Interaction Design Conference (IxDC) | Beijing | October 31, 2025

The Rise of Design for Agents

Hugh Dubberly

This presentation focuses on
how Large-Language Models (LLMs) can be used in software agents.

Issues beyond our scope and time include:
- ASI (Artificial Super Intelligence)
- AGI (Artificial General Intelligence)
- PoFAI (Plain-old-Fashioned AI)
- GenAI (Generative AI) more broadly

Outline:

PART ONE

Setting the context

PART TWO

Defining agents

PART THREE

Agent design process

PART FOUR

Examples of agents

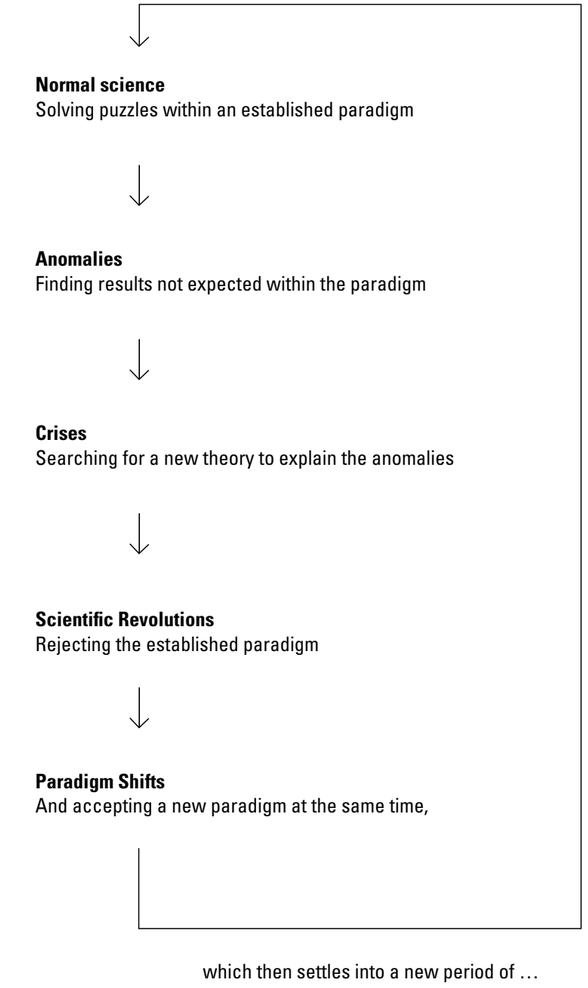
PART ONE

Setting the context

An axiom of design theory might be:

Design practice changes as new technologies emerge.

After a relatively stable, **normal period,**
technological change creates a **“crisis”,**
which results in a **new “paradigm”,**
which then becomes **normalized.**
And the cycle **repeats.**



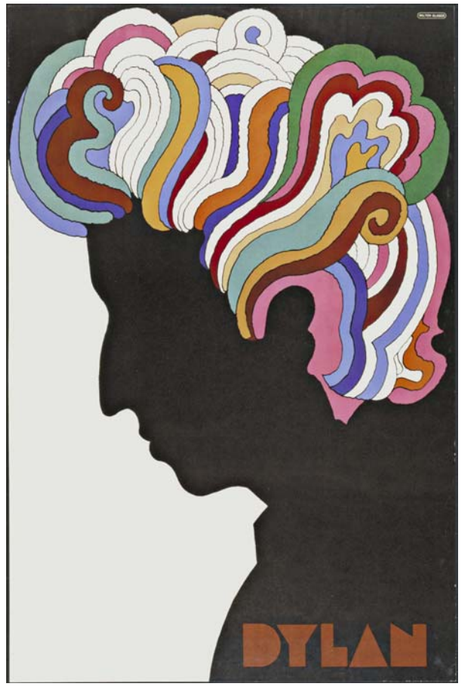
Cf., Thomas Kuhn, *The Structure of Scientific Revolutions,* 1962.

Kuhn Cycle of Scientific Advancement

The first design paradigm, circa ~1900:

Mass manufacturing replaced most hand-craft production.

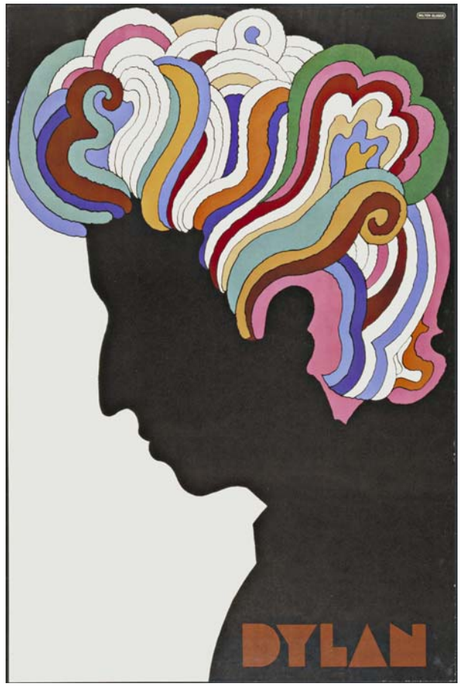
For millennia, design was an integral part of hand-craft production.



Graphic design and product design subsumed most of hand-craft design.

The second design paradigm, circa ~1980:

Computers became a design tool, then media, then material.



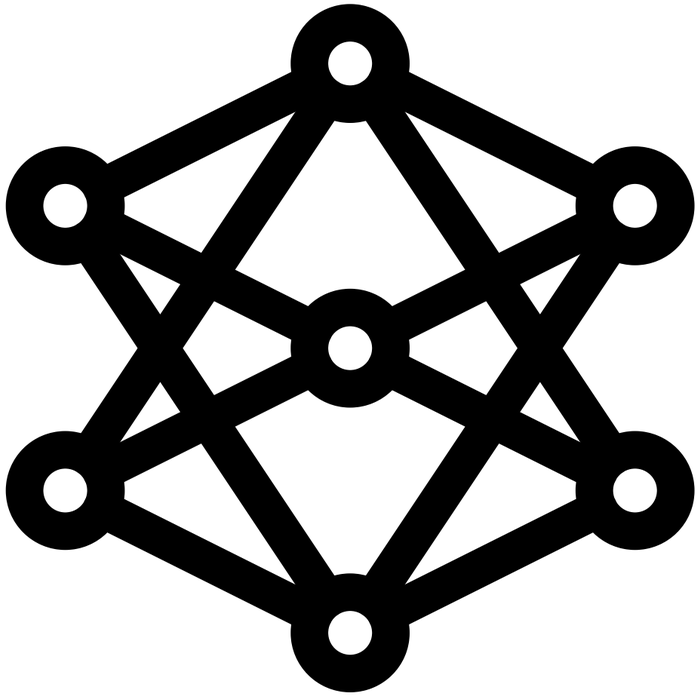
Graphic design and product design subsumed most of hand-craft design.

Interaction design subsumed large parts of graphic design.

The third design paradigm, circa ~2025:

LLMs are not only design tools but also a new material of design.

Interaction design subsumed large parts of graphic design.

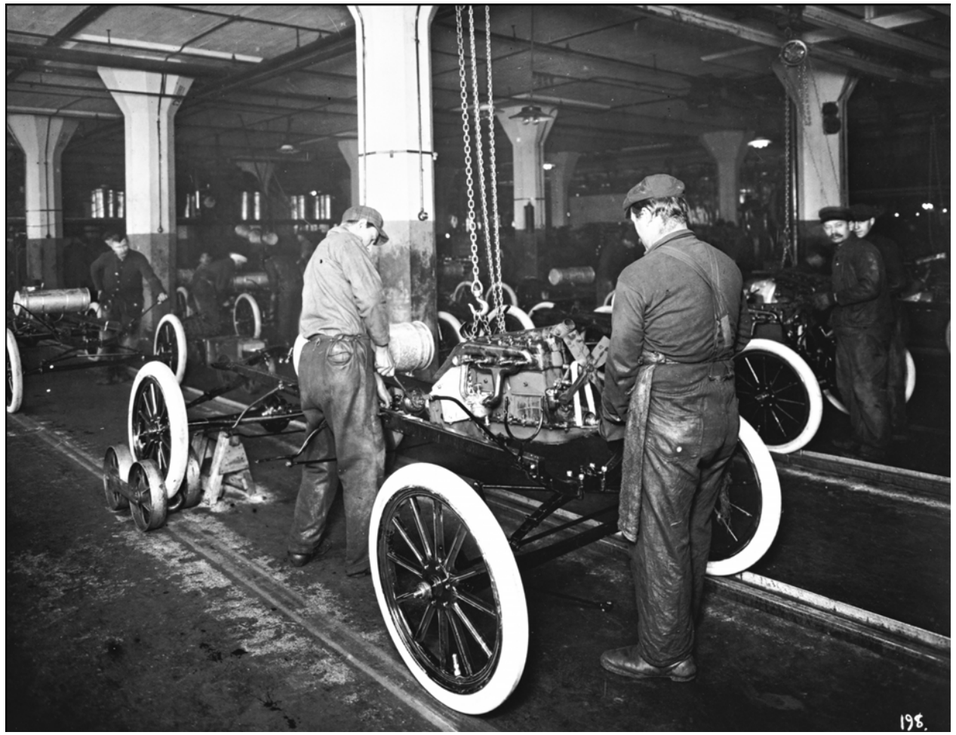


Design for agents will subsume large parts of interaction design.

A corollary is:

The economy has expanded from manufacturing to include services —

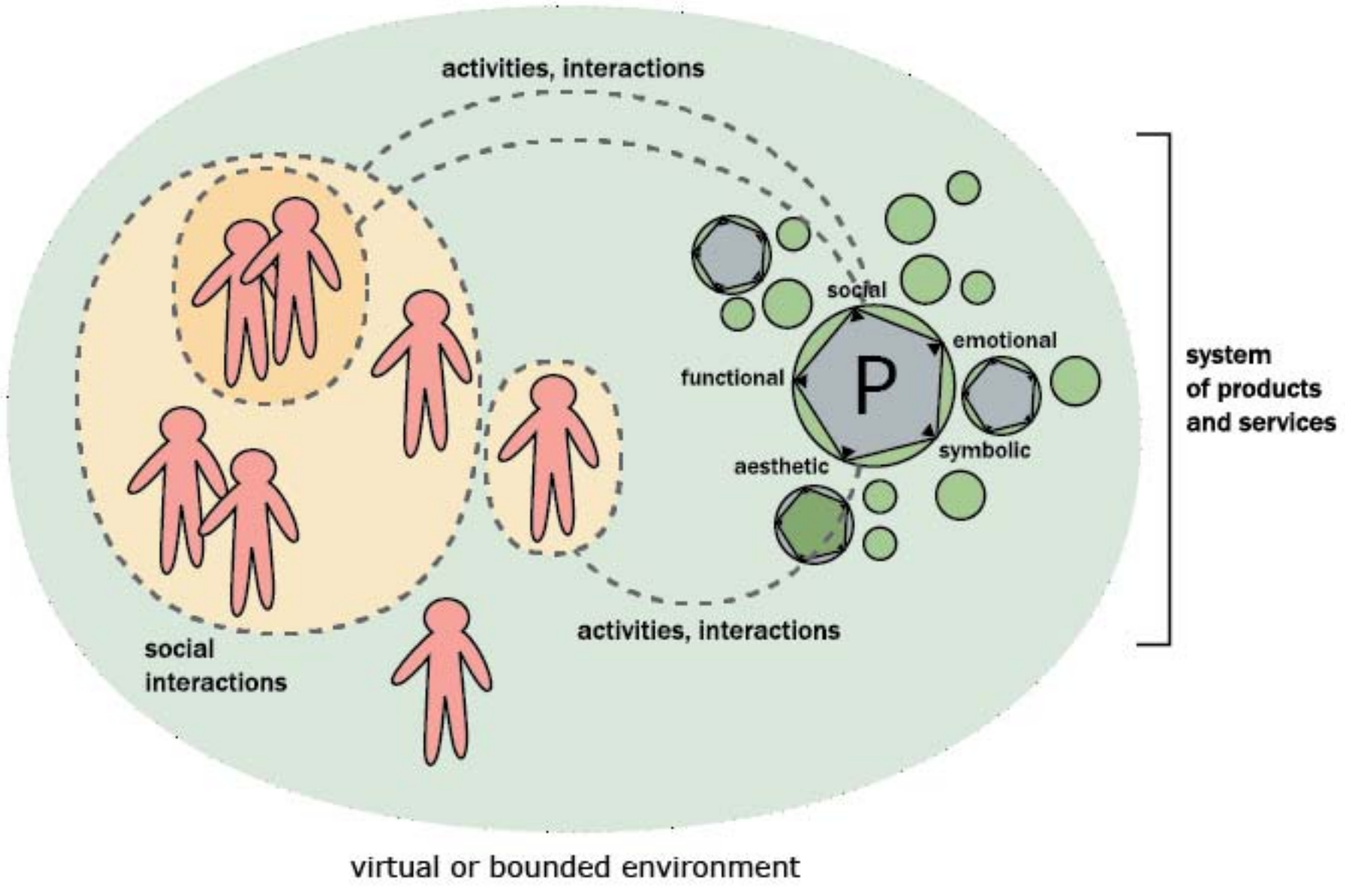
opening another new solution space for design.



Henry Ford sold cars.

General Electric leases jet engine “up-time”.
In other words, services have subsumed products.

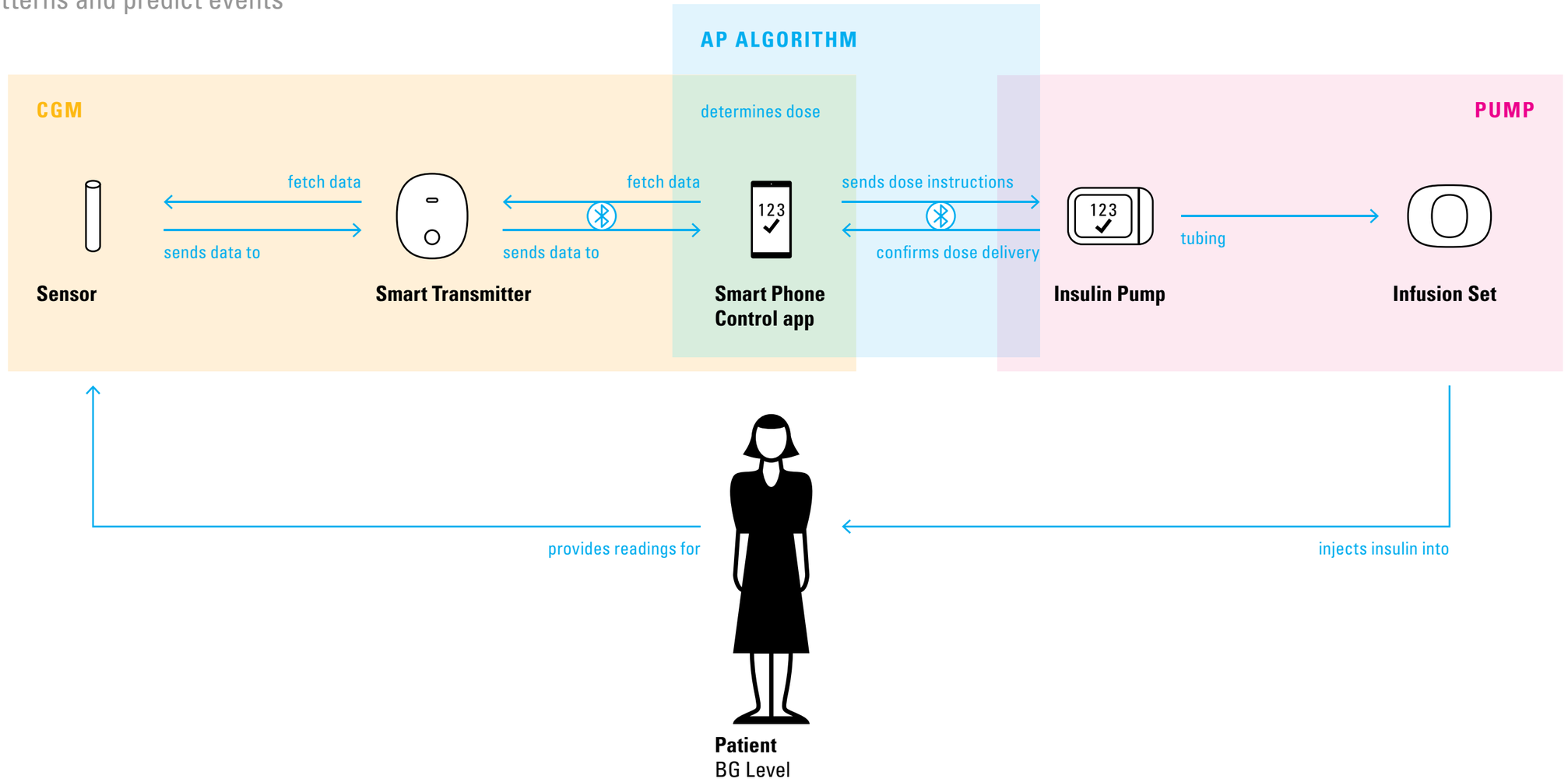
Services exist in networks of systems
or product-service ecologies.



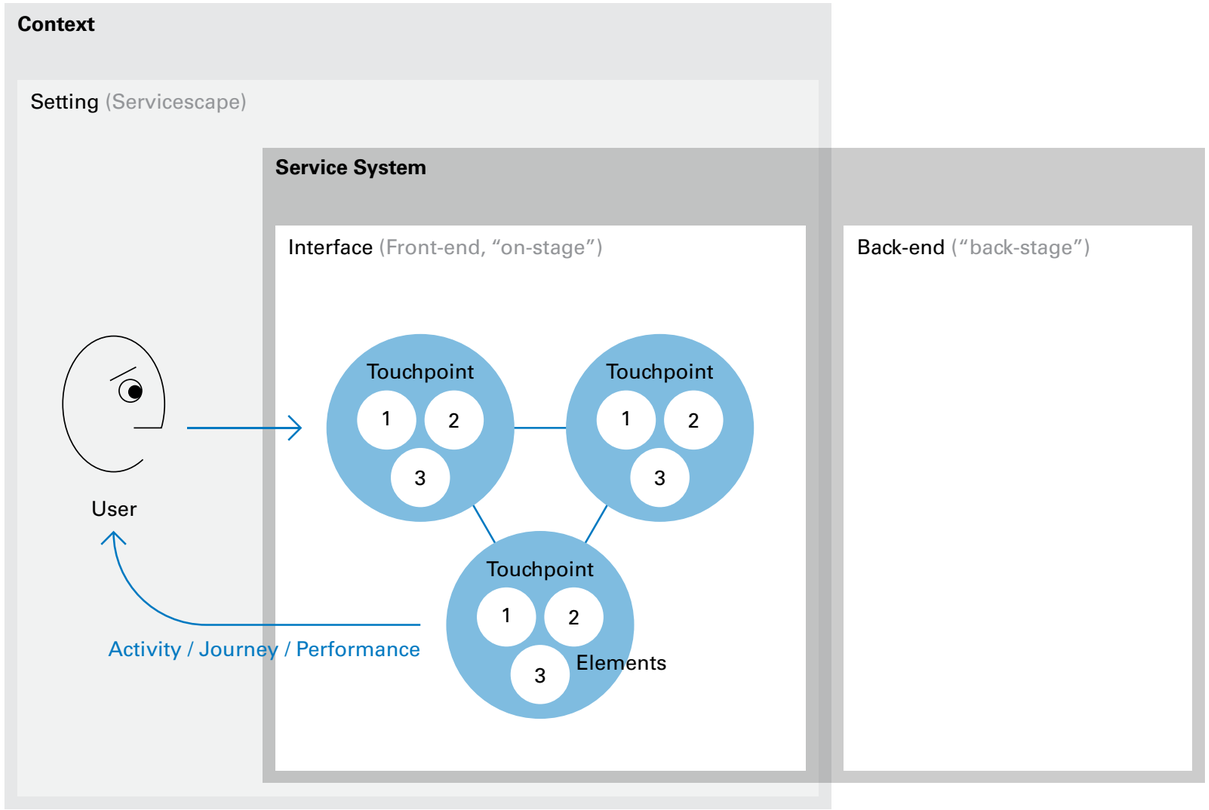
Source: Jodi Forlizzi

Modern services are scaffolded by information systems,

that
- recognize users and personalize responses
- collect, store, and analyze data
- look for patterns and predict events



Services are processes,
coordinating the actions of people and products.
They are co-created by producers and consumers at the point of delivery.

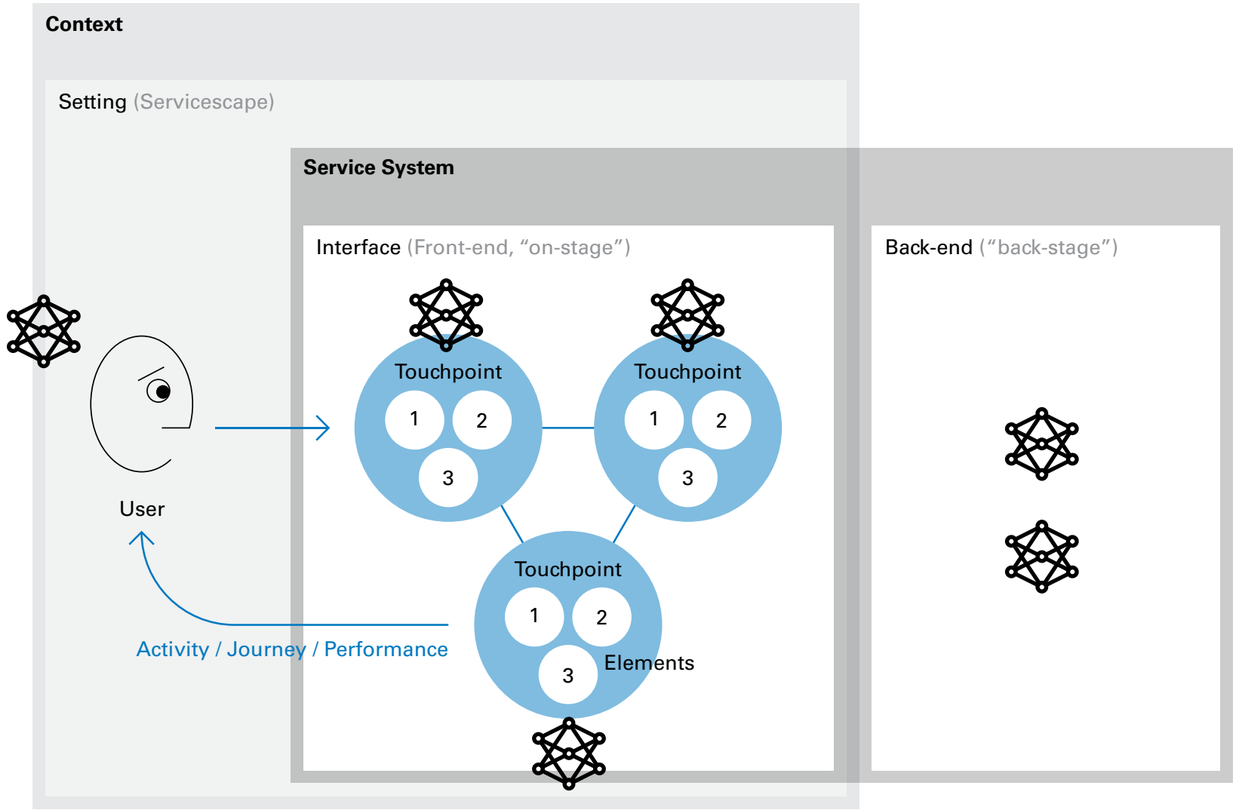


Based on Gupta, Vajic—L. Suchman, and J. Lave

Every service will offer software agents

to assist consumers and providers —
i.e., agent design is becoming core to service design.

Users will show up with their own agents as well.



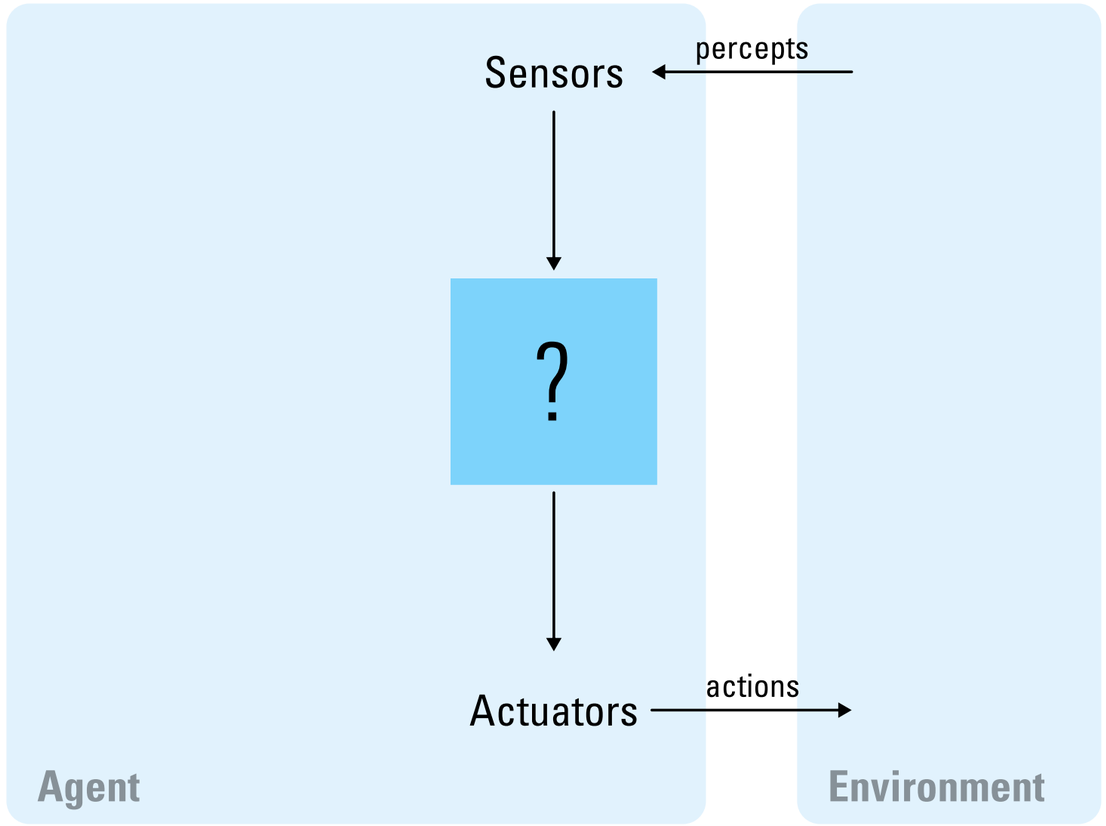
PART TWO

Defining agents

What is an “agent”? (also called an “intelligent agent”)

“An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.”

— Stuart Russell and Peter Norvig, *Artificial Intelligence, A Modern Approach,* 2010.



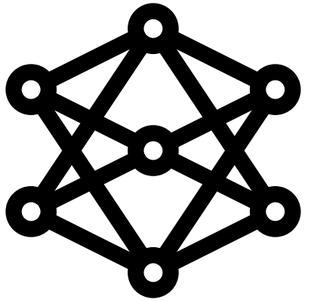
Implied is that agents have rules for choosing actions;
information, feedback, goals, teleology, cybernetics, learning, and even play are implicated.

Russell + Norvig's definition of agents includes:

- Humans
- Other living things
- Thermostats (control systems)
- Autonomous robots
- Software agents

1. the level of **Frameworks**
2. the level of **Clockworks**
3. the level of **Thermostats**
4. the level of the **Cell**
5. the **Genetic** and **Societal** level
6. the level of the **Animal**
7. the **Human** level
8. the level of the **Social Organism**
9. the level of **Transcendental Systems**

Kenneth Boulding's
Nine types of systems
General Systems Theory: The Skeleton of Science (1956)



The capabilities of software agents will build in a stack of layers.

1. Sensors and external data sources enable the comprehensive monitoring of:
   - the product's condition
   - the external environment
   - the product's operation and usage

   Monitoring also enables alerts and notifications of changes.

2. Software embedded in the product or in the product cloud enables:
   - Control of product functions
   - Personalization of the user experience

3. Monitoring and control capabilities enable algorithms that optimize product operation and use in order to:
   - Enhance product performance
   - Enable predictive diagnostics, service, and repair

4. Combining monitoring, control, and optimization enables:
   - Autonomous product operation
   - Self-coordination of operation with other products and systems
   - Autonomous product enhancement and personalization
   - Self-diagnosis and service

Michael Porter, *How Smart, Connected Products Are Transforming Competition,* Harvard Business Review, 2014.

Russell + Norvig categorized agents into five classes,
providing frameworks for software agent design:

1. **Simple reflex agents**
   select actions based on comparing a rule to the current input (only)
2. **Model-based reflex agents**
   also build a history of inputs and model the state of the “world”
3. **Goal-based agents**
   also track a set of goals they’re trying to achieve
4. **Utility-based agents**
   also include a utility function and choose actions to maximize it
5. **Learning agents**
   add a second layer that can experiment

For more on each type of agent see:
“Intelligent Agents, Multi-Agent Systems, & Cybernetics,” Dubberly, 2024
https://presentations.dubberly.com/Intelligent_Agents_and_Multi-Agent_Systems.pdf

LLM-based chatbots are a simple form of agent.

(e.g., in the US: ChatGPT, Claude, Cursor, and in China: ChatGLM, Ling-1T)

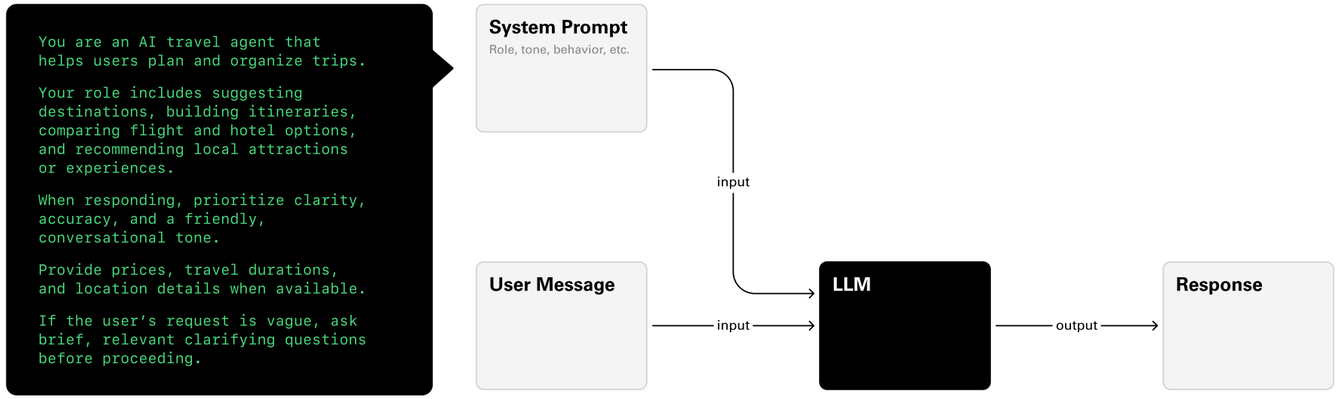
They take an input and transform* it to return an output.

*Much has been written describing how LLMs and transformers work. While interesting, that's not the topic of this talk.

See Stephen Wolfram's "What is ChatGPT Doing... and Why Does It Work?" https://writings.stephenwolfram.com/2023/02/what-is-chatgpt-doing-and-why-does-it-work/ Our summary is here: https://presentations.dubberly.com/text-to-text_AI.pdf

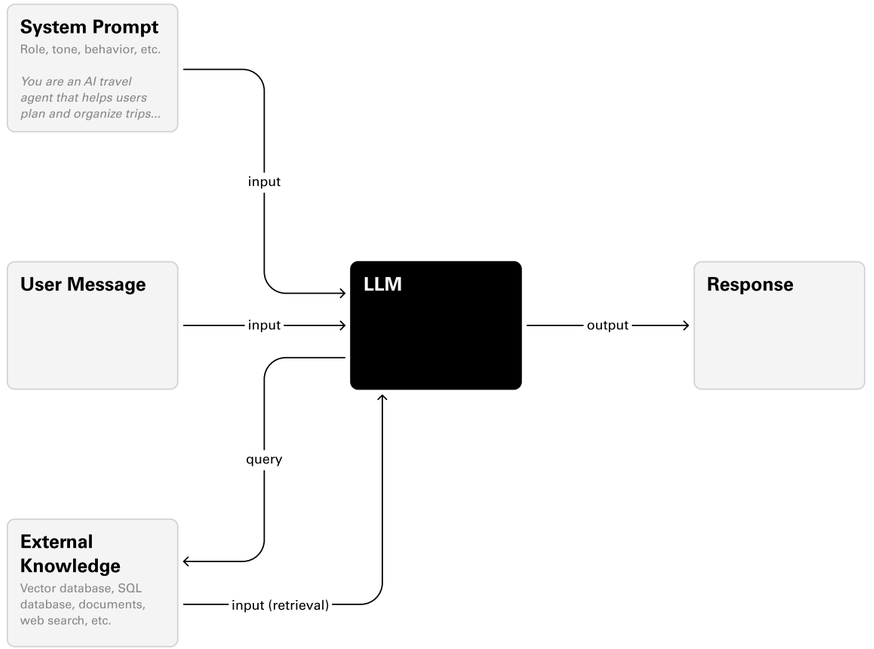
Agents often have a second input —

a system prompt, describing the agent's role, tone, and constraints.



Agents may draw on information sources outside the LLM through a process called Retrieval Augmented Generation (RAG).

Outside sources might include:
- unstructured text files
- structured databases
- embeddings and vector databases
- data directly from local sensors
- predictions from digital twins
- the world-wide web
- other cloud-based services



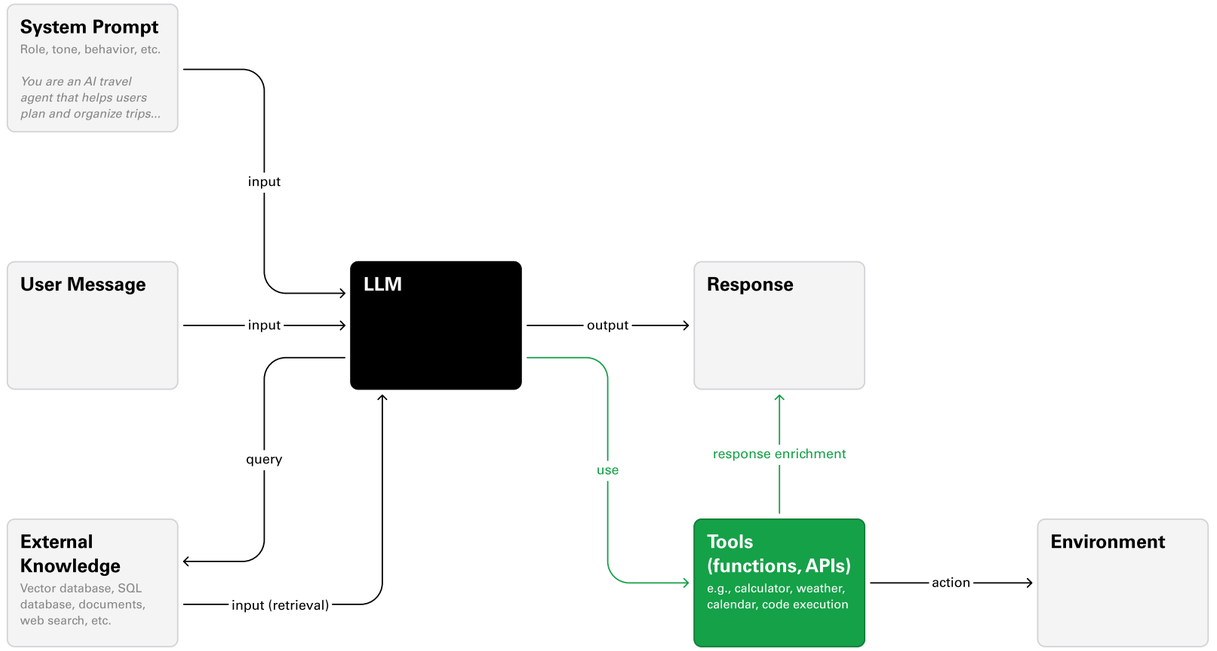
For more on digital twins, please see:
https://presentations.dubberly.com/Digital_Twins.pdf

Agents may also extend the LLM's capabilities beyond text prediction by adding various tools.

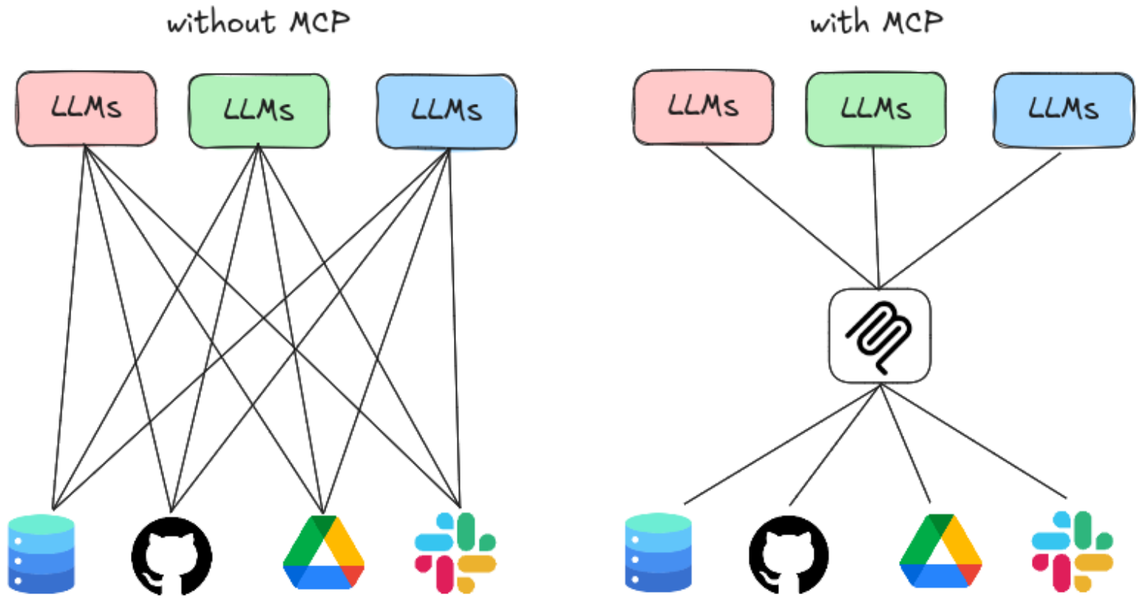
Tools might call APIs, run scripts, or execute larger code blocks.
(e.g., retrieving files from a cloud drive, fetching email, or weather data.)

Tools can be used to enrich the LLM's output response in the chat.
(e.g., using a calculator, embedding a calendar or weather widget).

Tools also enable LLMs to perform actions on the external environment.
(e.g., booking a flight, hotel, and car, and adding the events to your calendar.)



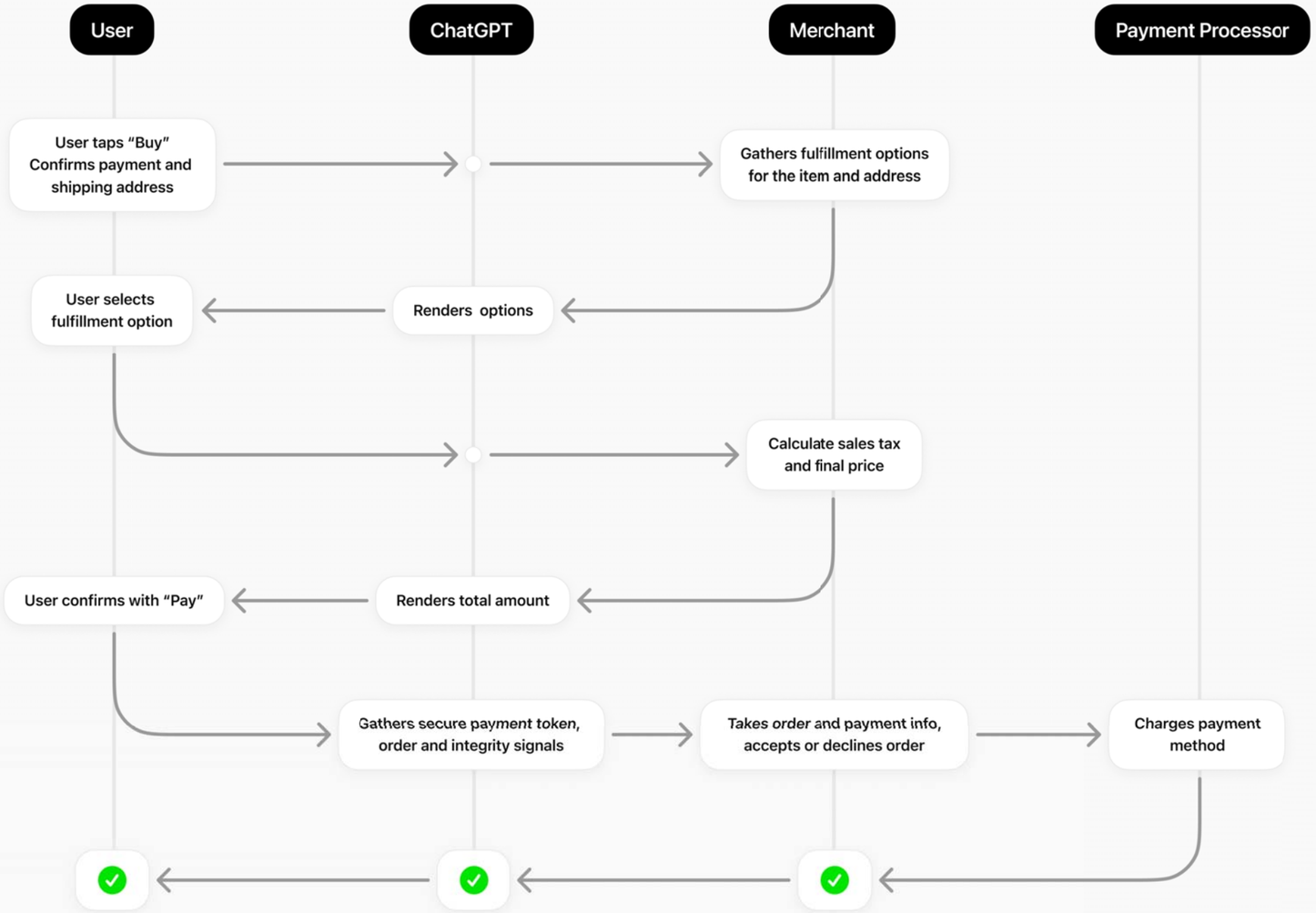
Agents may also connect to MCP servers (Model Context Protocol), offering APIs (Application Programming Interfaces) — for connecting to services, requesting and uploading data, talking with other agents, etc.



For example, Figma offers an MCP server —
enter an API key in Cursor, and it reads a selected frame in a Figma file,
so that the AI agent can collect structured information.

Source: https://hoangndst.com/blog/model-context-protocol

Recently, OpenAI introduced ACP (Agentic Commerce Protocol), which helps agents complete purchases.



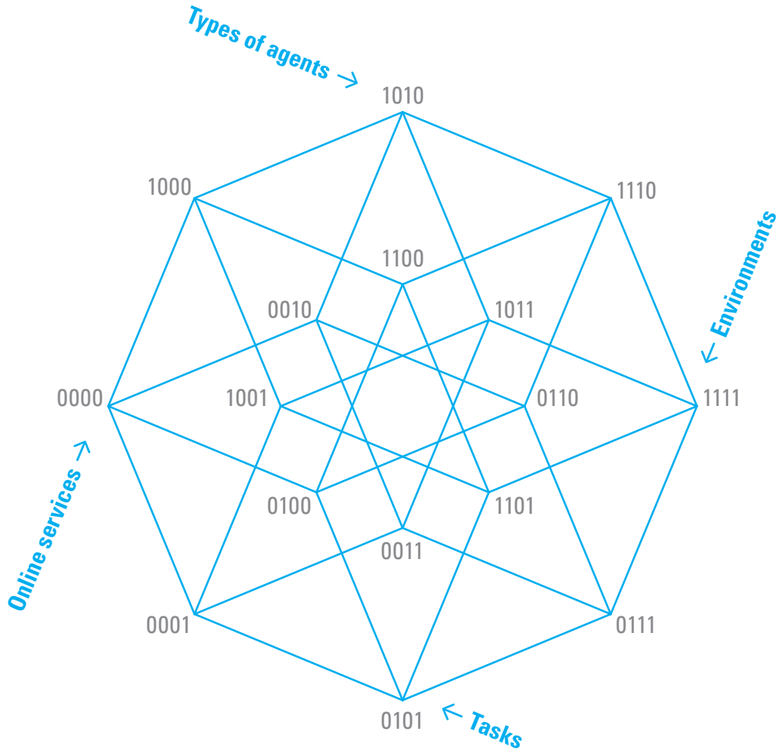
Source: OpenAI

The opportunity space of possible agents is huge and growing.

Consider some of the dimensions:

- Russell + Norvig's taxonomy of **types**
- Possible **environments** or contexts-of-use
- Possible **tasks** or domains-of-action
- Available **on-line services**

and their combinations + permutations.



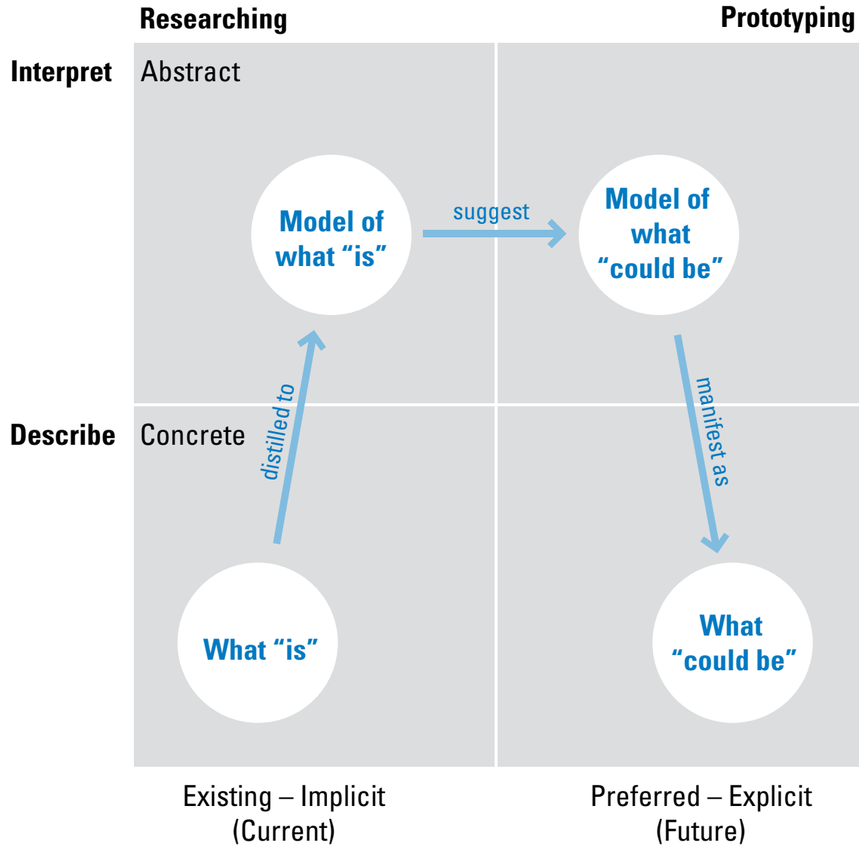
Agent design process

Designing agents follows a familiar process:

- Understand users, their needs, and context
- Model the current situation
- Model the preferred situation
- Build a prototype,

test

and iterate



For more on the bridge model, see:
https://www.dubberly.com/wp-content/uploads/2016/02/ddo_interactions_bridgemodel.pdf

For more on the design process, see:
https://www.dubberly.com/wp-content/uploads/2008/06/ddo_designprocess.pdf

A key aspect of designing agents is building systems models.

- What are people **doing today?**
  Observe users and collect example artifacts

- What **inputs** are available?
  Find expert content and useful data — clean it

- What **transformations** are possible?
  Define the rules for making decisions

- What **outputs** are desired?
  Write stories, prompt sets, and evals

Agents can be built using ADEs (Agent Development Environments),

such as

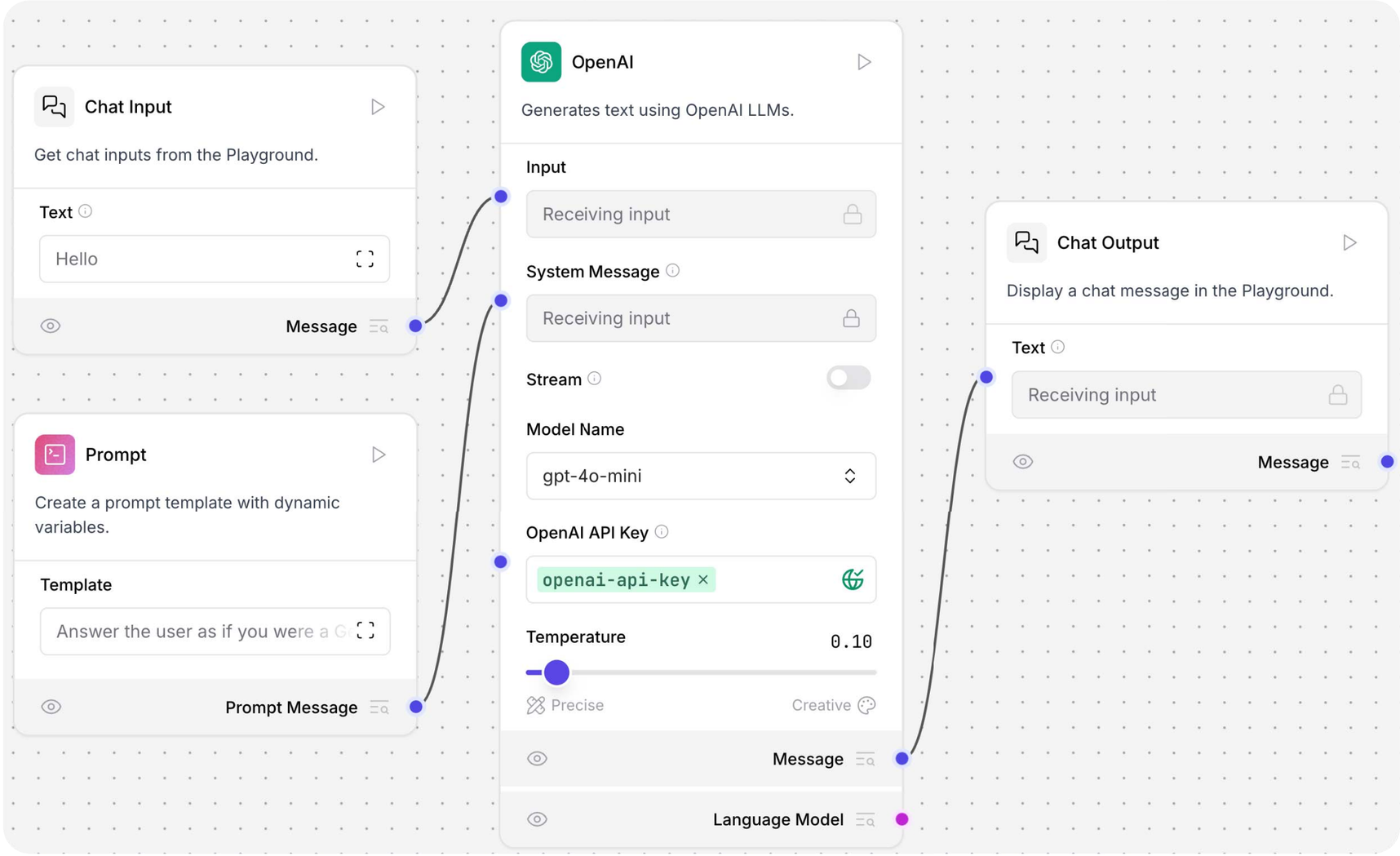
- AutoGen
- CrewAI
- AgentKit

```
import { Agent, run } from '@openai/agents';

const faqAgent = new Agent({
  name: 'FAQ',
  instructions: 'Answer common laptop support questions, like how to reset the laptop.',
});

const triageAgent = Agent.create({
  name: 'Triage',
  instructions: 'For any laptop support question, route to the FAQ agent.',
  handoffs: [faqAgent],
});

const result = await run(triageAgent, 'How do I reset my laptop?');
console.log(result.finalOutput);

```

Graphical ADEs make the agent's structure visible.

For example,

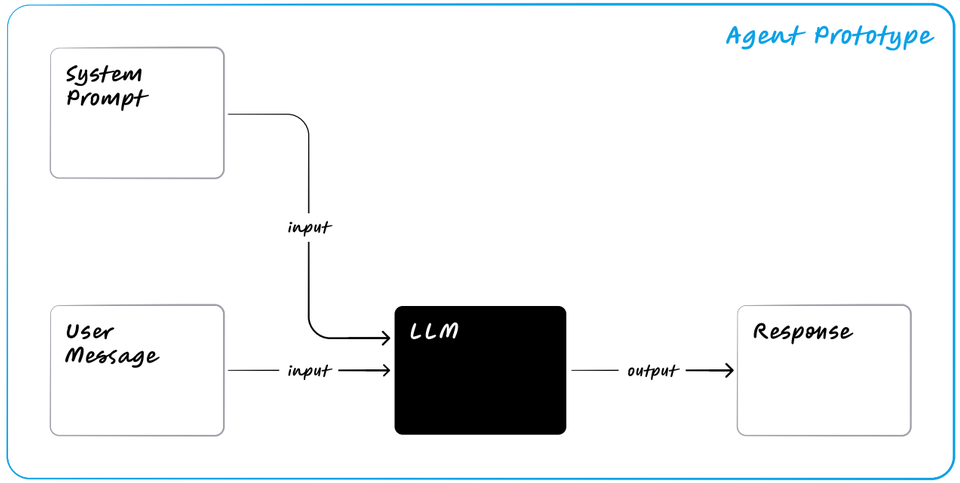
- LangFlow
- n8n
- Zerowidth



For the designers of agents,

diagramming the system can be creating a prototype —

i.e., boxes and arrows become a working simulation.



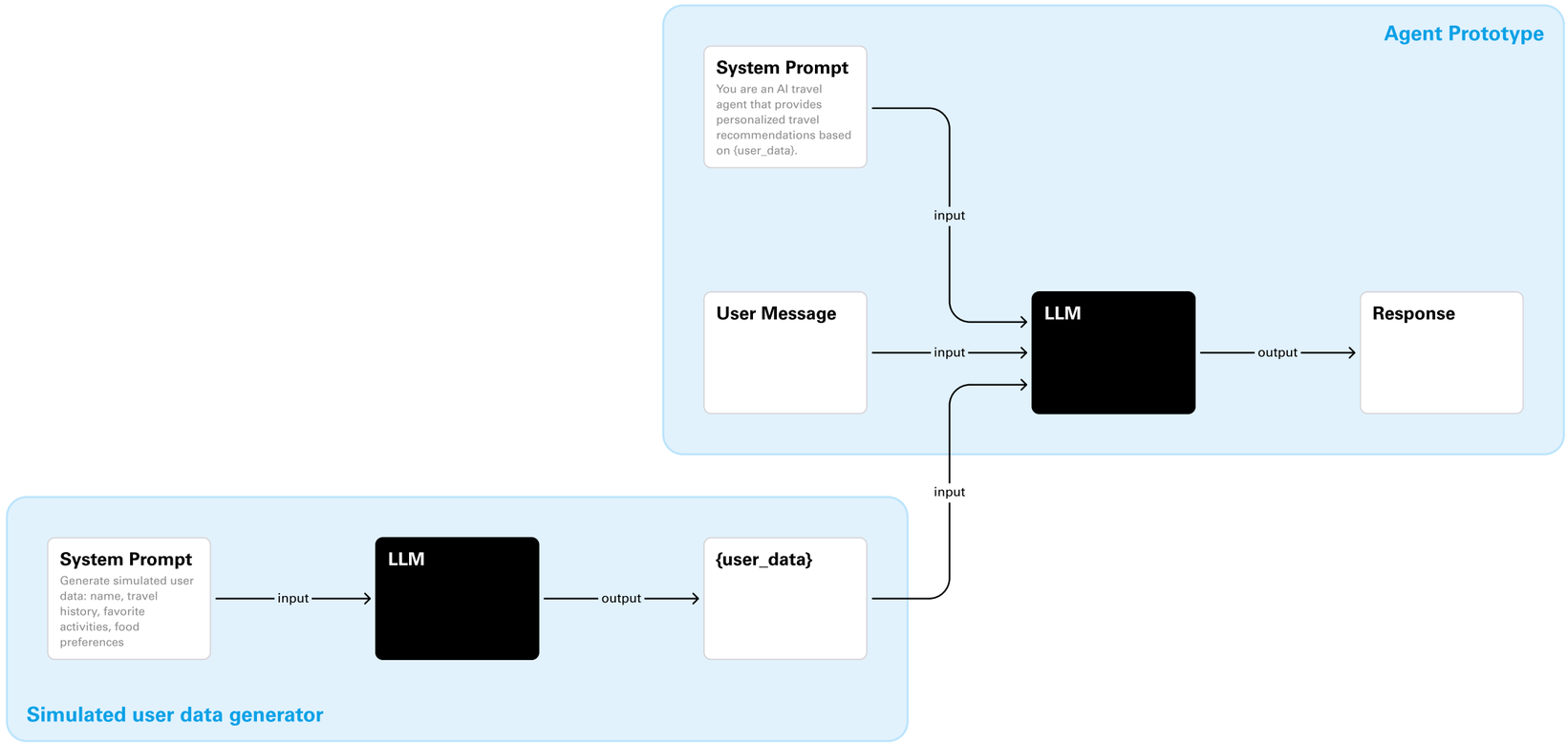
Diagramming used to be an abstraction removed from the actual product.

Now, diagramming is much closer to the real thing.

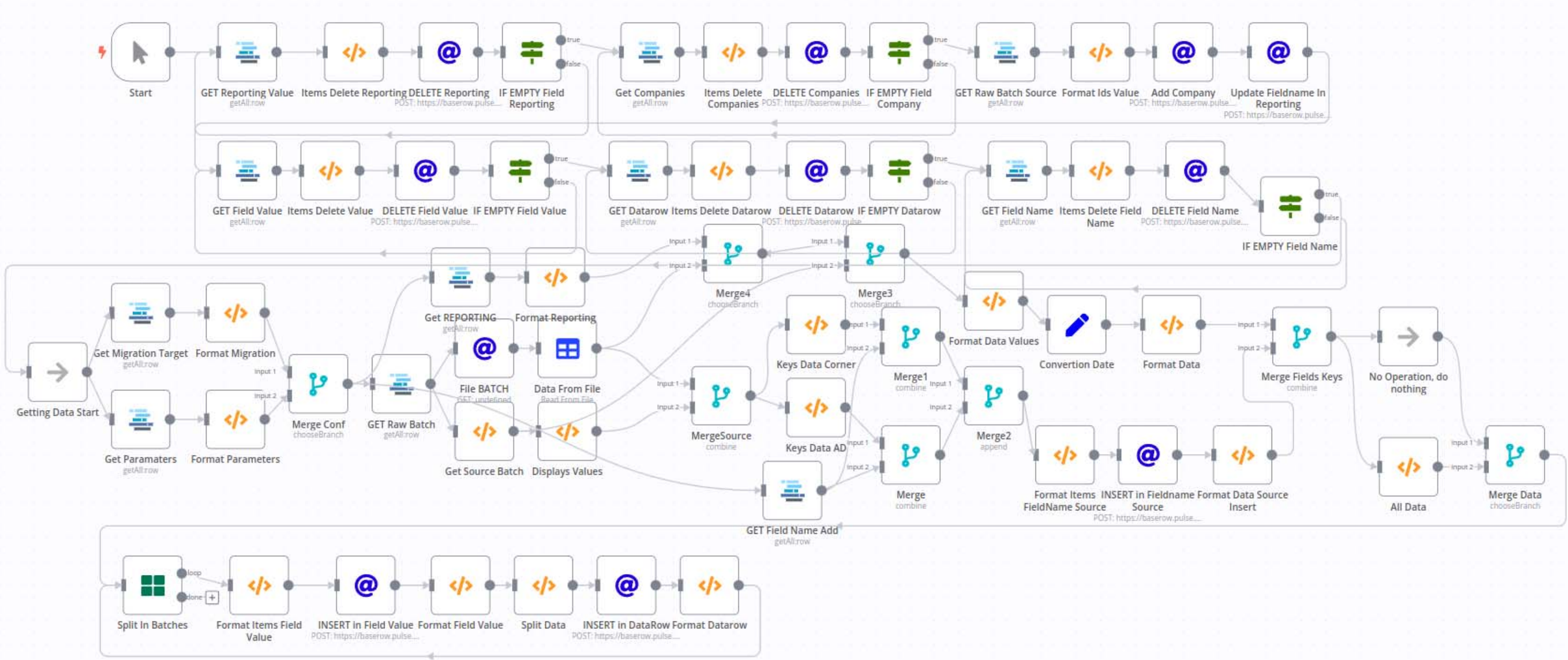
This is a major paradigm shift in design.

Additional agents may generate fake data for prototype agents.

Building a test environment for prototype agents enables faster iteration.



Over time, agents will become more sophisticated — and systems or networks of agents will emerge.



Mark Weiser's **Calm-Tech Design Principles** are relevant:

- Technology should require the **smallest** possible **amount of attention**
- Technology should inform and **create calm**
- Technology should **make use of the periphery**
- Technology should amplify the **best of technology** and the **best of humanity**
- Technology can communicate, but **doesn't need to speak**
- Technology **should work even when it fails**
- The right amount of technology is the **minimum needed to solve the problem**
- Technology should **respect social norms**

Mark Weiser has been called the father of ubiquitous computing.
He was CTO at Xerox PARC.

Source: https://calmtech.com/papers/designing-calm-technology

Examples of agents

Agents are currently in an awkward adolescence,
not yet having reached their potential.

Already, we see agents that...

- **summarize** patient notes
- **check** forms for compliance to regulations
- **answer** questions about your HVAC system
- **debug** your insulin pump
- **form** project teams at work
- **alert** you to research of interest
- **coach** you on preparing for your next 10k race

When they grow up, agents will be everywhere — billions of them.

Every physical process and virtual process (that matters)
will be overseen by a monitoring agent (a.k.a. Unix daemons),
alerting supervisor agents when errors occur or are predicted.

The supervisor agents will ladder up in a management tree
to an orchestrator agent at a human touch-point
(though humans may drill all the way down if they like).

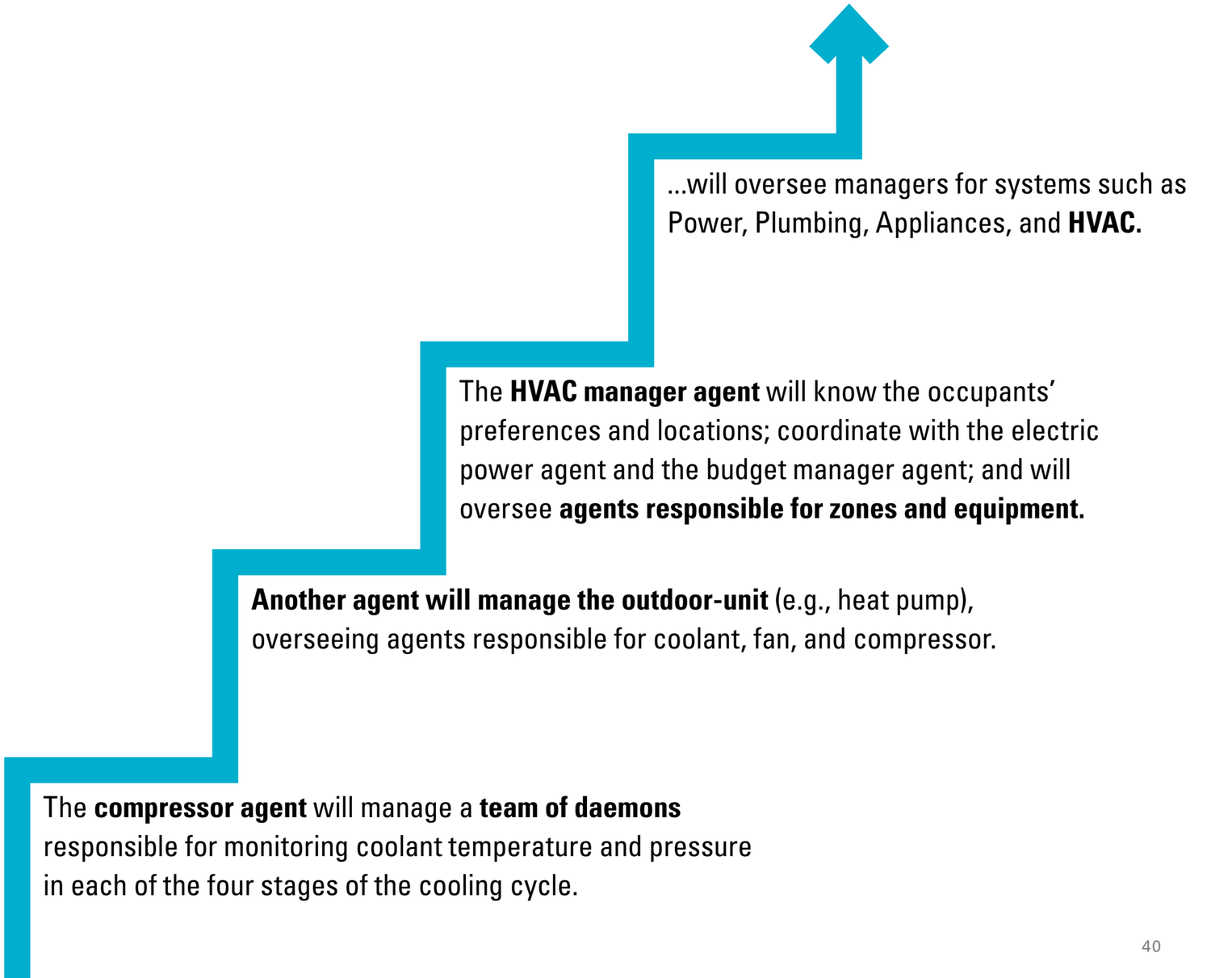
At the scale of the individual,
agents will help with many aspects of our lives.
Imagine a staff of personal assistants.

At the scale of the home,
management systems will include dozens of agents, similar to the staff of a 19th-century country estate.

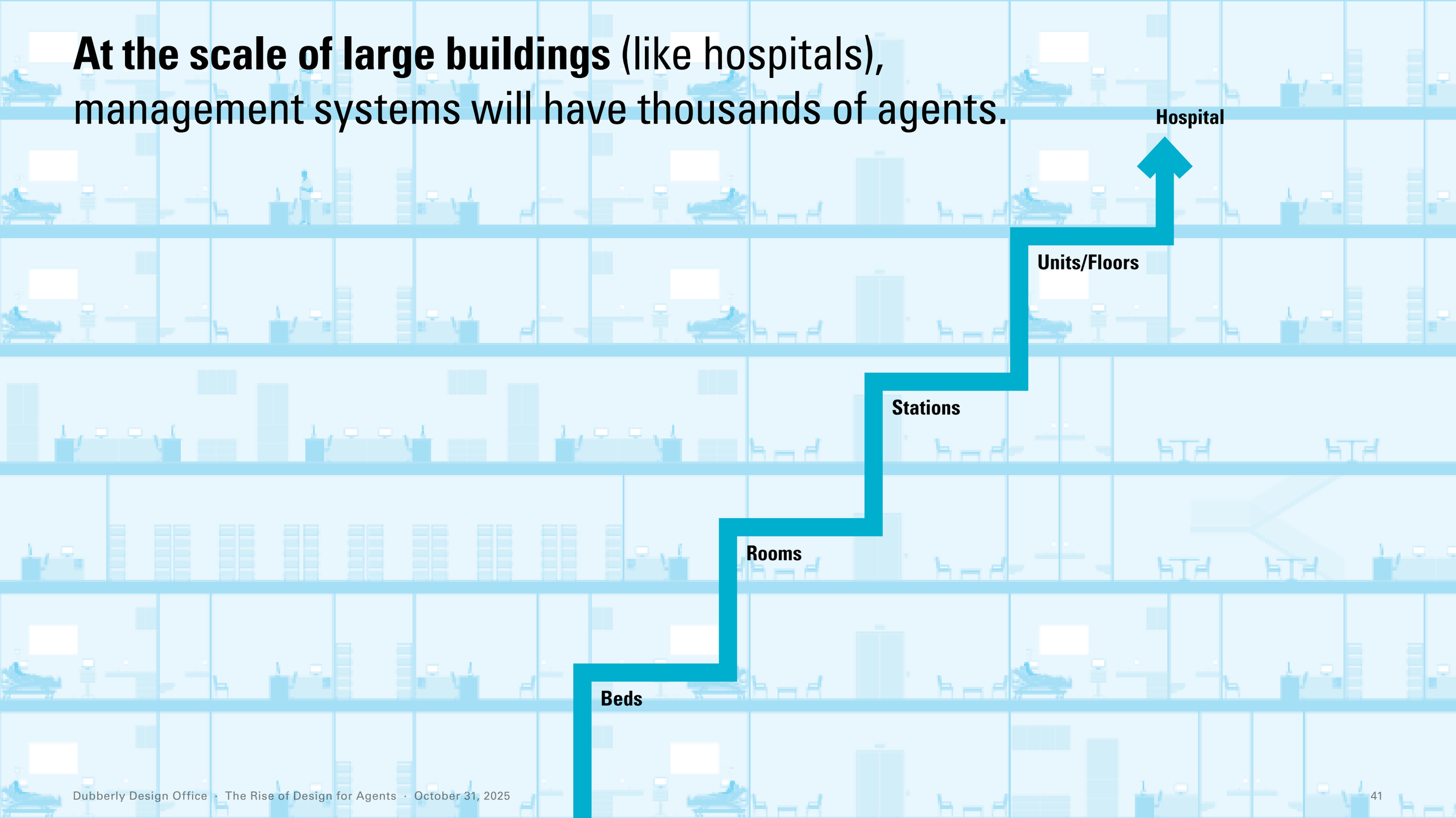
Xerox PARC researcher Rich Gold (1995) offered a dystopian counterview of “intelligent houses”:
“How smart does your bed have to be, before you are afraid to go to sleep at night?”
https://webarchive.ars.electronica.art/en/archives/festival_archive/festival_catalogs/festival_artikel.asp%3FiProjectID=8689.html

At the scale of appliances,
manager-agents will oversee teams of worker-agents.

Each device in each system of each household might have a similar hierarchy of agents; and agents in different branches might communicate as well.



At the scale of large buildings (like hospitals), management systems will have thousands of agents.



At the scale of organizations,
there will also be thousands of agents.

- **Translators**
  Move from natural language to programming code

- **Librarians**
  Locate relevant information and even monitor change

- **Analysts**
  Determine probable causes for change

- **Clerks**
  Prepare reports and recommend actions

- **Lookouts**
  Check a value against a threshold and alerts when reached

- **Assistants**
  Perform routine tasks, e.g., running through check-lists

- **Coaches**
  Suggest frameworks, manage training plans, answer questions

- **Orchestrators**
  Coordinate the work of other agents

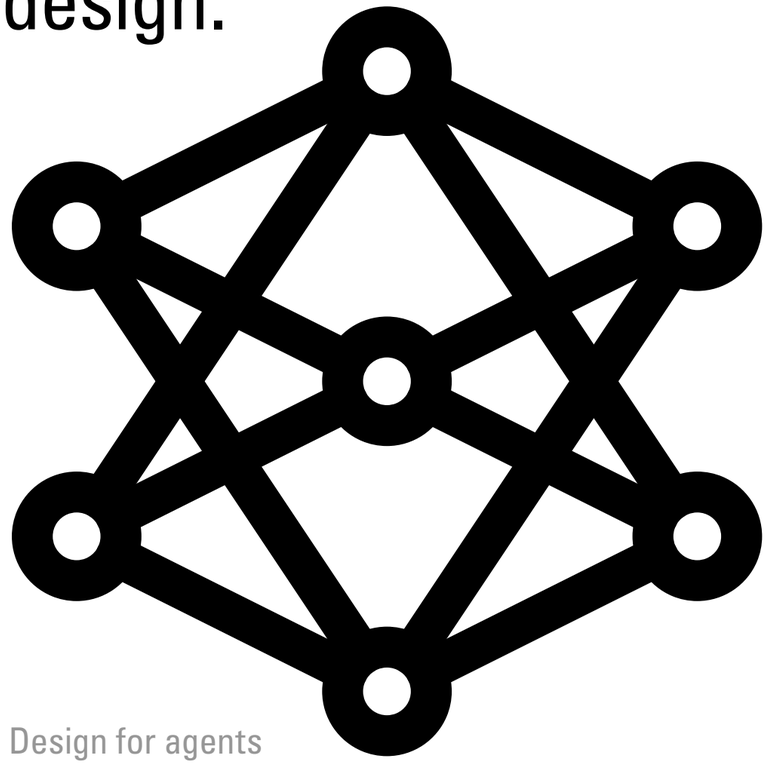
All these agents will need to be designed —
along with the **touchpoints** through which they interact,
the processes that constitute our **services,**
and the product-service **ecologies** that form our world.

This space of opportunities is part of the future of design.

Hand-craft

Graphic design

Interaction design



Design for agents

Special thanks to
Jing Su
Jack Hu
Barry Katz
Jenny Quan
Gavin Miller
Jamie Ikeda
Ryan Reposar

Presentation posted at
presentations.dubberly.com/Agent_Design.pdf

Email:
hugh@dubberly.com