에서 에이전트 AI의 기초 AWS

AWS 권장 가이드

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS 권장 가이드: 에서 에이전트 AI의 기초 AWS

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon의 상표 및 트레이드 드레스는 Amazon 외 제품 또는 서비스와 함께, Amazon 브랜드 이미지를 떨어뜨리거나 고객에게 혼동을 일으킬 수 있는 방식으로 사용할 수 없습니다. Amazon이 소유하지 않은 기타 모든 상표는 Amazon과 제휴 관계이거나 관련이 있거나 후원 관계와 관계없이 해당 소유자의 자산입니다.

Table of Contents

에서 에이전트 AI의 기초 AWS 1
대상 독자 2
목표 2
이 콘텐츠 시리즈 정보 2
소프트웨어 에이전트 소개 3
자율성에서 분산형 인텔리전스로 3
자율성의 초기 개념 3
액터 모델 및 비동기 실행 4
분산형 인텔리전스 및 다중 에이전트 시스템 4
Nwana의 유형과 소프트웨어 에이전트의 부상 4
Nwana의 에이전트 유형 5
유형부터 최신 에이전트 원칙까지 5
최신 소프트웨어 에이전트의 세 가지 원칙 6
자율성 6
비동기성 6
정의 원칙으로서의 기관 7
용도가 있는 기관 7
소프트웨어 에이전트의 목적 8
액터 모델에서 에이전트 인식으로 8
에이전트 함수: 인지, 이유, 행동 8
자율 협업 및 의도성 9
의도 위임 10
예측할 수 없는 동적 환경에서 운영 10
인간의 인지 부하 줄이기 10
분산 인텔리전스 활성화 4
반응뿐만 아니라 목적을 가지고 행동하기 11
소프트웨어 에이전트의 진화 12
소프트웨어 에이전트의 기반 13
1959 - 올리버 셀프리지: 소프트웨어 자율성의 탄생 13
1973년 - Carl Hewitt: 액터 모델 13
필드 성숙: 추론에서 행동으로 13
1977 - Victor Lesser: 다중 에이전트 시스템 13
1990년대 - Michael™dridge 및 Nicholas Jennings: 에이전트 스펙트럼 13
1996 - Hyacinth S. Nwana: 에이전트 개념 공식화 14

병렬 타임라인: 대규모 언어 모델의 부상 14
타임라인 수렴: 에이전트 AI의 등장 15
2023-2024 - 엔터프라이즈급 에이전트 플랫폼 15
2025년 1월~6월 - 엔터프라이즈 기능 확장 15
출현 - 에이전트 AI 15
에이전트 AI에 대한 소프트웨어 에이전트 17
소프트웨어 에이전트의 핵심 구성 요소 17
인식 모듈 18
인지 모듈 19
작업 모듈 20
학습 모듈 20
기존 에이전트 아키텍처: 인지, 이유, 행동 21
인식 모듈 22
이유 모듈 22
Act 모듈 23
생성형 AI 에이전트: 심볼 로직LLMs으로 대체 23
주요 개선 사항 24
LLM 기반 에이전트에서 장기 메모리 확보 24
에이전트 AI의 결합된 이점 25
기존 AI와 소프트웨어 에이전트 및 에이전트 AI 비교 26
다음 단계 28
리소스 29
AWS 참조 29
기타 참조 29
문서 기록 31
..... xxxii

에서 에이전트 AI의 기초 AWS

Aaron Sempf, Amazon Web Services

2025년 7월(문서 기록)

점점 더 지능적이고 분산된 자율 시스템의 세계에서 환경, 상태에 대한 이유를 인식하고 의도에 따라 행동할 수 있는 엔터티인 에이전트의 개념은 기초가 되었습니다. 에이전트는 단순히 지침을 실행하는 프로그램이 아니라 사용자, 시스템 또는 조직을 대신하여 결정을 내리는 목표 지향적인 컨텍스트 인식 엔터티입니다. 이들의 출현은 소프트웨어를 구축하고 생각하는 방식의 변화를 반영합니다. 즉, 절차적 로직 및 사후 대응 자동화에서 자율성과 목적으로 작동하는 시스템으로의 전환입니다.

AI, 분산 시스템 및 소프트웨어 엔지니어링의 교차점에는 에이전트 AI라고 하는 강력한 패러다임이 있습니다. 이 차세대 지능형 시스템은 적응형 동작, 복잡한 조정 및 위임된 의사 결정을 수행할 수 있는 소프트웨어 에이전트로 구성됩니다.

이 가이드에서는 최신 소프트웨어 에이전트를 정의하는 원칙을 소개하고 에이전트 AI에 대한 진화를 간략하게 설명합니다. 이 변화를 설명하기 위해 가이드는 개념적 배경을 제공한 다음 소프트웨어 에이전트의 에이전트 AI로의 진화를 추적합니다.

- 소프트웨어 에이전트 소개는 소프트웨어 에이전트를 정의하고, 이를 기존 소프트웨어 구성 요소와 비교하며, 확립된 프레임워크에서 끌어와 에이전트 행동을 기존 자동화와 차별화하는 필수 특성을 소개합니다.
- 소프트웨어 에이전트의 목적은 소프트웨어 에이전트가 존재하는 이유, 수행하는 역할, 해결하는 문제, 지능형 위임을 활성화하고, 인지 부하를 줄이고, 동적 환경에서 적응형 동작을 지원하는 방법을 검사합니다.
- 소프트웨어 에이전트의 진화는 자율성 및 동시성의 초기 개념부터 다중 에이전트 시스템 및 공식 에이전트 아키텍처의 출현에 이르기까지 소프트웨어 에이전트를 형성한 지적 및 기술적 이정표를 추적하여 생성형 AI와 수렴합니다.
- 에이전트 AI에 대한 소프트웨어 에이전트는 분산 에이전트 모델을 파운데이션 모델, 서버리스 컴퓨팅 및 오케스트레이션 프로토콜과 결합하는 수십 년의 발전의 정점으로 에이전트 AI를 도입합니다. 이 섹션에서는 이러한 수렴을 통해 자율성, 비동기성 및 실제 기관 규모로 작동하는 새로운 세대의 지능형 도구 사용 에이전트를 활성화하는 방법을 설명합니다.

대상 독자

이 가이드는 최신 클라우드 솔루션을 위해이 기술을 채택하기 전에 소프트웨어 에이전트의 역사, 주요 개념 및 에이전트 AI로의 진화를 이해하려는 아키텍트, 개발자 및 기술 리더를 위해 설계되었습니다 AWS.

목표

에이전트 아키텍처를 채택하면 조직이 다음과 같은 이점을 얻을 수 있습니다.

- 가치 실현 시간 단축: 지식 작업을 자동화 및 확장하고 수동 작업 및 지연 시간을 줄입니다.
- 고객 참여 개선: 도메인 간에 지능형 어시스턴트를 제공합니다.
- 운영 비용 절감: 이전에 사람의 의견이나 감독이 필요했던 의사 결정 흐름을 자동화합니다.
- 혁신 및 차별화 추진: 실시간으로 적응하고, 배우고, 경쟁하는 지능형 제품을 구축합니다.
- 레거시 워크플로 현대화: 스크립트와 모놀리스를 모듈식 추론 에이전트로 재구성합니다.

이 콘텐츠 시리즈 정보

이 가이드는 에이전트 AI on에 대한 시리즈의 일부입니다 AWS. 자세한 내용과이 시리즈의 다른 가이드를 보려면 AWS 권장 가이드 웹 사이트의 에이전트 AI를 참조하세요.

소프트웨어 에이전트 소개

소프트웨어 에이전트의 개념은 1960년대 자율 엔터티의 기반에서 1990년대 초 공식 탐색으로 크게 발전했습니다. 결정론적 스크립트에서 적응형 지능형 애플리케이션에 이르기까지 디지털 시스템이 점점 더 복잡해짐에 따라 소프트웨어 에이전트는 컴퓨팅 시스템에서 자율적이고 컨텍스트를 인식하며 목표 기반 동작을 활성화하기 위한 필수 구성 요소가 되었습니다. 클라우드 네이티브 및 AI 강화 아키텍처의 맥락에서, 특히 생성형 AI, 대규모 언어 모델(LLMs) 및 Amazon Bedrock과 같은 플랫폼이 등장함에 따라 소프트웨어 에이전트는 새로운 기능 및 규모 렌즈를 통해 재정의되고 있습니다.

이 소개는 Hyacinth S. Nwana(Nwana 1996)의 주요 작업인 Software Agents: An Overview에서 발췌한 것입니다. 소프트웨어 에이전트를 정의하고, 개념적 루트를 논의하며, 논의를 현대적 프레임워크로 확장하여 최신 소프트웨어 에이전트의 세 가지 중요한 원칙인 자율성, 비동기성 및 기관을 정의합니다. 이러한 원칙은 소프트웨어 에이전트를 다른 유형의 서비스 또는 애플리케이션과 구분하며, 이러한 에이전트가 분산된 실시간 환경에서 목적, 복원력 및 인텔리전스를 바탕으로 운영할 수 있도록 합니다.

이 섹션의 내용

- 자율성에서 분산형 인텔리전스로
- Nwana의 유형과 소프트웨어 에이전트의 부상
- 최신 소프트웨어 에이전트의 세 가지 원칙

자율성에서 분산형 인텔리전스로

소프트웨어 에이전트라는 용어가 주류에 진입하기 전에 초기 컴퓨팅 연구를 통해 독립적으로 행동하고, 입력에 반응하고, 내부 규칙 또는 목표에 따라 결정을 내릴 수 있는 시스템인 자율 디지털 엔터티에 대한 아이디어를 탐색했습니다. 이러한 초기 아이디어는 에이전트 패러다임이 될 것에 대한 개념적 토대를 마련했습니다. (기록 타임라인은이 가이드의 뒷부분에 있는 소프트웨어 에이전트의 진화 섹션을 참조하세요.)

자율성의 초기 개념

인간 운영자와 독립적으로 작동하는 기계 또는 프로그램의 개념은 수십 년 동안 시스템 디자이너에게 흥미를 불러일으켰습니다. 사이버넷, 인공 지능 및 제어 시스템의 초기 작업에서는 소프트웨어가 자체 규제 동작을 보이고, 변화에 동적으로 대응하고, 지속적인 사람의 감독 없이 운영할 수 있는 방법을 살펴보았습니다.

이러한 아이디어는 지능형 시스템의 핵심 속성으로 자율성을 도입하고 대응 또는 실행만 하는 대신 결정하고 조치를 취할 수 있는 소프트웨어 출현의 단계를 설정했습니다.

액터 모델 및 비동기 실행

1970년대에 인공 지능용 범용 모듈식 ACTOR 격식(Hewitt et al. 1973) 논문에 도입된 액터 모델은 분산된 메시지 기반 계산을 고려하기 위한 공식 프레임워크를 제공했습니다. 이 모델에서 액터는 비동기식 메시지를 전달하여 독점적으로 통신하고 확장 가능하고 동시적이며 내결함성 시스템을 지원하는 독립 엔터티입니다.

액터 모델은 현대 에이전트 설계에 계속 영향을 미치는 세 가지 주요 속성을 강조했습니다.

- 상태 및 동작 격리
- 개체 간의 비동기 상호 작용
- 작업의 동적 생성 및 위임

이러한 속성은 분산 시스템의 요구 사항에 부합하며 클라우드 네이티브 환경에서 소프트웨어 에이전트의 운영 특성을 미리 구성했습니다.

분산형 인텔리전스 및 다중 에이전트 시스템

1960년대 이후 컴퓨팅 시스템이 더 상호 연결됨에 따라 연구원은 분산형 인공 지능(DAI)을 탐색했습니다. 이 필드는 여러 자율 개체가 시스템 전체에서 협업하거나 경쟁적으로 작동하는 방식에 중점을 두었습니다. DAI는 각 에이전트가 로컬 목표, 지각 및 추론을 가지고 있지만 더 광범위하고 상호 연결된 환경에서도 운영되는 다중 에이전트 시스템을 개발했습니다.

의사 결정이 분산되고 에이전트 상호 작용으로 인해 새로운 동작이 발생하는 분산 인텔리전스에 대한 이러한 비전은 최신 에이전트 기반 시스템을 구상하고 구축하는 방법의 핵심으로 남아 있습니다.

Nwana의 유형과 소프트웨어 에이전트의 부상

1990년대 중반 소프트웨어 에이전트 개념의 공식화는 지능형 시스템의 진화에서 전환점을 보여주었습니다. 이 공식화에 대한 가장 영향력 있는 기여 중 하나는 Hyacinth S. Nwana의 논문인 Software Agents: An Overview (Nwana 1996)입니다.이 논문은 다양한 차원에서 소프트웨어 에이전트를 분류하고 이해하기 위한 최초의 포괄적인 프레임워크 중 하나를 제공했습니다.

이 백서에서 Nwana는 소프트웨어 에이전트 연구 상태를 설문 조사하고 에이전트가 정의되고 구현되는 방식에서 점점 더 다양해지는 것을 식별합니다. 이 백서는 일반적인 개념 프레임워크의 필요성을 강조하고 주요 기능에 따라 에이전트를 분류하는 유형을 제안합니다. 학계 및 산업의 대표적인 에이전트 시스템을 검토하고, 에이전트를 기존 프로그램 및 객체와 구분하며, 에이전트 기반 컴퓨팅의 과제와 기회를 간략하게 설명합니다.

Nwana는 소프트웨어 에이전트가 모놀리식 개념은 아니지만 정교함과 기능의 스펙트럼을 따라 존재한다는 점을 강조합니다. 이 유형은 이러한 환경을 명확히 하고 향후 설계 및 연구를 안내하는 역할을 합니다.

Nwana는 소프트웨어 에이전트를 특정 환경에서 지속적이고 자율적으로 작동하는 소프트웨어 엔터티로 정의합니다.이 엔터티는 종종 다른 에이전트와 프로세스에서 서식합니다. 이 정의는 두 가지 주요 특성을 강조합니다.

- 연속성: 에이전트는 지속적인 사람의 개입 없이 시간이 지남에 따라 지속적으로 작동합니다.
- 자율성: 에이전트는 환경에 대한 인식에 따라 독립적으로 결정을 내리고 조치를 취할 수 있습니다.

이 정의는 Nwana의 에이전트 유형과 결합되어 에이전트의 기본 특성으로서 위임된 권한(자율성을 통해)과 적극성을 강조합니다. 직접 명령에만 응답하는 대신 다른 엔터티를 대신하여 독립적으로 행동하고 목표를 추구하는 동작을 시작하는 에이전트의 능력을 강조하여 에이전트와 하위 루틴 또는 서비스를 구분합니다.

Nwana의 에이전트 유형

다양한 유형의 에이전트를 추가로 구분하기 위해 Nwana는 6가지 주요 속성을 기반으로 분류 시스템을 도입합니다.

- 자율성: 에이전트는 사람 또는 다른 사람의 직접적인 개입 없이 작동합니다.
- 소셜 기능: 에이전트는 통신 메커니즘을 사용하여 다른 에이전트 또는 사람과 상호 작용합니다.
- 반응성: 에이전트가 환경을 인식하고 적시에 응답합니다.
- 적극성: 에이전트가 이니셔티브를 통해 목표 지향적인 행동을 보입니다.
- 적응성 및 학습: 에이전트는 경험을 통해 시간이 지남에 따라 성능을 개선합니다.
- 이동성: 에이전트가 다양한 시스템 환경 또는 네트워크를 이동할 수 있습니다.

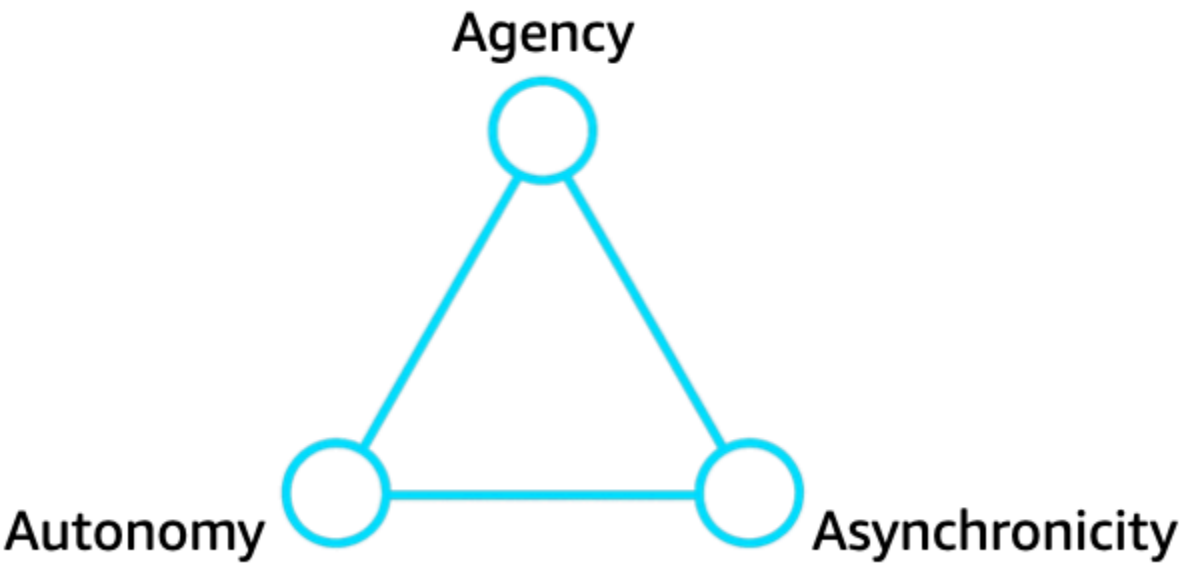
유형부터 최신 에이전트 원칙까지

Nwana의 작업은 분류 체계이자 컴퓨팅 커뮤니티가 소프트웨어에서 진화하는 형태의 기관을 평가할 수 있는 기본 렌즈 역할을 했습니다. 자율성, 능동성, 사용자 또는 시스템을 대신하여 행동하는 개념에 중점을 둔 그는 이제 에이전트적 행동을 고려하는 토대를 마련했습니다.

기술과 환경이 변경되었지만, 특히 생성형 AI, 서버리스 인프라 및 다중 에이전트 오케스트레이션 프레임워크가 증가함에 따라 Nwana 작업의 기본 인사이트는 여전히 관련이 있습니다. 초기 에이전트 이론과 소프트웨어 에이전트의 세 가지 최신 원칙 사이에 중요한 다리를 제공합니다.

최신 소프트웨어 에이전트의 세 가지 원칙

오늘날의 AI 기반 플랫폼, 마이크로서비스 아키텍처 및 이벤트 기반 시스템의 맥락에서 소프트웨어 에이전트는 표준 서비스 또는 자동화 스크립트와 구별되는 세 가지 상호 의존적 원칙, 즉 자율성, 비동기성 및 기관으로 정의할 수 있습니다. 다음 그림과 후속 다이어그램에서 삼각형은 현대 소프트웨어 에이전트의 세 가지 원칙을 나타냅니다.



자율성

최신 에이전트는 독립적으로 작동합니다. 인적 프롬프트 없이 내부 상태 및 환경 컨텍스트를 기반으로 결정을 내립니다. 이를 통해 실시간으로 데이터에 대응하고, 자체 수명 주기를 관리하고, 목표 및 상황 입력에 따라 동작을 조정할 수 있습니다.

자율성은 에이전트 행동의 기반입니다. 이를 통해 에이전트는 지속적인 감독이나 하드 코딩된 제어 흐름 없이 작동할 수 있습니다.

비동기성

에이전트는 기본적으로 비동기식입니다. 즉, 호출 차단이나 선형 워크플로에 의존하지 않고 이벤트, 신호 및 자극에 반응합니다. 이 특성을 통해 확장 가능한 비차단 통신, 분산 환경의 응답성, 구성 요소 간의 느슨한 결합이 가능합니다.

에이전트는 비동기성을 통해 실시간 시스템에 참여하고 다른 서비스 또는 에이전트와 유연하고 효율적으로 조정할 수 있습니다.

정의 원칙으로서의 기관

자율성과 비동기성이 필요하지만 이러한 기능만으로는 시스템을 실제 소프트웨어 에이전트로 만들기에 충분하지 않습니다. 중요한 차별화 요소는 다음을 도입하는 기관입니다.

- 목표 지향적 행동: 에이전트는 목표를 추구하고 목표에 대한 진행 상황을 평가합니다.
- 의사 결정: 에이전트는 옵션을 평가하고 규칙, 모델 또는 학습된 정책에 따라 작업을 선택합니다.
- 위임된 의도: 에이전트는 사람, 시스템 또는 조직을 대신하여 행동하며 목적 의식이 내재되어 있습니다.
- 컨텍스트 추론: 에이전트는 환경의 메모리 또는 모델을 통합하여 행동을 지능적으로 안내합니다.

자율적이고 비동기적인 시스템은 여전히 사후 대응 서비스일 수 있습니다. 소프트웨어 에이전트가 되는 이유는 에이전트가 되기 위해 의도와 목적으로 행동하는 능력입니다.

용도가 있는 기관

자율성, 비동기성 및 기관 원칙을 통해 시스템은 분산 환경에서 지능적이고 적응적이며 독립적으로 작동할 수 있습니다. 이러한 원칙은 수십 년간의 개념 및 아키텍처 진화에 기반을 두고 있으며, 이제 오늘날 구축되고 있는 많은 최첨단 AI 시스템을 뒷받침합니다.

생성형 AI, 목표 지향 오케스트레이션 및 다중 에이전트 협업의이 새로운 시대에서는 소프트웨어 에이전트를 진정으로 에이전트로 만드는 요소를 이해하는 것이 중요합니다. 기관을 정의 특성으로 인식하면 자동화를 넘어 목적이 있는 자율 지능 영역으로 전환하는 데 도움이 됩니다.

소프트웨어 에이전트의 목적

최신 시스템이 점점 더 복잡해지고 분산되고 지능화됨에 따라 소프트웨어 에이전트의 역할은 자율 운영부터 사용자 지원 기술에 이르기까지 다양한 도메인에서 두드러졌습니다. 하지만 소프트웨어 에이전트의 기본 목적은 무엇인가요? 스크립트, 서비스 또는 정적 모델을 넘어서는 시스템을 설계하는 대신 인식, 추론 및 조치를 수행할 수 있는 엔터티에 작업을 위임하는 이유는 무엇입니까?

이 섹션에서는 소프트웨어 에이전트의 기본 목적인 자율성, 적응성 및 의도적인 작업에 중점을 두고 동적 환경 내에서 작업을 지능적으로 위임할 수 있도록 하는 방법을 살펴봅니다. 소프트웨어 에이전트의 개념적 기반을 소개하고, 인지 구조를 추적하며, 해결할 수 있는 고유한 장비를 갖춘 실제 문제를 간략하게 설명합니다.

이 섹션의 내용

- 액터 모델에서 에이전트 인식으로
- 에이전트 함수: 인지, 이유, 행동
- 자율 협업 및 의도성

액터 모델에서 에이전트 인식으로

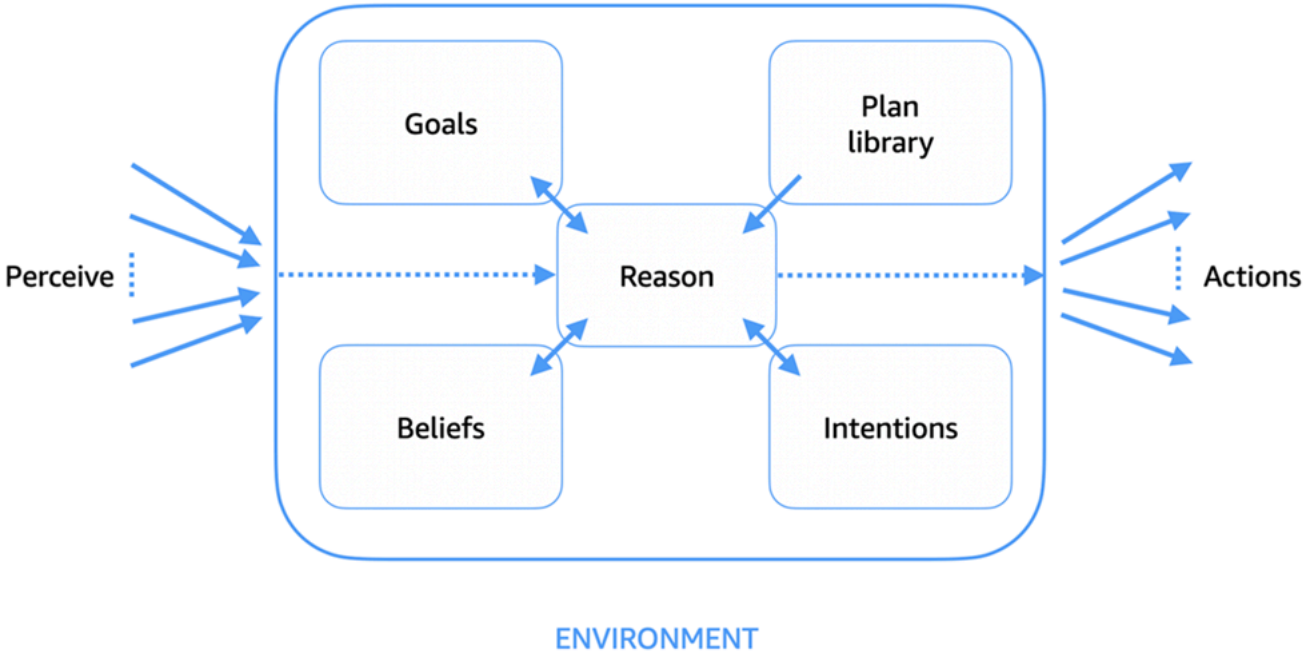
소프트웨어 에이전트의 목적과 구조는 초기 계산 모델, 특히 1970년대에 Carl Hewitt가 도입한 액터 모델에서 나온 아이디어에 기반을 두고 있습니다(Hewitt 외 1973).

액터 모델은 계산을 액터라고 하는 독립적이고 동시에 실행되는 엔터티의 모음으로 취급합니다. 각 액터는 자체 상태를 캡슐화하고, 비동기식 메시지 전달을 통해서만 상호 작용하며, 새 액터를 생성하고 작업을 위임할 수 있습니다.

이 모델은 분산 추론, 반응성 및 격리를 위한 개념적 기반을 제공했으며,이 모든 것이 최신 소프트웨어 에이전트의 행동 아키텍처를 뒷받침합니다.

에이전트 함수: 인지, 이유, 행동

모든 소프트웨어 에이전트의 핵심은 인식, 이유, 행동 루프로 설명되는 인지 주기입니다. 이 프로세스는 다음 다이어그램에 설명되어 있습니다. 동적 환경에서 에이전트가 자율적으로 작동하는 방식을 정의합니다.



- 지각: 에이전트는 환경에서 정보(예: 이벤트, 센서 입력 또는 API 신호)를 수집하고 내부 상태 또는 신념을 업데이트합니다.
- 이유: 에이전트는 계획 라이브러리 또는 로직 시스템을 사용하여 현재 신념, 목표 및 컨텍스트 지식을 분석합니다. 이 프로세스에는 목표 우선순위 지정, 충돌 해결 또는 의도 선택이 포함될 수 있습니다.
- 조치: 에이전트는 위임된 목표 달성에 더 가까워지는 조치를 선택하고 실행합니다.

이 아키텍처는 에이전트가 엄격한 프로그래밍을 넘어 작동할 수 있는 기능을 지원하며 유연하고 상황에 민감하며 목표 지향적인 동작을 지원합니다. 소프트웨어 에이전트의 더 광범위한 목적을 안내하는 멘탈 프레임워크를 형성합니다.

자율 협업 및 의도성

소프트웨어 에이전트의 목적은 최신 컴퓨팅에 자율성, 컨텍스트 인식 및 지능형 위임을 제공하는 것입니다. 에이전트는 액터 모델의 원칙을 기반으로 구축되고 인식, 근거, 행동 주기에 구현되므로 사후 대응적일 뿐만 아니라 선제적이고 목적 지향적인 시스템을 활성화합니다.

에이전트는 소프트웨어가 복잡한 환경에서 결정, 적응 및 조치를 취할 수 있도록 지원합니다. 사용자를 대표하고, 목표를 해석하고, 기계 속도로 작업을 구현합니다. 에이전트 AI의 시대로 깊이 들어가면서 소프트웨어 에이전트는 인간의 의도와 지능형 디지털 행동 간의 운영 인터페이스가 되고 있습니다.

의도 위임

기존 소프트웨어 구성 요소와 달리 소프트웨어 에이전트는 사용자, 다른 시스템 또는 상위 수준 서비스 등 다른 것을 대신하여 작업합니다. 위임된 의도를 가지고 있습니다. 즉, 다음과 같습니다.

- 시작 후 독립적으로 작동합니다.
- 위임자의 목표에 맞게 선택합니다.
- 실행 시 불확실성과 장단점을 탐색합니다.

에이전트는 지침과 결과 간의 격차를 해소하므로 사용자는 명시적 지침이 필요하지 않고 더 높은 수준의 추상화에서 의도를 표현할 수 있습니다.

예측할 수 없는 동적 환경에서 운영

소프트웨어 에이전트는 조건이 지속적으로 변경되고 데이터가 실시간으로 도착하며 제어 및 컨텍스트가 분산되는 환경을 위해 설계되었습니다.

정확한 입력 또는 동기식 실행이 필요한 정적 프로그램과 달리 에이전트는 주변 환경에 적응하고 동적으로 응답합니다. 이는 클라우드 네이티브 인프라, 엣지 컴퓨팅, 사물 인터넷(IoT) 네트워크 및 실시간 의사 결정 시스템에서 중요한 기능입니다.

인간의 인지 부하 줄이기

소프트웨어 에이전트의 주요 목적 중 하나는 인간에 대한 인지 및 운영 부담을 줄이는 것입니다. 에이전트는 다음을 수행할 수 있습니다.

- 시스템 및 워크플로를 지속적으로 모니터링합니다.
- 사전 정의되거나 긴급한 상태를 감지하고 이에 대응합니다.
- 반복적이고 대량의 결정을 자동화합니다.
- 지연 시간을 최소화하면서 환경 변화에 대응합니다.

의사 결정이 사용자에서 에이전트로 전환되면 시스템은 더 응답적이고 복원력이 뛰어나며 인간 중심적이 되고 새로운 정보 또는 중단에 실시간으로 적응할 수 있습니다. 이를 통해 복잡한 환경이나 대규모 환경에서 반응 처리 시간을 단축하고 운영 연속성을 높일 수 있습니다. 그 결과 마이크로 수준의 의사 결정에서 전략적 감독 및 창의적인 문제 해결로 인적 초점이 전환됩니다.

분산 인텔리전스 활성화

소프트웨어 에이전트가 개별적으로 또는 집합적으로 운영할 수 있으므로 환경 또는 조직 간에 조정되는 다중 에이전트 시스템(MAS)을 설계할 수 있습니다. 이러한 시스템은 작업을 지능적으로 분산하고 복합 목표를 향해 협상, 협력 또는 경쟁할 수 있습니다.

예를 들어 글로벌 공급망 시스템에서는 개별 에이전트가 공장, 배송, 창고 및 라스트 마일 배송을 관리합니다. 각 에이전트는 로컬 자율성으로 운영됩니다. 팩토리 에이전트는 리소스 제약 조건에 따라 프로덕션을 최적화하고, 창고 에이전트는 인벤토리 흐름을 실시간으로 조정하며, 배송 에이전트는 트래픽 및 고객 가용성에 따라 배송 경로를 변경합니다.

이러한 에이전트는 동적으로 통신하고 조정하며 중앙 집중식 제어 없이 포트 지연 또는 트럭 고장과 같은 중단에 적응합니다. 시스템의 전체 인텔리전스는 이러한 상호 작용에서 비롯되며 단일 구성 요소의 기능을 넘어서는 복원력이 뛰어나고 최적화된 물류를 가능하게 합니다.

이 모델에서 에이전트는 더 광범위한 인텔리전스 패브릭에서 노드 역할을 합니다. 단일 구성 요소만으로는 처리할 수 없는 문제를 해결할 수 있는 새로운 시스템을 형성합니다.

반응뿐만 아니라 목적을 가지고 행동하기

복잡한 시스템에서는 자동화만으로는 충분하지 않습니다. 소프트웨어 에이전트의 정의 목적은 목적을 가지고 행동하고, 목표를 평가하고, 컨텍스트를 평가하고, 정보에 입각한 선택을 하는 것입니다. 즉, 소프트웨어 에이전트는 트리거에만 응답하는 대신 목표를 추구합니다. 경험이나 피드백을 기반으로 신념과 의도를 수정할 수 있습니다. 이 컨텍스트에서 신념은 에이전트의 지각(입력 및 센서)을 기반으로 환경에 대한 에이전트의 내부 표현(예: “패키지 X는 창고 A에 있음”)을 나타냅니다. 의도는 에이전트가 목표를 달성하기 위해 선택한 계획을 나타냅니다(예: "전달 경로 B를 사용하고 수신자에게 알림"). 에이전트는 필요에 따라 작업을 에스컬레이션, 연기 또는 조정할 수도 있습니다.

이러한 의도로 인해 소프트웨어 에이전트는 사후 대응 실행기뿐만 아니라 지능형 시스템의 자율 공동 작업자가 됩니다.

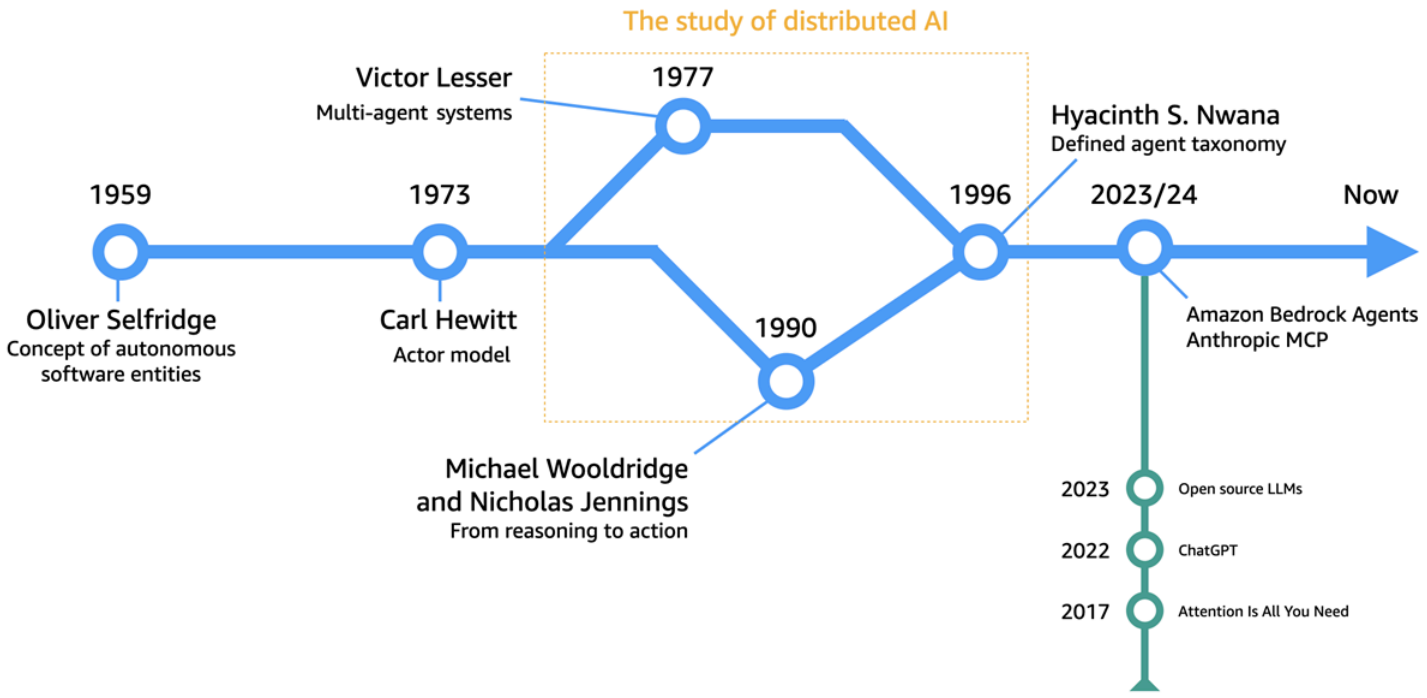
소프트웨어 에이전트의 진화

단순한 자동화 시스템에서 지능형, 자율 및 목표 지향 소프트웨어 에이전트로의 여정은 컴퓨터 과학, 인공 지능 및 분산 시스템의 수십 년 진화를 반영합니다.

그런 다음 기계 학습이 증가하여 패러다임이 수제 규칙에서 통계 패턴 인식으로 바뀌었습니다. 이러한 시스템은 데이터에서 학습할 수 있으며 지각, 분류 및 의사 결정의 발전을 가능하게 했습니다.

대규모 언어 모델(LLMs)은 규모, 아키텍처 및 비지도 학습의 수렴을 나타냅니다. LLMs 작업별 훈련을 거의 또는 전혀 하지 않고 작업을 추론, 생성 및 조정할 수 있습니다. LLMs 확장 가능한 클라우드 네이티브 인프라 및 구성 가능한 아키텍처와 결합하면 이제 엔터프라이즈 규모에서 자율성, 컨텍스트 인식 및 적응성을 갖춘 지능형 소프트웨어 에이전트인 에이전트 AI의 전체 비전을 달성할 수 있습니다.

이 섹션에서는 다음 다이어그램과 같이 기본 이론부터 최신 관행까지 소프트웨어 에이전트의 기록을 살펴봅니다. 분산형 인공 지능(DAI)과 변환기 기반 생성형 AI의 수렴을 강조하고 에이전트 AI의 출현을 형성한 주요 마일스톤을 식별합니다.



이 섹션의 내용

- 소프트웨어 에이전트의 기반
- 필드 성숙: 추론에서 행동으로
- 병렬 타임라인: 대규모 언어 모델의 부상

- 타임라인 수렴: 에이전트 AI의 등장

소프트웨어 에이전트의 기반

1959 - 올리버 셀프리지: 소프트웨어 자율성의 탄생

소프트웨어 에이전트의 루트는 환경을 인식하고 독립적으로 행동할 수 있는 자율 소프트웨어 엔터티(데몬)의 개념을 도입한 올리버 셀프리지로 거슬러 올라갑니다(Selfridge 1959). 기계 지각 및 학습에 대한 초기 작업은 독립적이고 지능적인 시스템으로 에이전트의 미래 개념에 대한 철학적 토대를 마련했습니다.

1973년 - Carl Hewitt: 액터 모델

핵심 발전은 에이전트를 독립적인 동시 엔터티로 설명하는 공식 계산 모델인 Carl Hewitt의 액터 모델(Hewitt et al. 1973)과 함께 이루어졌습니다. 이 모델에서 에이전트는 자신의 상태 및 동작을 캡슐화하고, 비동기식 메시지 전달을 사용하여 통신하고, 다른 액터를 동적으로 생성하고, 작업을 위임할 수 있습니다.

액터 모델은 분산된 에이전트 기반 시스템에 대한 이론적 기초와 아키텍처 패러다임을 모두 제공했습니다. 이 모델은 Erlang 프로그래밍 언어 및 Akka 프레임워크와 같은 최신 동시성 구현을 사전 설계했습니다.

필드 성숙: 추론에서 행동으로

1977 - Victor Lesser: 다중 에이전트 시스템

1970년대 후반에 분산형 인공 지능(DAI)이 등장했습니다. 다중 에이전트 시스템(MAS)을 혁신하는 것으로 널리 인정받는 Victor Lesser의 지지를 받았습니다. 그는 독립 소프트웨어 엔터티가 협력, 조정 및 협상하는 방법에 초점을 맞추었습니다(리소스 섹션 참조). 이러한 개발로 인해 복잡한 문제를 집합적으로 해결할 수 있었던 시스템은 분산 인텔리전스를 구축하는 데 필수적인 도약이었습니다.

1990년대 - Michael™dridge 및 Nicholas Jennings: 에이전트 스펙트럼

1990년대까지 분산 인텔리전스 분야는 Michael™dridge 및 Nicholas Jennings와 같은 연구자의 기여로 성숙해졌습니다. 이러한 학자들은 비인지 시스템에서 목표 중심의 추론 에이전트(Wooldridge and Jennings 1995)에 이르기까지 스펙트럼을 따라 에이전트를 분류했습니다. 이들의 작업은 에이전트가

더 이상 추상적인 아이디어가 아니라 로봇 공학부터 엔터프라이즈 소프트웨어에 이르기까지 다양한 실용적인 도메인에 적용되고 있음을 강조했습니다.

또한 이러한 연구자는 중앙 집중식 추론에서 분산 작업으로의 전환에 중점을 두었습니다. 에이전트는 더 이상 단순한 사상가가 아니라 자율성과 목적을 갖춘 실시간 환경에서 운영되는 사람이었습니다.

1996 - Hyacinth S. Nwana: 에이전트 개념 공식화

1996년, Hyacinth S. Nwana는 지금까지 가장 포괄적인 에이전트 분류를 제공한 영향력 있는 소프트웨어 에이전트: 개요를 발표했습니다. 그의 유형에는 자율성, 사회적 능력, 반응성, 능동성, 학습 및 이동성과 같은 속성이 포함되었으며, 소프트웨어 에이전트와 기존 소프트웨어 구문을 구분했습니다.

또한 Nwana는 이제 널리 받아들여지는 정의를 제공했습니다. 즉, 소프트웨어 에이전트는 위임 개념에서 파생된 기관 관계의 사용자 또는 기타 프로그램을 위해 작동하는 소프트웨어 기반 컴퓨터 프로그램입니다.

이 공식화는 소프트웨어 에이전트를 이론적 구성에서 실제 애플리케이션으로 전환하는 데 중요한 역할을 했습니다. 이를 통해 통신, 워크플로 자동화, 지능형 어시스턴트와 같은 여러 분야에서 에이전트 기반 시스템이 생성되었습니다.

Nwana의 작업은 초기 분산 AI 연구와 현대 에이전트의 운영 아키텍처의 수렴 지점에 있습니다. 이는 에이전트의 인지 이론과 오늘날 시스템에서의 실제 배포 간의 중요한 연결고리입니다.

병렬 타임라인: 대규모 언어 모델의 부상

에이전트 프레임워크가 발전하는 동안 자연어 처리 및 기계 학습에서 병렬적이고 수렴적인 혁신이 이루어졌습니다.

- 2017 - 변환기: 필요한 모든 것에 주의를 기울이는 백서(Vaswani et al. 2017)는 변환기 아키텍처를 도입하여 기계가 언어를 처리하고 생성하는 방식을 크게 개선했습니다.
- 2022년 - ChatGPT: OpenAI는 ChatGPT라는 GPT-3.5에 대한 채팅 기반 인터페이스를 출시하여 범용 AI 시스템과의 자연스러운 대화형 대화를 가능하게 했습니다.
- 2023 - 오픈 소스 LLMs: Llama, Falcon 및 Mistral 릴리스는 강력한 모델에 널리 액세스하고 오픈 소스 및 엔터프라이즈 환경에서 에이전트 프레임워크 개발을 가속화했습니다.

이러한 혁신을 통해 언어 모델은 컨텍스트를 구문 분석하고, 작업을 계획하고, 응답을 연결할 수 있는 추론 엔진으로 바뀌었으며, LLMs 지능형 소프트웨어 에이전트의 주요 조력자가 되었습니다.

타임라인 수렴: 에이전트 AI의 등장

2023-2024 - 엔터프라이즈급 에이전트 플랫폼

분산 소프트웨어 에이전트 아키텍처와 변환기 기반 LLMs의 수렴은 에이전트 AI의 부상에 절정입니다.

- Amazon Bedrock Agents는 Amazon Bedrock의 파운데이션 모델을 사용하여 목표 기반 도구 사용 소프트웨어 에이전트를 구축하는 완전 관리형 방법을 도입했습니다.
- Anthropic의 모델 컨텍스트 프로토콜(MCP)은 대규모 언어 모델이 외부 도구, 환경 및 메모리에 액세스하고 상호 작용할 수 있는 방법을 정의했습니다. 이는 상황별, 지속적 및 자율 동작의 핵심입니다.

이 두 마일스톤은 기관과 인텔리전스의 합성을 나타냅니다. 에이전트는 더 이상 정적 워크플로 또는 엄격한 자동화로 제한되지 않았습니다. 이제 여러 단계에서 추론하고, 도구 및 APIs와 조정하고, 컨텍스트 상태를 유지하고, 시간이 지남에 따라 학습하고 적응할 수 있습니다.

2025년 1월~6월 - 엔터프라이즈 기능 확장

2025년 상반기에 에이전트 AI 환경은 새로운 엔터프라이즈 기능으로 크게 확장되었습니다. 2025년 2월 Anthropic은 시장 최초의 하이브리드 추론 모델인 Claude 3.7 Sonnet을 출시했으며, MCP 사양은 널리 채택되었습니다.

Amazon Q Developer, Cursor 및 WindSurf와 같은 AI 코딩 어시스턴트는 코드 생성, 리포지토리 분석 및 개발 워크플로를 표준화하기 위해 MCP를 통합했습니다. MCP 2025년 3월 릴리스에는 OAuth 2.1 보안 통합, 다양한 데이터 액세스를 위한 확장된 리소스 유형, 스트리밍 가능한 HTTP를 통한 향상된 연결 옵션 등 중요한 엔터프라이즈 지원 기능이 도입되었습니다. 이 기반을 기반으로 AWS 2025년 5월에 MCP 운영 위원회에 가입하여 새로운 agent-to-agent 통신 기능에 기여한다고 발표했습니다. 이렇게 하면 에이전트 AI 상호 운용성을 위한 업계 표준으로서 프로토콜의 입지가 더욱 강화됩니다.

2025년 5월에 Strands Agents 프레임워크를 오픈 소싱하여 에이전트 AI 워크플로를 구축하기 위한 고객 옵션을 AWS 강화했습니다. https://strandsagents.com 이 공급자와 무관하고 모델에 구애받지 않는 프레임워크를 통해 개발자는 심층적인 AWS 서비스 통합을 유지하면서 플랫폼 전체에서 파운데이션 모델을 사용할 수 있습니다. AWS 오픈 소스 블로그에서 강조 표시된 것처럼 Strands Agents는 파운데이션 모델을 에이전트 인텔리전스의 핵심에 배치하는 모델 우선 설계 철학을 따릅니다. 이를 통해 고객은 특정 사용 사례에 맞게 정교한 AI 에이전트를 더 쉽게 구축하고 배포할 수 있습니다.

출현 - 에이전트 AI

초기 자율성 아이디어에서 최신 LLM 지원 오케스트레이션에 이르기까지 소프트웨어 에이전트의 진화는 오래되고 계층화되어 왔습니다. 프로그램 인식에 대한 올리버 셀프리지의 비전으로 시작된 것은 협

업, 적응 및 추론이 가능한 지능형, 컨텍스트 인식, 목표 기반 소프트웨어 에이전트로 구성된 강력한 에코시스템으로 성장했습니다.

분산형 인공 지능(DAI)과 변환기 기반 생성형 AI의 수렴은 소프트웨어 에이전트가 더 이상 도구일 뿐만 아니라 지능형 시스템의 자율 액터인 새로운 시대의 시작을 나타냅니다.

에이전트 AI는 소프트웨어 시스템의 차세대 진화를 나타냅니다. 자율적이고 비동기적이며 에이전트적인 지능형 에이전트 클래스를 제공하며, 위임된 의도로 행동하고 동적 분산 환경에서 의도적으로 운영할 수 있습니다. 에이전트 AI는 다음을 통합합니다.

- 다중 에이전트 시스템의 아키텍처 계보 및 액터 모델
- 인지, 이유, 행동의 인지 모델
- LLMs 및 변환기의 생성 능력
- 클라우드 네이티브 및 서버리스 컴퓨팅의 운영 유연성

에이전트 AI에 대한 소프트웨어 에이전트

소프트웨어 에이전트는 환경을 인식하고, 목표에 대한 이유를 파악하고, 그에 따라 행동하도록 설계된 자율 디지털 엔터티입니다. 고정 로직을 따르는 기존 소프트웨어 프로그램과 달리 에이전트는 컨텍스트 입력 및 의사 결정 프레임워크를 기반으로 행동을 조정합니다. 따라서 클라우드 네이티브 시스템, 로봇 공학, 지능형 자동화, 현재 생성형 AI 오케스트레이션과 같은 동적 분산 환경에 이상적입니다.

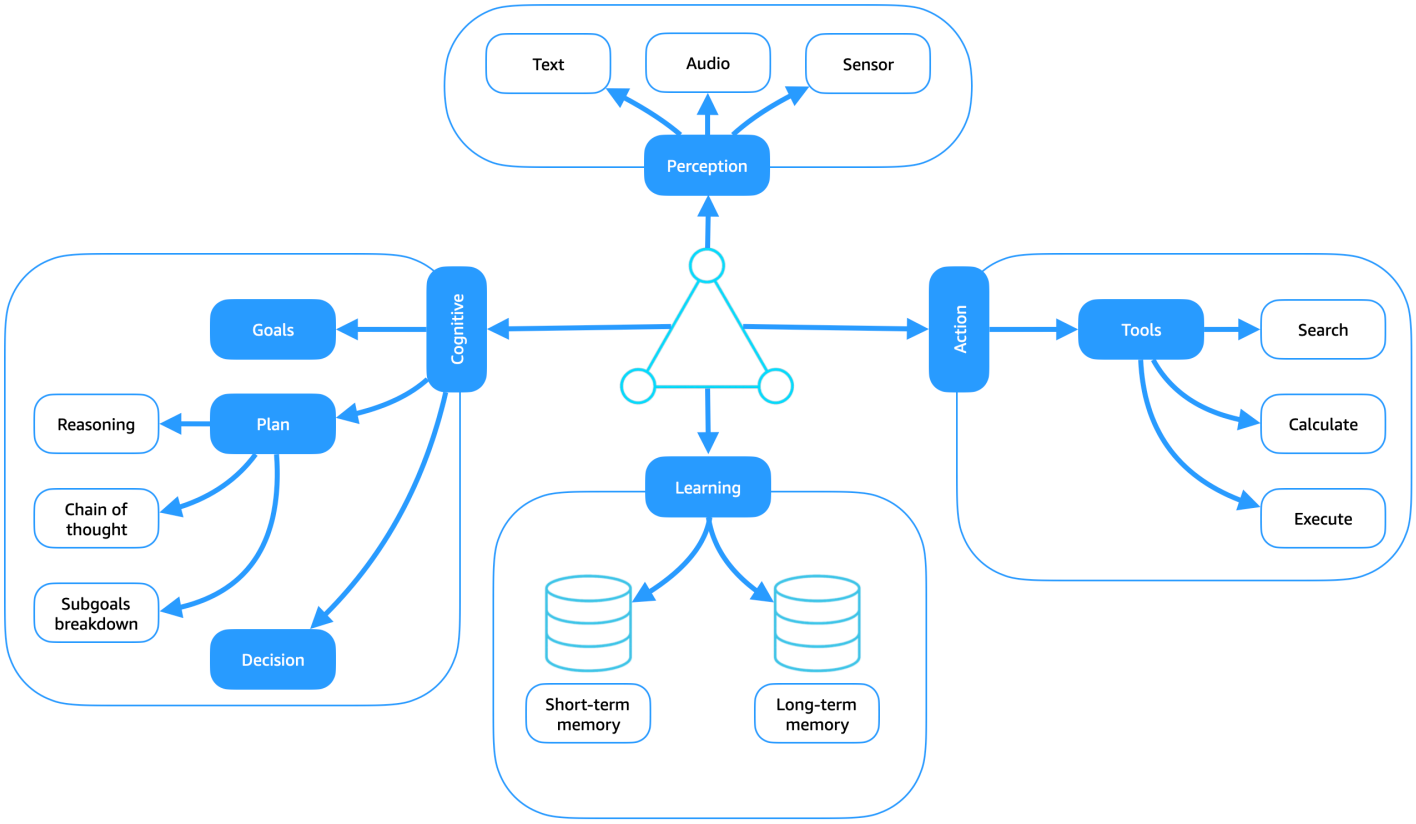
이 섹션에서는 소프트웨어 에이전트의 핵심 구성 요소를 소개하고 인식, 근거, 행동 모델을 기반으로 이러한 구성 요소가 기존 아키텍처 내에서 상호 작용하는 방법을 설명합니다. 생성형 AI, 특히 대규모 언어 모델(LLMs)이 소프트웨어 에이전트의 추론과 계획 방식을 어떻게 변화시켰는지 설명합니다. 이는 규칙 기반 시스템에서 데이터 기반 학습된 에이전트 AI 인텔리전스로 근본적으로 전환된다는 의미입니다.

이 섹션의 내용

- 소프트웨어 에이전트의 핵심 구성 요소
- 기존 에이전트 아키텍처: 인지, 이유, 행동
- 생성형 AI 에이전트: 심볼 로직LLMs으로 대체
- 기존 AI와 소프트웨어 에이전트 및 에이전트 AI 비교

소프트웨어 에이전트의 핵심 구성 요소

다음 다이어그램은 대부분의 지능형 에이전트에서 찾을 수 있는 주요 기능 모듈을 보여줍니다. 각 구성 요소는 복잡한 환경에서 에이전트가 자율적으로 운영할 수 있는 능력에 기여합니다.



인식, 이유, 행동 루프의 맥락에서 에이전트의 추론 기능은 인지 및 학습 모듈 모두에 분산됩니다. 메모리와 학습의 통합을 통해 에이전트는 과거 경험을 기반으로 적응형 추론을 개발합니다. 에이전트는 환경 내에서 행동할 때 새로운 피드백 루프를 생성합니다. 각 행동은 미래 지각에 영향을 미치며, 그 결과로 얻은 경험은 학습 모듈을 통해 메모리 및 내부 모델에 통합됩니다. 이러한 지속적인 지각, 추론 및 행동 루프를 통해 에이전트는 시간이 지남에 따라 개선되고 전체 지각, 이유, 행동 주기를 완료할 수 있습니다.

인식 모듈

지각 모듈을 사용하면 에이전트가 텍스트, 오디오 및 센서와 같은 다양한 입력 양식을 통해 환경과 인터페이스할 수 있습니다. 이러한 입력은 모든 추론 및 작업이 기반으로 하는 원시 데이터를 형성합니다. 텍스트 입력에는 자연어 프롬프트, 구조화된 명령 또는 문서가 포함될 수 있습니다. 오디오 입력에는 음성 지침 또는 환경 사운드가 포함됩니다. 센서 입력에는 시각적 피드, 모션 신호 또는 GPS 좌표와 같은 물리적 데이터가 포함됩니다. 지각의 핵심 기능은이 원시 데이터에서 의미 있는 특징과 표현을 추출하는 것입니다. 이를 통해 에이전트는 현재 컨텍스트를 정확하고 실행 가능하게 이해할 수 있습니다. 이 프로세스에는 특성 추출, 객체 또는 이벤트 인식, 의미 해석이 포함될 수 있으며 지각, 이유, 행동 루프의 중요한 첫 번째 단계를 구성합니다. 효과적인 지각은 다운스트림 추론 및 의사 결정이 관련성 있는 up-to-date 상황 인식을 기반으로 하도록 합니다.

인지 모듈

인지 모듈은 소프트웨어 에이전트의 의도적인 코어 역할을 합니다. 이는 지각을 해석하고, 의도를 형성하고, 목표 기반 계획 및 의사 결정을 통해 목적 있는 행동을 안내하는 역할을 합니다. 이 모듈은 입력을 구조화된 추론 프로세스로 변환하므로 에이전트가 사후 대응이 아닌 의도적으로 작동할 수 있습니다. 이러한 프로세스는 목표, 계획 및 의사 결정이라는 세 가지 주요 하위 모듈을 통해 관리됩니다.

목표 하위 모듈

목표 하위 모듈은 에이전트의 의도와 방향을 정의합니다. 목표는 명시적(예: "위치로 이동" 또는 "보고서 제출")이거나 암시적(예: "사용자 참여 극대화" 또는 "지연 시간 최소화")일 수 있습니다. 에이전트의 추론 주기의 핵심이며 계획 및 결정을 위한 대상 상태를 제공합니다.

에이전트는 목표에 대한 진행 상황을 지속적으로 평가하고 새로운 지각 또는 학습을 기반으로 목표의 우선순위를 다시 지정하거나 다시 생성할 수 있습니다. 이러한 목표 인식은 동적 환경에서 에이전트의 적응성을 유지합니다.

하위 모듈 계획

계획 하위 모듈은 에이전트의 현재 목표를 달성하기 위한 전략을 구성합니다. 작업 시퀀스를 생성하고, 작업을 계층적으로 분해하고, 사전 정의되거나 동적으로 생성된 계획 중에서 선택합니다.

비결정적 또는 변화하는 환경에서 효과적으로 운영하기 위해 계획은 정적이지 않습니다. 현대 에이전트는 chain-of-thought 시퀀스를 생성하고, 하위 목표를 중간 단계로 도입하고, 조건이 변경될 때 실시간으로 계획을 수정할 수 있습니다.

이 하위 모듈은 메모리 및 학습과 밀접하게 연결되며 에이전트는 과거 결과를 기반으로 시간 경과에 따른 계획을 세분화할 수 있습니다.

의사 결정 하위 모듈

의사 결정 하위 모듈은 사용 가능한 계획과 작업을 평가하여 가장 적합한 다음 단계를 선택합니다. 지각, 현재 계획, 에이전트의 목표 및 환경 컨텍스트의 입력을 통합합니다.

의사 결정 계정:

- 충돌하는 목표 간의 절충
- 신뢰도 임계값(예: 지각의 불확실성)
- 작업의 결과
- 에이전트의 학습된 경험

아키텍처에 따라 에이전트는 심볼 추론, 휴리스틱, 강화 학습 또는 언어 모델(LLMs)에 의존하여 정보에 입각한 결정을 내릴 수 있습니다. 이 프로세스는 에이전트의 동작 컨텍스트 인식, 목표 정렬 및 적응을 유지합니다.

작업 모듈

작업 모듈은 에이전트가 선택한 결정을 실행하고 외부 세계 또는 내부 시스템과 상호 작용하여 의미 있는 효과를 생성하는 역할을 합니다. 의도가 행동으로 변환되는 지각, 이유, 행동 루프의 행동 단계를 나타냅니다.

인지 모듈이 작업을 선택하면 작업 모듈은 특수 하위 모듈을 통해 실행을 조정합니다. 여기서 각 하위 모듈은 에이전트의 통합 환경에 맞게 조정됩니다.

- 물리적 작동: 로봇 시스템 또는 IoT 디바이스에 포함된 에이전트의 경우이 하위 모듈은 결정을 실제 물리적 이동 또는 하드웨어 수준 지침으로 변환합니다.

  예: 로봇 조향, 밸브 트리거, 센서 켜기.
- 통합 상호 작용:이 하위 모듈은 소프트웨어 시스템, 플랫폼 또는 APIs와의 상호 작용과 같은 비물리적이지만 외부에서 볼 수 있는 작업을 처리합니다.

  예: 클라우드 서비스에 명령 전송, 데이터베이스 업데이트, API를 호출하여 보고서 제출.
- 도구 호출: 에이전트는 특수 도구를 사용하여 다음과 같은 하위 작업을 수행하여 기능을 확장하는 경우가 많습니다.
  - 검색: 정형 또는 비정형 지식 소스 쿼리
  - 요약: 대규모 텍스트 입력을 상위 수준 개요로 압축
  - 계산: 논리적, 수치적 또는 심볼 계산 수행

  도구 호출은 모듈식의 호출 가능한 기술을 통해 복잡한 동작 구성을 가능하게 합니다.

학습 모듈

학습 모듈을 통해 에이전트는 경험을 기반으로 시간이 지남에 따라 적응, 일반화 및 개선할 수 있습니다. 지각 및 행동의 피드백을 사용하여 에이전트의 내부 모델, 전략 및 결정 정책을 지속적으로 개선하여 추론 프로세스를 지원합니다.

이 모듈은 단기 및 장기 메모리와 함께 작동합니다.

- 단기 메모리: 대화 상태, 현재 작업 정보 및 최근 관찰과 같은 일시적인 컨텍스트를 저장합니다. 에이전트가 상호 작용 및 작업 내에서 연속성을 유지하는 데 도움이 됩니다.
- 장기 메모리: 이전에 발견한 목표, 행동 결과 및 환경 상태를 포함하여 과거 경험의 지속적인 지식을 인코딩합니다. 장기 메모리를 사용하면 에이전트가 패턴을 인식하고, 전략을 재사용하고, 실수가 반복되지 않도록 할 수 있습니다.

학습 모드

학습 모듈은 다양한 환경과 에이전트 역할을 지원하는 지도, 비지도 및 강화 학습과 같은 다양한 패러다임을 지원합니다.

- 지도 학습: 레이블이 지정된 예제를 기반으로 내부 모델을 업데이트합니다. 종종 인적 피드백 또는 훈련 데이터 세트에서 업데이트합니다.

  예: 이전 대화를 기반으로 사용자 의도를 분류하는 방법을 알아봅니다.
- 비지도 학습: 명시적 레이블 없이 데이터에서 숨겨진 패턴 또는 구조를 식별합니다.

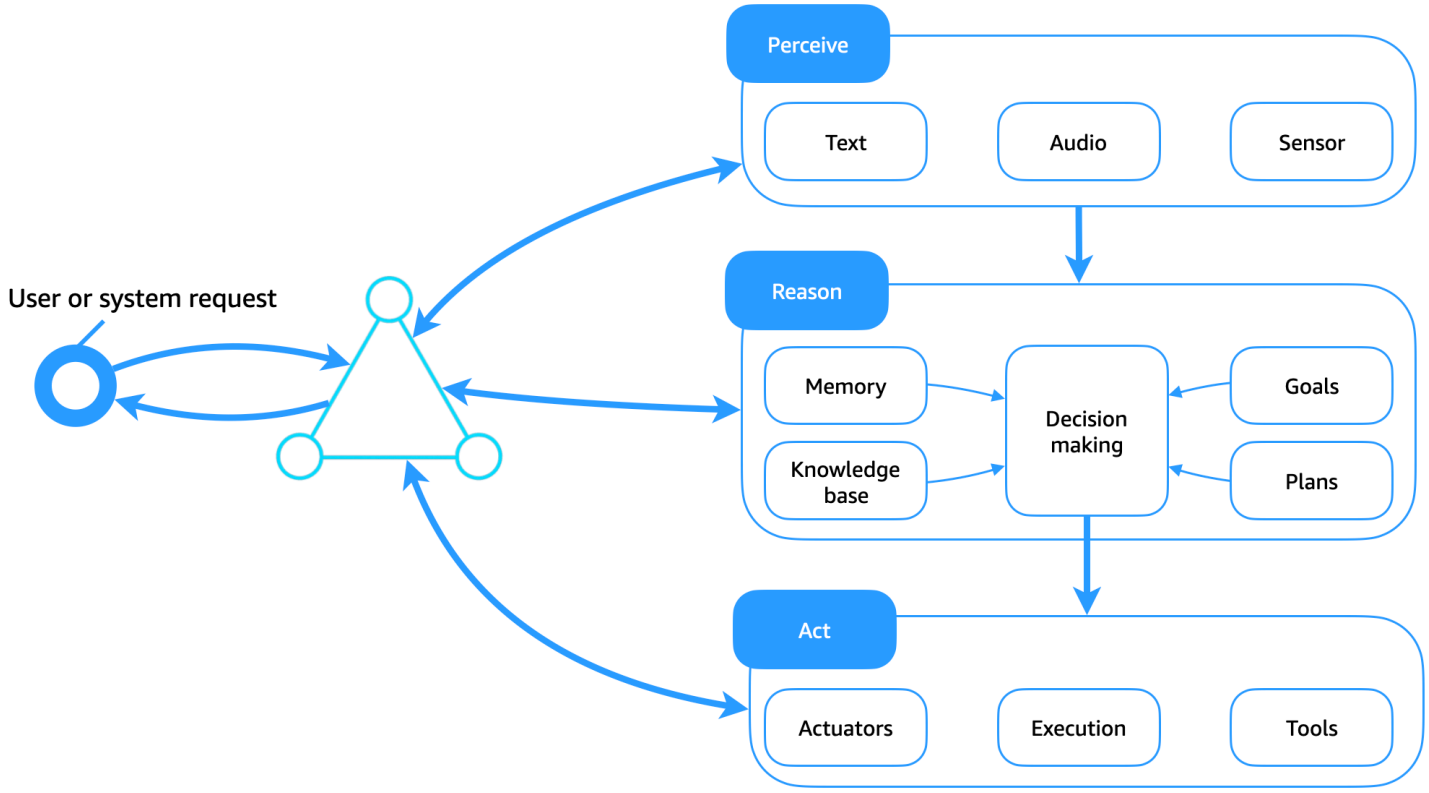
  예: 환경 신호를 클러스터링하여 이상을 감지합니다.
- 강화 학습: 대화형 환경에서 누적 보상을 극대화하여 시도 및 오류를 통해 동작을 최적화합니다.

  예: 어떤 전략이 가장 빠른 작업 완료로 이어지는지 학습합니다.

학습은 에이전트의 인지 모듈과 긴밀하게 통합됩니다. 과거 결과를 기반으로 계획 전략을 구체화하고, 과거 성공 평가를 통해 의사 결정을 개선하고, 지각과 행동 간의 매핑을 지속적으로 개선합니다. 이 닫힌 학습 및 피드백 루프를 통해 에이전트는 사후 대응 실행을 넘어 시간이 지남에 따라 새로운 목표, 조건 및 컨텍스트에 적응할 수 있는 자체 개선 시스템으로 발전합니다.

기존 에이전트 아키텍처: 인지, 이유, 행동

다음 다이어그램은 이전 섹션에서 설명한 구성 요소가 인식, 이유, 행동 주기에 따라 작동하는 방식을 보여줍니다.



인식 모듈

지각 모듈은 에이전트와 외부 세계의 감각 인터페이스 역할을 합니다. 원시 환경 입력을 추론에 정보를 제공하는 구조화된 표현으로 변환합니다. 여기에는 텍스트, 오디오 또는 센서 신호와 같은 멀티모달 데이터 처리가 포함됩니다.

- 텍스트 입력은 사용자 명령, 문서 또는 대화에서 비롯될 수 있습니다.
- 오디오 입력에는 음성 지침 또는 환경 사운드가 포함됩니다.
- 센서 입력은 모션, 시각적 피드 또는 GPS와 같은 실제 신호를 캡처합니다.

원시 입력이 수집되면 지각 프로세스가 특성 추출을 수행한 다음 객체 또는 이벤트 인식 및 의미 해석을 수행하여 현재 상황에 대한 의미 있는 모델을 생성합니다. 이러한 출력은 다운스트림 의사 결정을 위한 구조화된 컨텍스트를 제공하고 에이전트의 추론을 실제 관측치에 고정합니다.

이유 모듈

이유 모듈은 에이전트의 인지 코어입니다. 컨텍스트를 평가하고, 의도를 공식화하고, 적절한 조치를 결정합니다. 이 모듈은 학습한 지식과 추론을 모두 사용하여 목표 기반 행동을 오케스트레이션합니다.

이유 모듈은 긴밀하게 통합된 하위 모듈로 구성됩니다.

- 메모리: 대화 상태, 작업 컨텍스트 및 간헐적 기록을 단기 및 장기 형식으로 유지합니다.
- 지식 기반: 심볼 규칙, 온톨로지 또는 학습된 모델(예: 임베딩, 사실 및 정책)에 대한 액세스를 제공합니다.
- 목표 및 계획: 원하는 결과를 정의하고 이를 달성하기 위한 행동 전략을 수립합니다. 목표는 동적으로 업데이트할 수 있으며 계획은 피드백에 따라 조정적으로 수정할 수 있습니다.
- 의사 결정: 옵션을 평가하고, 장단점을 평가하고, 다음 작업을 선택하여 중앙 중재 엔진 역할을 합니다. 이 하위 모듈은 신뢰도 임계값, 목표 정렬 및 컨텍스트 제약 조건을 고려합니다.

이러한 구성 요소를 함께 사용하면 에이전트가 환경을 추론하고, 신념을 업데이트하고, 경로를 선택하고, 일관된 적응형 방식으로 행동할 수 있습니다. 이유 모듈은 지각과 행동 간의 격차를 해소합니다.

Act 모듈

작업 모듈은 디지털 또는 물리적 환경과 상호 작용하여 에이전트가 선택한 결정을 실행하여 작업을 수행합니다. 여기에서 의도가 조치가 됩니다.

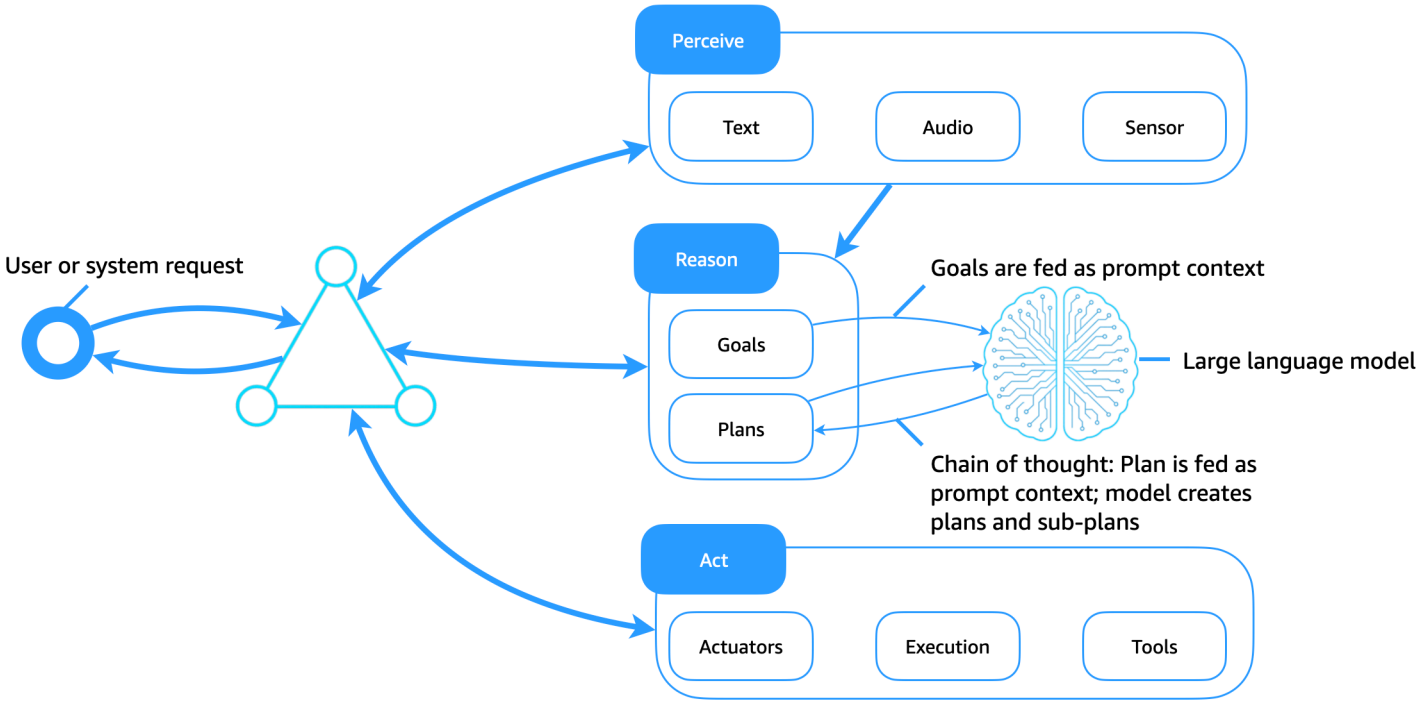
이 모듈에는 세 가지 기능 채널이 포함되어 있습니다.

- 액추에이터: 물리적으로 존재하는 에이전트(예: 로봇 및 IoT 디바이스)의 경우는 이동, 조작 또는 신호와 같은 하드웨어 수준 상호 작용을 제어합니다.
- 실행: APIs 호출, 명령 디스패치, 시스템 업데이트 등 소프트웨어 기반 작업을 처리합니다.
- 도구: 검색, 요약, 코드 실행, 계산 및 문서 처리와 같은 기능 기능을 활성화합니다. 이러한 도구는 동적이고 컨텍스트를 인식하여 에이전트의 유틸리티를 확장하는 경우가 많습니다.

작업 모듈의 출력은 환경으로 다시 공급되고 루프를 닫습니다. 이러한 결과는 에이전트가 다시 인식합니다. 에이전트의 내부 상태를 업데이트하고 향후 결정을 알려 지각, 이유, 행동 주기를 완료합니다.

생성형 AI 에이전트: 심볼 로직LLMs으로 대체

다음 다이어그램은 대규모 언어 모델(LLMs 이제 소프트웨어 에이전트를 위한 유연하고 지능적인 인지 코어 역할을 하는 방법을 보여줍니다. 정적 계획 라이브러리 및 수동 코딩 규칙에 의존하는 기존 심볼로직 시스템과 달리 LLMs 적응형 추론, 컨텍스트 계획 및 동적 도구 사용을 지원하여 에이전트가 인식, 추론 및 행동하는 방식을 변화시킵니다.



주요 개선 사항

이 아키텍처는 다음과 같이 기존 에이전트 아키텍처를 개선합니다.

- 인지 엔진으로서LLMs: 목표, 계획 및 쿼리는 프롬프트 컨텍스트로 모델에 전달됩니다. LLM은 추론 경로(예: 사고 체인)를 생성하고, 작업을 하위 목표로 분해하고, 다음 작업을 결정합니다.
- 프롬프트를 통한 도구 사용: 도구 사용 에이전트 또는 APIs 및 행동(ReAct) 프롬프트를 통해 LLMs을 전달할 수 있습니다.
- 컨텍스트 인식 계획: 에이전트는 하드 코딩된 계획 라이브러리 없이 에이전트의 현재 목표, 입력 환경 및 피드백을 기반으로 계획을 동적으로 생성하거나 수정합니다.
- 프롬프트 컨텍스트를 메모리로: 에이전트는 심볼 지식 기반을 사용하는 대신 메모리, 계획 및 목표를 모델에 전달되는 프롬프트 토큰으로 인코딩합니다.
- 단 몇 번의 컨텍스트 내 학습을 통한 학습: LLMs 프롬프트 엔지니어링을 통해 동작을 조정하므로 명시적 재훈련 또는 엄격한 계획 라이브러리의 필요성이 줄어듭니다.

LLM 기반 에이전트에서 장기 메모리 확보

구조화된 지식 기반에 장기 메모리를 저장한 기존 에이전트와 달리 생성형 AI 에이전트는 LLMs의 컨텍스트 기간 제한 내에서 작동해야 합니다. 메모리를 확장하고 지속적인 인텔리전스를 지원하기 위해 생

성형 AI 에이전트는 에이전트 스토어, 검색 증강 생성(RAG), 컨텍스트 내 학습 및 프롬프트 체인, 사전 훈련 등 여러 보완 기법을 사용합니다.

에이전트 스토어: 외부 장기 메모리

에이전트 상태, 사용자 기록, 결정 및 결과는 장기 에이전트 메모리 스토어(예: 벡터 데이터베이스, 객체 스토어 또는 문서 스토어)에 저장됩니다. 관련 메모리는 온디맨드 방식으로 검색되어 런타임 시 LLM 프롬프트 컨텍스트에 주입됩니다. 이렇게 하면 에이전트가 세션, 작업 또는 상호 작용 간에 연속성을 유지하는 영구 메모리 루프가 생성됩니다.

RAG

RAG는 검색된 지식과 생성형 기능을 결합하여 LLM 성능을 개선합니다. 목표 또는 쿼리가 발행되면 에이전트는 검색 인덱스를 검색합니다(예: 문서 의미 체계 검색, 이전 대화 또는 구조화된 지식을 통해). 검색된 결과는 LLM 프롬프트에 추가되며, 이는 외부 사실 또는 개인화된 컨텍스트에서 생성을 기반으로 합니다. 이 방법은 에이전트의 유효 메모리를 확장하고 신뢰성과 사실적 정확성을 개선합니다.

컨텍스트 내 학습 및 프롬프트 체인

에이전트는 세션 내 토큰 컨텍스트와 구조화된 프롬프트 체인을 사용하여 단기 메모리를 유지합니다. 현재 계획, 이전 작업 결과, 에이전트 상태와 같은 컨텍스트 요소는 행동 안내를 위한 호출 간에 전달됩니다.

지속적인 사전 훈련 및 미세 조정

도메인별 에이전트의 경우 로그, LLMs을 계속 사전 훈련할 수 있습니다. 또는 인적 피드백(RLHF)을 통한 지침 미세 조정 또는 강화 학습을 통해 에이전트와 유사한 동작을 모델에 직접 포함할 수 있습니다. 이렇게 하면 추론 패턴이 프롬프트 시간 로직에서 모델의 내부 표현으로 전환되고 프롬프트 길이가 줄어들며 효율성이 향상됩니다.

에이전트 AI의 결합된 이점

이러한 기술을 함께 사용하면 생성형 AI 에이전트가 다음을 수행할 수 있습니다.

- 시간이 지남에 따라 상황 인식을 유지합니다.
- 사용자 기록 또는 기본 설정에 따라 동작을 조정합니다.
- up-to-date, 사실적 또는 비공개 지식을 사용하여 결정을 내립니다.
- 지속적이고 규정을 준수하며 설명 가능한 행동을 통해 엔터프라이즈 사용 사례로 확장합니다.

외부 메모리, 검색 계층 및 지속적인 훈련으로 LLMs을 보강하면 에이전트는 심볼 시스템만으로는 이전에는 달성할 수 없었던 수준의 인지 연속성과 목적을 달성할 수 있습니다.

기존 AI와 소프트웨어 에이전트 및 에이전트 AI 비교

다음 표에서는 기존 AI, 소프트웨어 에이전트 및 에이전트 AI를 자세히 비교합니다.

기능	기존 AI	소프트웨어 에이전트	에이전트 AI
예제	스팸 필터, 이미지 분류기, 추천 엔진	챗봇, 작업 스케줄러, 에이전트 모니터링	AI 어시스턴트, 자율 개발자 에이전트, 다중 에이전트 LLM 오케스트레이션
실행 모델	배치 또는 동기	이벤트 기반 또는 예약	비동기식, 이벤트 기반 및 목표 기반
자율성	제한적입니다. 종종 인적 또는 외부 오케스트레이션이 필요합니다.	중간, 사전 정의된 경계 내에서 독립적으로 작동	높음, 적응형 전략과 독립적으로 작동
반응성	입력 데이터에 대한 대응	환경 및 이벤트에 대응	대응적이고 선제적입니다. 조치를 예상하고 시작합니다.
사전 대응	드물게	일부 시스템에 있음	핵심 속성, 목표 지향 동작 주도
통신	미니멀, 일반적으로 독립 실행형 또는 API 바인딩	에이전트 간 또는 에이전트-인간 메시징	풍부한 다중 에이전트 및 human-in-the-loop 상호 작용
의사 결정	모델 추론 전용(분류, 예측 등)	심볼 추론 또는 규칙 기반 또는 스크립팅된 결정	컨텍스트, 목표 기반, 동적 추론(대개 LLM 강화)
위임된 의도	아니요. 사용자가 직접 정의한 작업을 수행합니다.	부분적, 범위가 제한된 사용자 또는 시스템을 대신합니다.	예. 종종 서비스, 사용자 또는 시스템 전반에

기능	기존 AI	소프트웨어 에이전트	에이전트 AI
			서 위임된 목표를 가지고 행동합니다.
학습 및 적응	종종 모델 중심(예: ML 훈련)	때때로 적응형	임베디드 학습, 메모리 또는 추론(예: 피드백, 자체 수정)
기관	없음, 인간을 위한 도구	암시적 또는 기본적	명시적, 목적, 목표 및 자기 주도적으로 운영됨
컨텍스트 인식	낮음, 상태 비저장 또는 스냅샷 기반	보통, 일부 상태 추적	높음, 메모리, 상황 컨텍스트 및 환경 모델 사용
인프라 역할	앱 또는 분석 파이프라인에 포함됨	미들웨어 또는 서비스 계층 구성 요소	클라우드, 서버리스 또는 엣지 시스템과 통합된 구성 가능한 에이전트 메시

요약하면 다음과 같습니다.

- 기존 AI는 도구 중심이며 기능적으로 좁습니다. 예측 또는 분류에 중점을 둡니다.
- 기존 소프트웨어 에이전트는 자율성과 기본 통신을 도입하지만 규칙 바인딩 또는 정적인 경우가 많습니다.
- 에이전트 AI는 자율성, 비동기성 및 기관을 결합합니다. 복잡한 시스템 내에서 추론, 행동 및 적응할 수 있는 지능형 목표 기반 엔터티를 지원합니다. 따라서 에이전트 AI는 클라우드 네이티브 AI 기반 미래에 이상적입니다.

다음 단계

이 가이드에서는 기존 소프트웨어 에이전트가 생성형 AI로 구동되는 자율적이고 지능적인 시스템으로 진화하는 것을 나타내는 에이전트 AI의 역사와 기초에 대해 설명했습니다. 초기 소프트웨어 에이전트가 사전 정의된 규칙과 로직을 따라 고정된 경계 내에서 작업을 자동화하는 방법을 설명하고 에이전트가 개방형 환경에서 동적으로 추론, 학습 및 적응할 수 있는 대규모 언어 모델을 통합하여 에이전트 AI 가이 기반을 구축하는 방법을 설명했습니다.

이 시리즈의 다음 간행물을 검토하여 에이전트 AI를 자세히 살펴볼 수 있습니다.

- 에서 에이전트 AI를 운영하면 AWS 에이전트 AI를 격리된 실험에서 엔터프라이즈 규모의 가치 창출 인프라로 변환할 수 있는 조직 전략을 제공합니다.
- 의 에이전트 AI 패턴 및 워크플로 AWS에서는 목표 지향 AI 에이전트를 설계, 구성 및 오케스트레이션하는 데 사용되는 기본 청사진과 모듈식 구문에 대해 설명합니다.
- 의 에이전트 AI 프레임워크, 프로토콜 및 도구는 AWS 에이전트 AI 솔루션을 구축할 때 고려해야 할 소프트웨어 기반, 툴킷 및 프로토콜을 다룹니다.
- 에서 에이전트 AI용 서버리스 아키텍처를 구축 AWS하면 최신 AI 워크로드의 자연스러운 기반으로서 서버리스 아키텍처에 대해 설명하고에서 AI 네이티브 서버리스 아키텍처를 구축하는 방법을 설명합니다 AWS 클라우드.
- 에서 에이전트 AI를 위한 다중 테넌트 아키텍처를 구축 AWS하면 호스팅 고려 사항, 배포 모델 및 컨트롤 플레인을 포함한 다중 테넌트 설정에서 AI 에이전트를 사용하는 방법을 설명합니다.

리소스

이 가이드에서 설명하는 개념에 대한 자세한 내용은 다음 가이드 및 문서를 참조하세요.

AWS 참조

- Amazon Bedrock Agents
- Amazon Q Developer
- Strands 에이전트 SDK

기타 참조

- Hewitt, Carl, Peter Bishop, Richard Steiger. “인공 지능을 위한 범용 모듈식 ACTOR 공식화.” 제3차 인공 지능 공동 컨퍼런스(1973) 논문: 235-245. https://www.ijcai.org/Proceedings/73/Papers/027B.pdf
- Lesser, Victor R., 관련 간행물(전체 목록 참조):
  - Lesser, Victor R. 및 Daniel D. Corkill. "함수적으로 정확하고 협력적인 분산 시스템." Systems, Man 및 Cybernetics 11, no. 1(1981): 81-96. https://ieeexplore.ieee.org/abstract/document/4308581
  - Decker, Keith S. 및 Victor R. Lesser. “협정 서비스에서의 커뮤니케이션.” AAAI Workshop on Planning for Interagent Communication (1994). https://www.researchgate.net/profile/Victor-Lesser/publication/2768884_Communication_in_the_Service_of_Coordination/links/00b7d51cc2a0750cb4000000/Communication-in-the-Service-of-Coordination.pdf
  - Durfee, Edmund H., Victor R. Lesser 및 Daniel D. Corkill. "협동 분산 문제 해결의 추세" IEEE Transactions on knowledge and data Engineering(1989). http://mas.cs.umass.edu/Documents/ieee-tkde89.pdf
  - Durfee, Edmund H., V.R. Lesser 및 D.D. Corkill, “분산 인공 지능.” 분산 문제 해결 네트워크의 통신을 통한 협력(1987): 29-58. https://www.academia.edu/download/79885643/durf94_1.pdf
  - Lâasri, Brigitte, Hassan Lâasri, Susan Lander 및 Victor Lesser. “지능형 협상 에이전트를 위한 일반 모델.” International Journal of Cooperative Information Systems 01, no. 02(1992): 291-317. https://doi.org/10.1142/S0218215792000210
  - Lander, Susan E. 및 Victor R. Lesser. “이종 에이전트 간의 분산 검색에서 협상의 역할 이해.” IJCAI'93: 인공 지능에 대한 제13차 국제 공동 회의(1993)의 절차: 438-444. https://www.ijcai.org/Proceedings/93-1/Papers/062.pdf

  - Lander, Susan, Victor R. Lesser 및 Margaret E. Connell. “전문 에이전트 협력을 위한 충돌 해결 전략” CKBS'90: 지식 기반 시스템 협력에 대한 국제 실무 회의(1990년 10월): 183-200. https://doi.org/10.1007/978-1-4471-1831-2_10
  - Prasad, M.V. Nagendra, Victor Lesser 및 Susan E. Lander. “이종 다중 에이전트 시스템에서 실험 학습.” 다중 에이전트 시스템의 적응 및 학습에 대한 IJCAI-95 워크숍(1995): 59-64. https://www.researchgate.net/publication/2784280_Learning_Experiments_in_a_Heterogeneous_Multi-agent_System
- Nwana, Hyacinth S. "소프트웨어 에이전트: 개요" 지식 엔지니어링 검토 11, 아니요. 3(1996년 10월/11월): 205-244. https://teaching.shu.ac.uk/aces/rh1/elearning/multiagents/introduction/nwana.pdf
- Selfridge, Oliver G. "판데모니움: 학습의 패러다임" 사고 프로세스의 메커니즘: 국립 물리 연구소 1(1959년)에서 개최된 심포지엄의 절차: 511~529. https://aitopics.org/download/classics:504E1BAC
- Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N.™, Lukasz Kaiser 및 Illia Meshsukhin. “주의는 필요한 것뿐입니다.” 신경 정보 처리 시스템(NIPS)에 대한 31차 컨퍼런스의 절차. 신경 정보 처리 시스템 30의 발전(2017): 5998-6008. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
- ™dridge, Michael 및 Nicholas R. Jennings. "지능형 에이전트: 이론 및 실습" 지식 엔지니어링 검토 10, 아니요. 2(1995년 1월): 115-152. https://www.cs.cmu.edu/~motionplanning/papers/sbp_papers/integrated1/woodridge_intelligent_agents.pdf

문서 기록

아래 표에 이 가이드의 주요 변경 사항이 설명되어 있습니다. 향후 업데이트에 대한 알림을 받으려면 RSS 피드를 구독하십시오.

변경 사항	설명	날짜
최초 게시	—	2025년 7월 14일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.