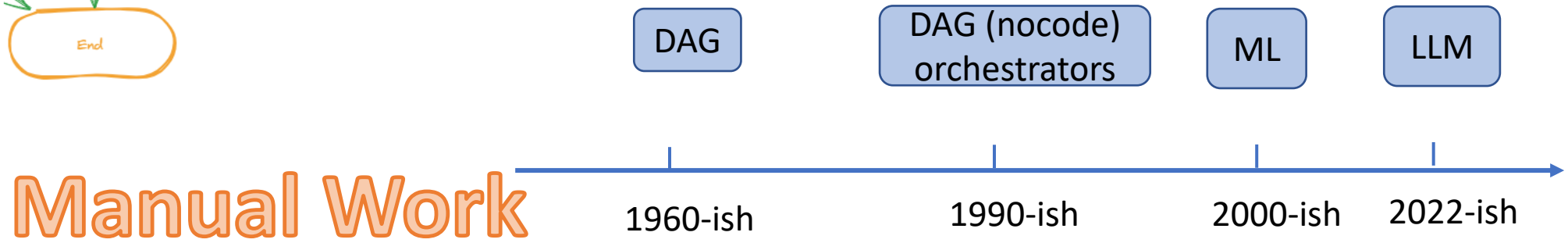


LLM as a building block



LLM is used to perform complex actions.
The flow is still manually defined.
BUT:

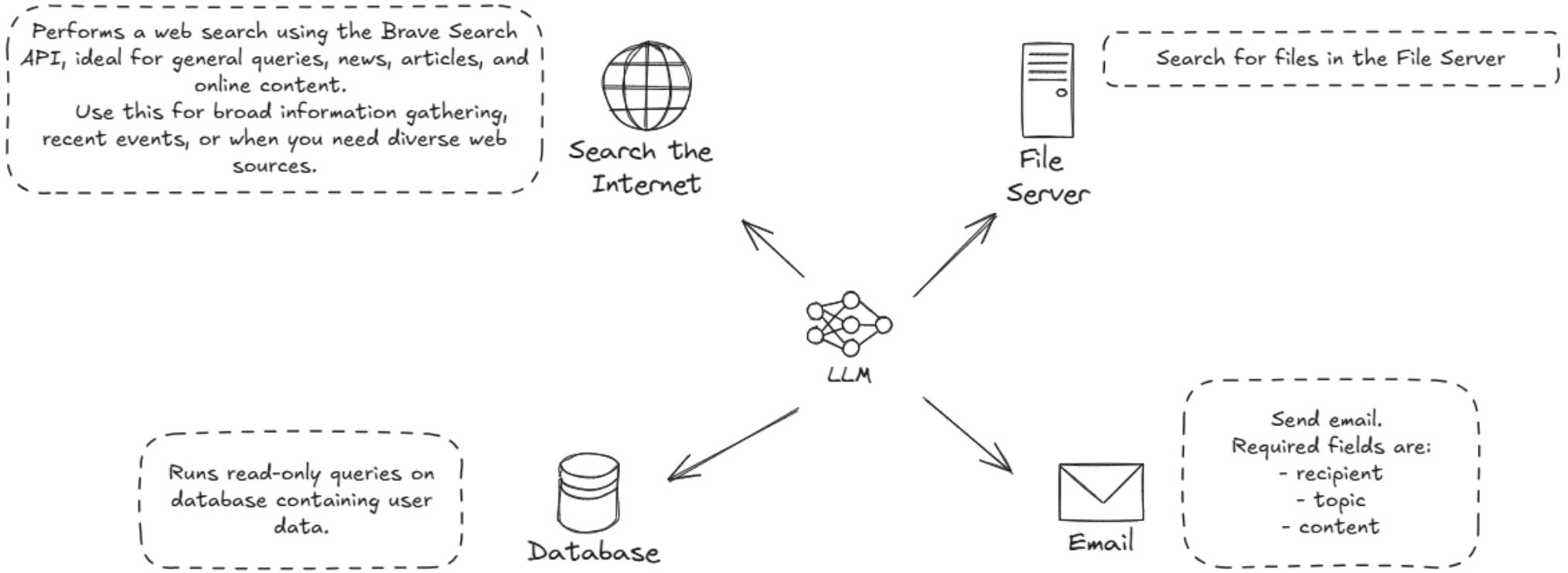
- The building blocks are flexible and extendable
- No need to re-train if a new classification is added
- Easy to personalize – in-context.
- Easy to guide by providing examples (in-context learning)
- Actions are performed with semantic understanding.



Manual Work

Tools

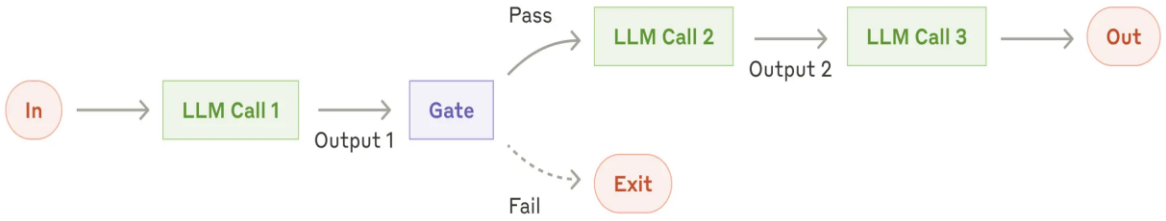
- Tools are any external capability that LLM/Agent can invoke to get information or to act
- Tool provides input data and short info when to use it.
- Usually JSON is used as a data format.



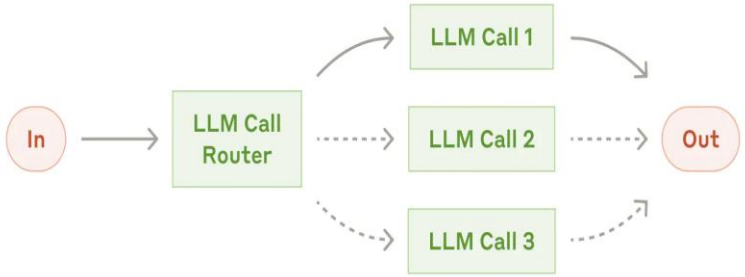
Design Patterns

Shamelessly copied from https://www.anthropic.com/engineering/building-effective-agents

Call Chaining – decomposable task



Routing – different types of tasks, better no overlap



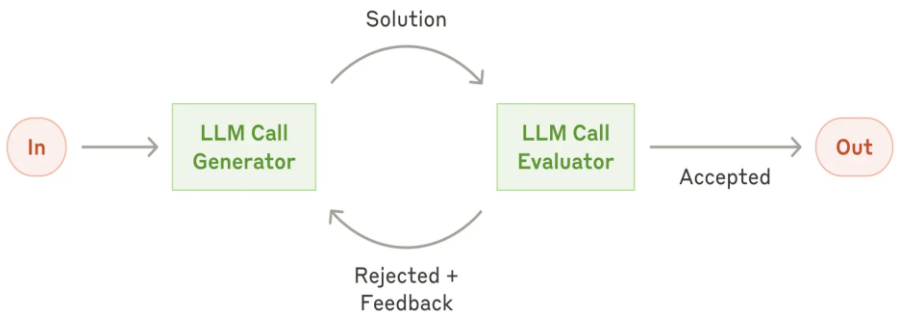
Orchestrator – complex tasks with unsure next step

Design Patterns - 2

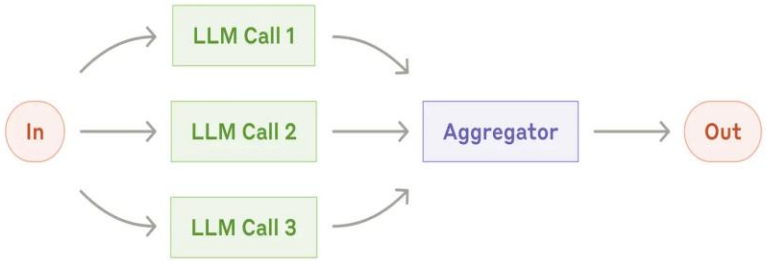
Call Chaining – decomposable task

Shamelessly copied from https://www.anthropic.com/engineering/building-effective-agents

Evaluator – especially good when evaluation is easy

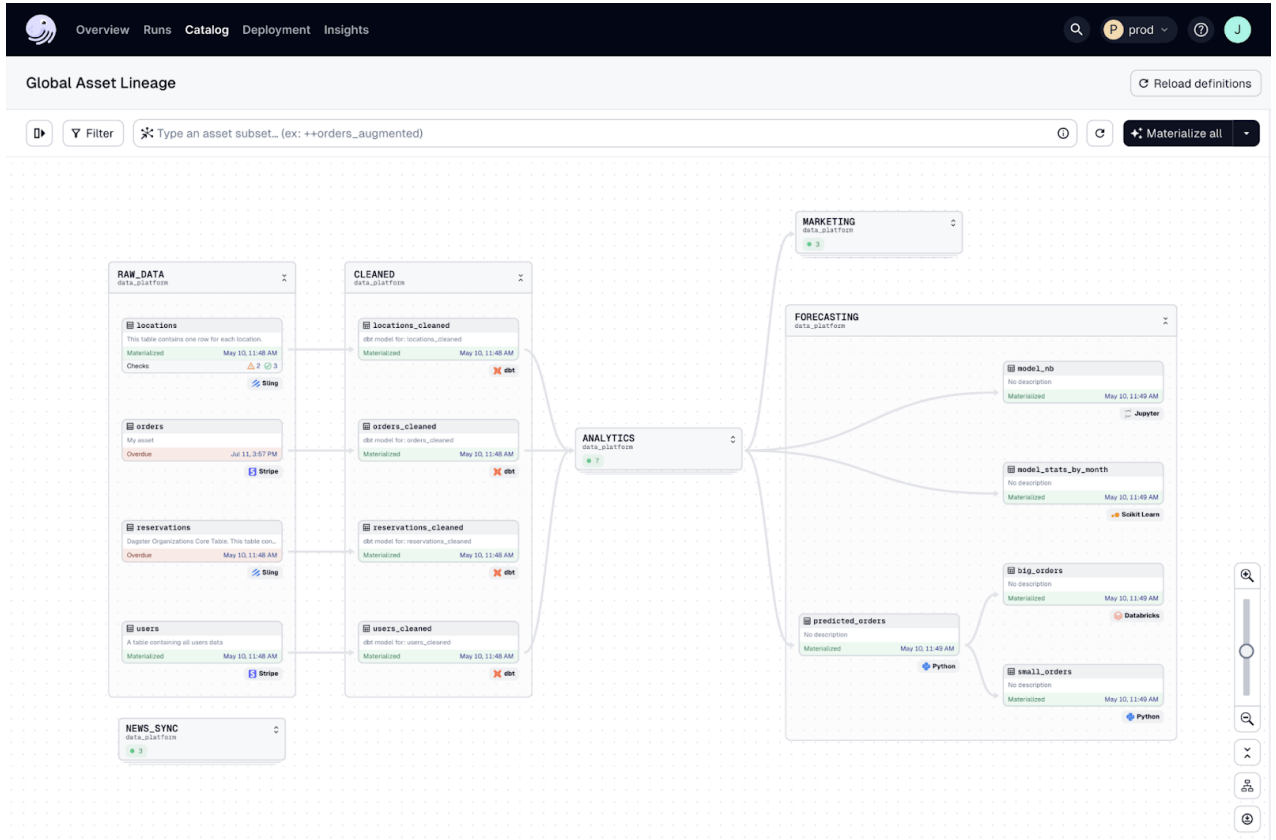


Parallelization – segmentation or majority voting




The End

Flow Chart Automations



Let's automate DAG execution!!!

Still alive and kicking, especially for data flows:

- Apache Airflow
- Prefect
- Dagster
- AWS Step Functions
- …

DAG

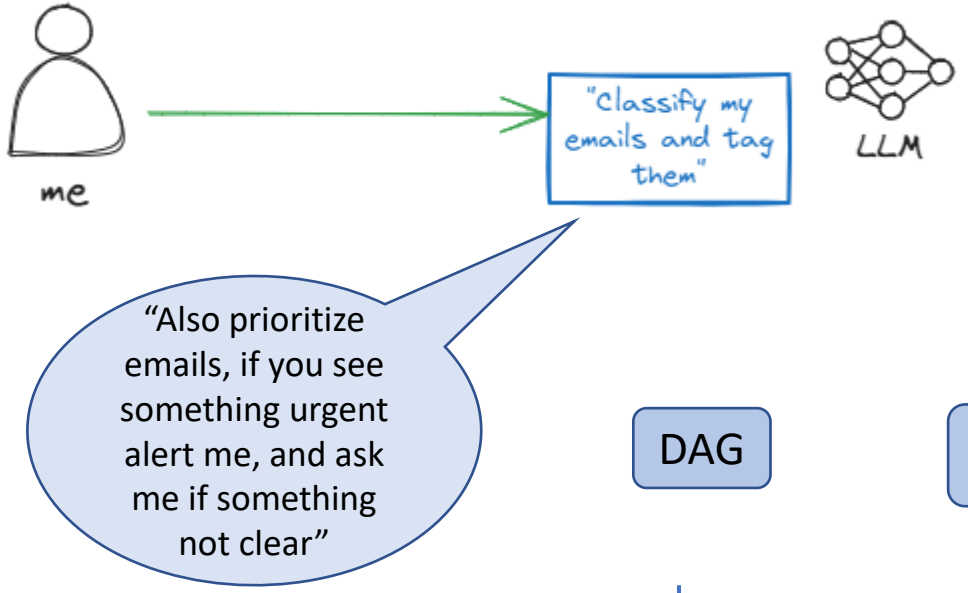
DAG (nocode) orchestrators

1960-ish

1990-ish

Agents

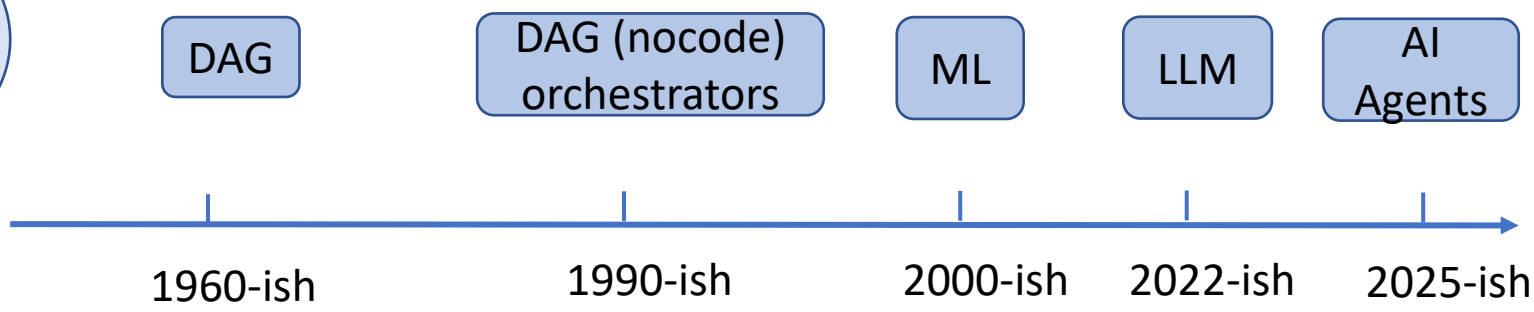
Promise of Agents: No flow charts



No more DAG / Flow Chart.
Let LLM make decisions.

Agent as a loop:

1. LLM determines the next step
2. Code executes the next step action
3. The result is sent to the LLM
4. Until the task is completed, go to 1.



Tools - Example

MCP Server: https://github.com/modelcontextprotocol/servers-archived/tree/main/src/brave-search

```
{
  "mcpServers": {
    "brave-search": {
      "command": "npx",
      "args": [
        "-y",
        "@modelcontextprotocol/server-brave-search"
      ],
      "env": {
        "BRAVE_API_KEY": "YOUR_API_KEY_HERE"
      }
    }
  }
}
```

```
const WEB_SEARCH_TOOL: Tool = {
  name: "brave_web_search",
  description:
    "Performs a web search using the Brave Search API, ideal for general queries, news, articles, and online content. "
    "Use this for broad information gathering, recent events, or when you need diverse web sources. " +
    "Supports pagination, content filtering, and freshness controls. " +
    "Maximum 20 results per request, with offset for pagination. ",
  inputSchema: {
    type: "object",
    properties: {
      query: {
        type: "string",
        description: "Search query (max 400 chars, 50 words)"
      },
      count: {
        type: "number",
        description: "Number of results (1-20, default 10)",
        default: 10
      },
      offset: {
        type: "number",
        description: "Pagination offset (max 9, default 0)",
        default: 0
      },
    },
    required: ["query"],
  },
};
```

```
server.setRequestHandler(CallToolRequestSchema, async (request) => {
  try {
    const { name, arguments: args } = request.params;

    if (!args) {
      throw new Error("No arguments provided");
    }

    switch (name) {
      case "brave_web_search": {
        if (!isBraveWebSearchArgs(args)) {
          throw new Error("Invalid arguments for brave_web_search");
        }
        const { query, count = 10 } = args;
        const results = await performWebSearch(query, count);
        return {
          content: [{ type: "text", text: results }],
          isError: false,
        };
      }
```

Model Content Protocol - MCP

- MCP was introduced by Anthropic in November 2024 as an open source standard. The protocol replaced OpenAI function-calling and ChatGPT plug-in due to its openness.
- In March 2025 OpenAI officially adopted the protocol by adding support in Agents API
- Google announced coming support for MCP in April 2025
- Microsoft announced native MCP support in Windows on May 2025 (Build 2025)
- MS Semantic Kernel added support for MCP in April 2025
- May 2025 – CrewAI added support for MCP
- …