에이전트형 운영체제의 인터페이스에 대한 재해석: 사용자 인터페이스 이론을 중심으로

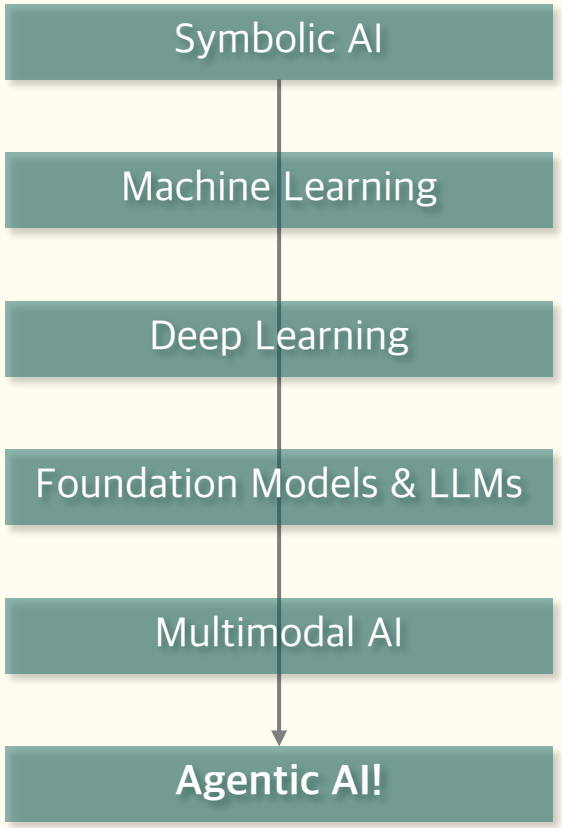
Reinterpreting Interfaces in Agentic Operating Systems: A User Interface Theory Perspective

Jin Myung Lee | B.S. ECE @ SNU ↗

0. 목차

1. 서론 / 에이전트형 인공지능

History of AI



What is Agentic AI

Traditional LLM	Agentic AI
Prompt to Response	Prompt to Action
단일 응답 생성	다단계 작업 수행

"Agentic AI는 제한적인 인간의 감독 아래에서 추론, 계획, 행동을 통하여 자율적으로 목표를 달성할 수 있는 인공지능 시스템이다"

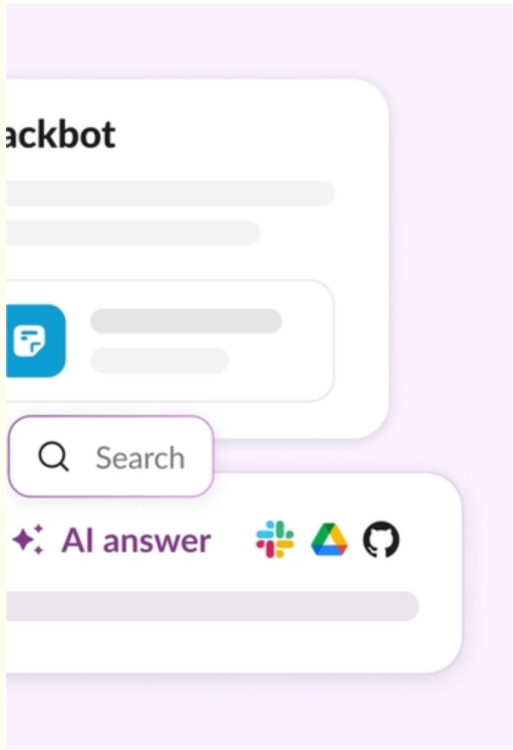
—IBM, *What is Agentic AI?*

1. 서론 / 에이전트형 운영체제

What is Agentic OS

“에이전트형 운영체제는 자율 에이전트를 조율하고 데이터 및 애플리케이션과 연결하여, 인간의 감독 아래 목표 달성을 위한 계획·추론·행동을 수행할 수 있도록 지원하는 계층이다”

—slack, *What Is an Agentic OS? A Practical Guide*



slack from Salesforce

GET STARTED

PRODUCTIVITY

What Is an Agentic OS? A Practical Guide

More than other AI tools, an agentic operating system provides the infrastructure, context, and safeguards for AI agents to work alongside humans.

By the team at Slack
April 3rd, 2026

TABLE OF CONTENTS

8 min read

Key takeaways

- Agentic OS systems include orchestration layers, security guardrails, and human-in-the-loop controls that let teams safely integrate autonomous agents into everyday work.
- AI agents can plan, reason, and act – moving beyond sim

Ask Agentforce

1. 서론 / 에이전트형 운영체제

What is Agentic OS

“Windows 365 for Agents는 에이전트가 다양한 소프트웨어를 넘나들며 다단계 워크플로우를 수행할 수 있는…”

—Microsoft, *Made for developers and agents, Windows 365 at Build 2026*

“Computer-using agents in Microsoft Copilot Studio are now generally available”

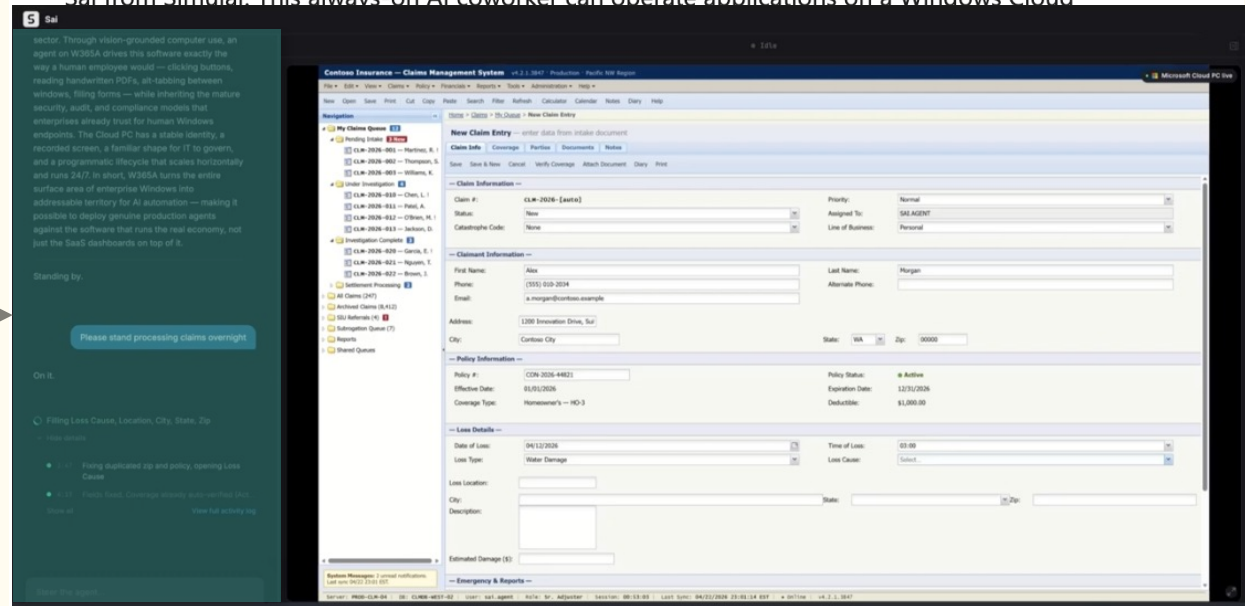
—Microsoft, *Copilot Studio Blog*

에이전트형 운영체제의 인터페이스

Register Sign In

Windows 365 for Agents is now generally available. Agent makers can use it as part of Agent 365 tools or through Microsoft Copilot Studio (preview). It enables enterprise AI automation by providing agents with secured, managed, and available Cloud PCs that run within real business environments. Agents can interact directly with applications and browsers, execute multi-step workflows, and operate across modern and legacy systems. Each Cloud PC is Entra-joined, Intune-managed, and policy-enforced, helping IT scale agents with consistent security, governance, and compliance. While Agent 365 secures and governs the agent, Windows 365 for Agents provides a dedicated workspace to support performance and security needs.

Designed to support agent creators, Windows 365 for Agents works with agents built using both no-code and pro-code approaches. It's already powering agentic experiences across Microsoft, from computer-use scenarios in Researcher to Project Opal within Microsoft Copilot Studio, demonstrating enterprise readiness. Beyond Microsoft's own experiences, the platform also supports third-party agents, including partner-provided examples such as Sai from Simular. This always-on AI coworker can operate applications on a Windows Cloud



Selecting “burst pipe” and other items from the dropdown menu, navigating the UI efficiently.

1. 서론 / 운영체제의 인터페이스

명령줄 인터페이스 CLI [Command-Line Interface])

```
    Hardware Overview:

      Model Name: MacBook Pro
      Model Identifier: MacBookPro18,1
      Model Number: MK183KH/A
      Chip: Apple M1 Pro
      Total Number of Cores: 10 (8 Performance and 2 Efficiency)
      Memory: 16 GB
      System Firmware Version: 18000.120.36
      OS Loader Version: 18000.120.36
      Serial Number (system): XDXRWC3FPG
      Hardware UUID: 3054CBB0-A723-55C3-9CFC-4789EFD78960
      Provisioning UDID: 00006000-000610A2117A401E
      Activation Lock Status: Enabled

[jinmyounglee@Xlaudes-MacBook-Pro ~ % ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
        options=1203<RXCSUM,TXCSUM,TXSTATUS,SW_TIMESTAMP>
        inet 127.0.0.1 netmask 0xff000000
        inet6 ::1 prefixlen 128
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
        nd6 options=201<PERFORMNUD,DAD>
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
stf0: flags=0<> mtu 1280
anpi1: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=400<CHANNEL_IO>
        ether 4e:e9:1c:01:93:6d
        media: none
        status: inactive
anpi0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=400<CHANNEL_IO>
        ether 4e:e9:1c:01:93:6c
        media: none
        status: inactive
anpi2: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=400<CHANNEL_IO>
        ether 4e:e9:1c:01:93:6e
        media: none
        status: inactive
en4: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=400<CHANNEL_IO>
        ether 4e:e9:1c:01:93:4c
        nd6 options=201<PERFORMNUD,DAD>
        media: none
        status: inactive
```

그래픽 사용자 인터페이스 GUI [Graphic User Interface]

1. 서론

” 에이전트형 운영체제는 CUI와 GUI가 공유하는 어떠한 **기반**과 거리가 멀다.

이를 규명하고, 에이전트형 운영체제와 다른 점을 사용자 인터페이스 이론으로 논하자.”

“이를 통해 빅테크 기업들의 **운영체제**라는 명명이 적절한지 통찰을 얻을 수 있다.”

2. 이론적 배경 / 사용자 인터페이스 이론

Direct manipulation systems offer the satisfying experience of operating on visible objects. The computer becomes transparent, and users can concentrate on their tasks.

Direct Manipulation: A Step Beyond Programming Languages

Ben Shneiderman, University of Maryland

Leibniz sought to make the form of a symbol reflect its content. "In signs," he wrote, "one sees an advantage for discovery that is greatest when they express the exact nature of a thing briefly and, as it were, picture it; then, indeed, the labor of thought is wonderfully diminished."

Frederick Kreiling, "Leibniz," *Scientific American*, May 1968

Certain interactive systems generate glowing enthusiasm among users—in marked contrast with the more common reaction of grudging acceptance or outright hostility. The enthusiastic users' reports are filled with positive feelings regarding

- mastery of the system,
- competence in the performance of their task,
- ease in learning the system originally and in assimilating advanced features,
- confidence in their capacity to retain mastery over time,
- enjoyment in using the system,
- eagerness to show it off to novices, and
- desire to explore more powerful aspects of the system.

These feelings are not, of course, universal, but the amalgam does convey an image of the truly pleased user. As I talked with these enthusiasts and examined the systems they used, I began to develop a model of the features that produced such delight. The central ideas seemed to be visibility of the object of interest; rapid, reversible, incremental actions; and replacement of complex command language syntax by direct manipulation of the object of interest—hence the term "direct manipulation."

Examples of direct manipulation systems

No single system has all the attributes or design features that I admire—that may be impossible—but those described below have enough to win the enthusiastic support of many users.

Display editors. "Once you've used a display editor, you'll never want to go back to a line editor. You'll be spoiled." This reaction is typical of those who use full-page display editors, who are great advocates of their systems over line-oriented text editors. I heard similar comments from users of stand-alone word processors such as the Wang system and from users of display editors such as EMACS on the MIT/Honeywell Multics system or "vi" (for visual editor) on the Unix system. A beaming advocate called EMACS "the one true editor."

Roberts[1] found that the overall performance time of display editors is only half that of line-oriented editors, and since display editors also reduce training time, the evidence supports the enthusiasm of display editor devotees. Furthermore, office automation evaluations consistently favor full-page display editors for secretarial and executive use.

The advantages of display editors include

Display of a full 24 to 66 lines of text. This full display enables viewing each sentence in context and simplifies reading and scanning the document. By contrast, the

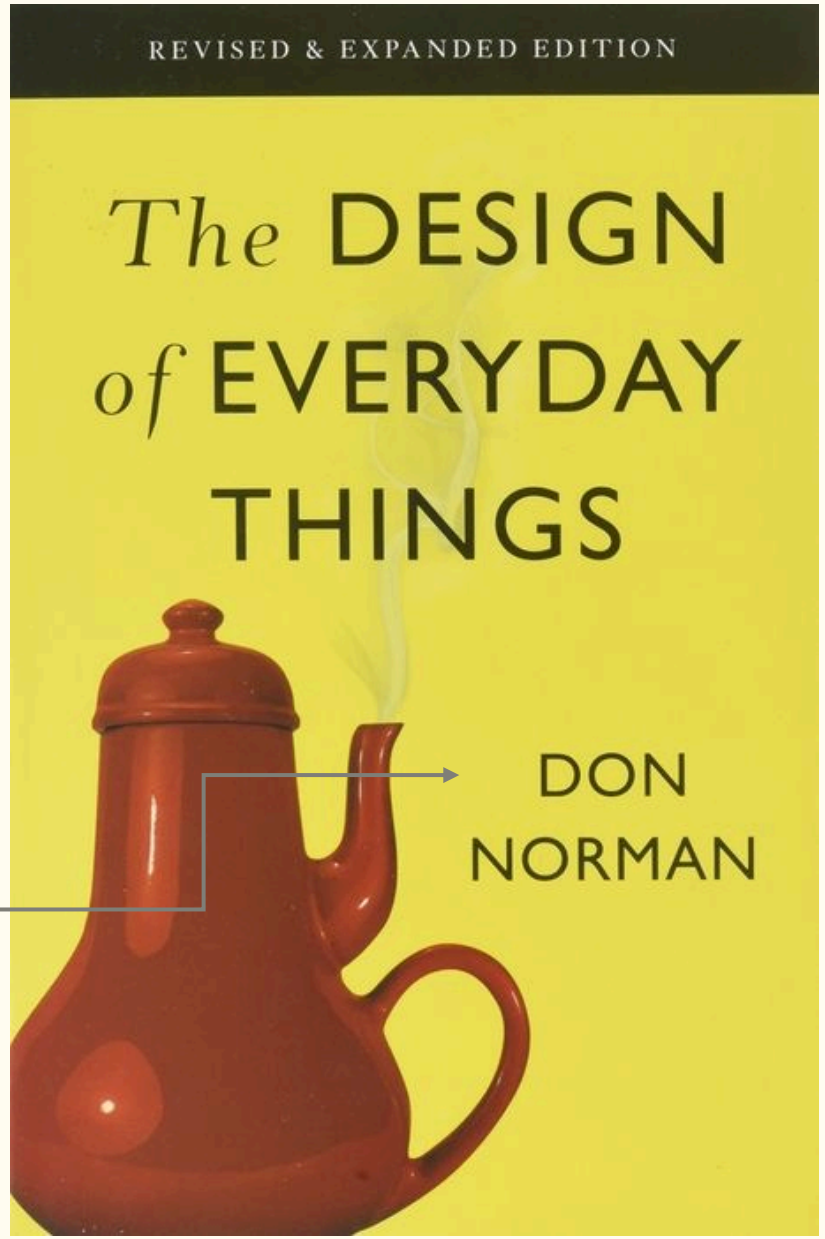
A portion of this article was derived from the author's keynote address at the NYU Symposium on User Interfaces, "The Future of Interactive Systems and the Emergence of Direct Manipulation," published in *Human Factors in Interactive Computer Systems*, Y. Vassiliou, ed., Ablex Publishing Co., Norwood, N.J., 1983.

August 1983 0018-9162/83/0800-0057$01.00 1983 IEEE

벤 슈나이더맨—직접 조작 이론

메릴랜드 대학교 컴퓨터과학과 명예 석좌 교수

HCI·정보 시각화 분야를 개척한 세계적인 연구자



도널드 노먼—인지과학적 접근

현대 디자인과 기술 산업에서

사용자 중심 디자인과 사용자 경험을 정립한

인지과학자 및 디자인 이론가

2. 이론적 배경 / 사용자 인터페이스 이론

벤 슈나이더맨

학습 용이성 & 수행 효율 & 통제·예측 가능성을 위한 4가지 조건

1. 관심 대상이 화면 위에 가시적으로 나타나야 함

2. 사용자의 행위가 결과로 바로 이어져야 함

3. 행위는 작은 단위로 쪼개어 점진적으로 수행할 수 있어야 함

4. 잘못된 결과는 되돌릴 수 있어야 함

tion. (Deeper understanding of visual perception can be obtained from Arnheim[17] and McKim.[18])

Physical, spatial, or visual representations are also easier to retain and manipulate. Wertheimer[19] found that subjects who memorized the formula for the area of a parallelogram, $A = h \times b$, mastered such calculations rapidly. On the other hand, subjects who were given a structural explanation (cut a triangle from one end and place it on the other) retained the knowledge and applied it in similar circumstances more effectively. In plane geometry theorem proving, a spatial representation facilitates discovery of proof procedures more than an axiomatic representation. The diagram provides heuristics that are difficult to extract from the axioms. Similarly, students of algebra are often encouraged to draw a picture to represent a word problem.

Papert's Logo language[20] creates a mathematical microworld in which the principles of geometry are visible. Influenced by the Swiss psychologist Jean Piaget's theory of child development, Logo offers students the opportunity to create line drawings with an electronic turtle displayed on a screen. In this environment, users can receive rapid feedback about their programs, can easily determine what has happened, can quickly spot and repair errors, and can experience creative satisfaction.

Problems with direct manipulation. Some professional programming tasks can be aided by the use of graphic representations such as high-level flowcharts, record structures, or database schema diagrams, but additional effort may be required to absorb the rules of the representation. Graphic representations can be especially helpful when there are multiple relationships among objects and when the representation is more compact than the detailed object. In these cases, selectively screening out detail and presenting a suitable abstraction can facilitate performance.

However, using spatial or graphic representations of the problem does not necessarily improve performance. In a series of studies, subjects given a detailed flowchart did no better in comprehension, debugging, or modification than those given the code only.[21] In a program comprehension task, subjects given a graphic representation of control flow or data structure did no better than those given a textual description.[22] On the other hand, subjects given the data structure documentation consistently did better than subjects given the control flow documentation. This study suggests that the content of graphic representations is a critical determinant of their utility. The wrong information, or a cluttered presentation, can lead to greater confusion.

A second problem is that users must learn the meaning of the components of the graphic representation. A graphic icon, although meaningful to the designer, may require as much—or more—learning time as a word. Some airports serving multilingual communities use graphic icons extensively, but their meaning may not be obvious. Similarly, some computer terminals designed for international use have icons in place of names, but the meaning is not always clear.

A third problem is that the graphic representation may be misleading. The user may rapidly grasp the analogical representation, but then make incorrect conclusions about permissible operations. Designers must be cautious in selecting the displayed representation and the operations. Ample testing must be carried out to refine the representation and minimize negative side effects.

A fourth problem is that graphic representations may take excessive screen display space. For experienced users, a tabular textual display of 50 document names is far more appealing than only 10 document graphic icons with the names abbreviated to fit the icon size. Icons

Choosing the right representations and operations is not easy. Simple metaphors, analogies, or models with a minimal set of concepts seem most appropriate.

should be evaluated first for their power in displaying static information about objects and their relationship, and second for their utility in the dynamic processes of selection, movement, and deletion.

Choosing the right representations and operations is not easy. Simple metaphors, analogies, or models with a minimal set of concepts seem most appropriate. Mixing metaphors from two sources adds complexity, which contributes to confusion. The emotional tone of the metaphor should be inviting rather than distasteful or inappropriate[16]—sewage disposal systems are an inappropriate metaphor for electronic message systems. Since users may not share the designer's metaphor, analogy, or conceptual model, ample testing is required.

The syntactic/semantic model. The attraction of systems that use principles of direct manipulation is confirmed by the enthusiasm of their users. The designers of the examples given had an innovative inspiration and an intuitive grasp of what users wanted. Each example has features that could be criticized, but it seems more productive to construct an integrated portrait of direct manipulation:

- Continuous representation of the object of interest.
- Physical actions (movement and selection by mouse, joystick, touch screen, etc.) or labeled button presses instead of complex syntax.
- Rapid, incremental, reversible operations whose impact on the object of interest is immediately visible.
- Layered or spiral approach to learning that permits usage with minimal knowledge. Novices can learn a modest and useful set of commands, which they can exercise till they become an "expert" at level 1 of the system. After obtaining reinforcing feedback from successful operation, users can gracefully expand their knowledge of features and gain fluency.[23]

By using these four principles, it is possible to design systems that have these beneficial attributes:

- Novices can learn basic functionality quickly, usually through a demonstration by a more experienced user.

2. 이론적 배경 / 사용자 인터페이스 이론

도널드 노먼

인터페이스 경험을 인지과학적 관점에서 접근

사용자가 '결정론적' 시스템을 효과적으로 다루려면,

머릿속에 시스템의 작동 방식에 대한 모형을 구축해야 됨

폴더·파일—개념 모형이며, 사용자는 이러한 모형과 상호작용

can only produce a high-frequency beep. Just as with the lights, the only way to signal different states of the machine is by beeping different patterns. What do all these different patterns mean? How can we possibly learn and remember them? It doesn't help that every different machine uses a different pattern of lights or beeps, sometimes with the same patterns meaning contradictory things for different machines. All the beeps sound alike, so it often isn't even possible to know which machine is talking to us.

Feedback has to be planned. All actions need to be confirmed, but in a manner that is unobtrusive. Feedback must also be prioritized, so that unimportant information is presented in an unobtrusive fashion, but important signals are presented in a way that does capture attention. When there are major emergencies, then even important signals have to be prioritized. When every device is signaling a major emergency, nothing is gained by the resulting cacophony. The continual beeps and alarms of equipment can be dangerous. In many emergencies, workers have to spend valuable time turning off all the alarms because the sounds interfere with the concentration required to solve the problem. Hospital operating rooms, emergency wards. Nuclear power control plants. Airplane cockpits. All can become confusing, irritating, and life-endangering places because of excessive feedback, excessive alarms, and incompatible message coding. Feedback is essential, but it has to be done correctly. Appropriately.

CONCEPTUAL MODELS

A conceptual model is an explanation, usually highly simplified, of how something works. It doesn't have to be complete or even accurate as long as it is useful. The files, folders, and icons you see displayed on a computer screen help people create the conceptual model of documents and folders inside the computer, or of apps or applications residing on the screen, waiting to be summoned. In fact, there are no folders inside the computer—those are effective conceptualizations designed to make them easier to use. Sometimes these depictions can add to the confusion, however. When reading e-mail or visiting a website, the material appears to be on

3. 에이전트형 운영체제의 분석

벤 슈나이더맨 관점

1. 에이전트와 상호작용 시 조작할 객체는 제한적

2. 작은 단위의 점진적 행위가 아닌 묶음 단위로 처리됨

3. 작은 요소를 하나씩 되돌리는 것이 어려움

SO. 4가지 조건을 만족하지 않으므로, GUI와 다름

BUT. 이것이 문제는 아님 [CLI도 마찬가지이나, 대표 인터페이스임]

도널드 노먼 관점

에이전트 계층은 비결정론적 시스템임

SO. 노먼의 전제에 어긋나므로, 사용자는 모형을 형성하기 어려움

CLI & GUI

1. 모든 명령어를 외우지 않아도, 시스템 구조를 이해하면 새로운 명령어 조합이 가능

2. 시스템의 작동 방식을 '유한한 규칙들의 조합'으로 이해, 스스로 규칙 조합하여 추론 가능

→ 에이전트형 운영체제 != CLI & GUI

3. 에이전트형 운영체제의 분석

기술 발전으로 에이전트가 결정론적 시스템에 수렴한다면?

근본적인 모형 형성의 어려움

시스템의 부분적 이해가 전체의 추론으로 이어져야 함

에이전트가 작업을 수행한 과정은 단순 정보에 불과함

에이전트는 자연어에 기반,

자연어는 유한하게 환원시키기 어려운 맥락에 의존

결정론적 시스템임에도 한계점

날씨는 결정론적임—같은 조건에서 같은 결과가 물리법칙으로부터 나옴

BUT. 우리는 날씨에 대한 모형을 가지지 못함 / 경험적 기대뿐

날씨가 비결정론적이어서가 아님,

날씨의 형성 과정을 학습할 수 없기 때문

4. 결론

“에이전트형 운영체제는 CLI와 GUI의 후속작이 아님.

둘은 겉모습은 달라도 사용자가 유한한 규칙을 익혀 조합할 수 있다는 공통점을 지님.”

“에이전트형 운영체제는 그렇지 않음.

에이전트가 발전해 결정론적 시스템에 수렴하는 경우까지 논하였음.”

“다만 이것이 결함은 아니라, 이를 발전시킬 새로운 논의가 필요하다는 뜻임.”

“그 논의 속에서 ‘운영체제’라는 이름의 적합성을 점차 찾아가길 기대함.”

References

Norman, Donald A., *The Design of Everyday Things*, New York: Basic Books, 2013.

Shneiderman, Ben. “Direct Manipulation: A Step Beyond Programming Languages,” IEEE Computer 16(8), 1983, pp.17-69.