DeepL
Subscribe to DeepL Pro to translate larger documents.
Visit www.DeepL.com/pro for more information.

소프트웨어 엔지니어링이 고급 AI의 이점을 활용할 수 있을까요?

메리 쇼 및 리밍 주

인공 지능(AI)을 통해 우리는 이전에는 불가능하다고 여겨졌던 것 이상의 시스템을 구축할 수 있게 되었습니다. AI 지원 시스템의 요구 사항을 충족하기 위해 소프트웨어 엔지니어링의 기존 기술을 발전시킬 수 있을까요, 아니면 이를 위해 독특하고 새

포인트: 엔지니어링을 머신 러닝에 도입하기

우리는 대규모 데이터 세트에서 추론을 도출하는 머신러닝(ML) 알고리즘을 통합한 시스템이 주도하는 '인공지능(AI) 혁명'의 한가운데에 있다고 **합니다.** 어떤 사람들은 ML 시스템이 기존 소프트웨어 시스템과 크게 다르기 때문에 이에 대응하기 위해 새로운 소프트웨어 학문이 필요하다고 주장합니다.

오히려 소프트웨어 엔지니어는 기존 방법의 포트폴리오를 개선하여 새로운 유형의 소프트웨어 기회에 정기적으로 대응합니다. 수년 동안 AI의 다른 아이디어에 대해 해왔던 것처럼 ML에 대해서도 마찬가지로 그렇게 해야 합니다.

AI는 오랫동안 새로운 소프트웨어 아이디어의 원천이자 인큐베이터 역할을 해왔습니다.

디지털 객체 식별자 10.1109/MS.2022.3203200 현재 버전 날짜: 2022년 10월 24일

이러한 아이디어는 처음에는 다소 엉뚱하고 비현실적으로 보일 수 있지만, 결국에는 그 능력을 인정받아 주류로 자리 잡게 되고, 그 과정에서 AI에 뿌리를 둔 아이디어는 거의 잊혀지게 됩니다. 소프트웨어 기술의 레퍼토리는 AI의 영향으로 더욱 풍부해졌습니다.

AI에서 주류 소프트웨어로 발전한 몇 가지 아이디어만 살펴보세요. AI는 디자인을 문제 해결로 보는 관점을 가져왔으며, 이는 종종 디자인을 문제 설정으로 보는 포괄적인 관점을 덮어버렸습니다. 목록 처리는 동적 목록 구조와 함수형 프로그래밍을 도입하면서 반세기 전에 AI 커뮤니티에서 시작되었습니다. 검색 전략은 검색이 주요 산업으로 부상하기 몇 년 전부터 AI의 주류였습니다.

AI 연구자들은 특정 용도의 프로그램을 작성하려는 시도를 지원하기 위해 탐색적 프로그래밍을 최초로 사용했습니다.

"지능"이 필요한 것으로 간주되는 작업을 수행했습니다. 전문가 시스템을 구현하는 데 사용한 프로덕션 시스템은 현대의 게시-구독 시스템의 선구자였습니다.

현재 AI에서 주류 소프트웨어 시스템에 이르기까지 어떤 솔루션이 제공되고 있을까요? ML[1,2,3]을 통합하는 시스템에 대해 자주 제기되는 우려는 무엇보다도 데이터의 지배적인 역할과 큐레이션입니다. 또한 불투명성, 사양의 부재, 정확성, 동적 변화, 비결정성, 불안정성에 뿌리를 둔 정확성 및 비결정성에 대한 우려도 있습니다. 실제 소프트웨어 컴포넌트는 항상 이러한 우려에 시달려 왔지만, 머신러닝 컴포넌트는 이를 단숨에 해결합니다. ML 컴포넌트를 통합하는 시스템에 대한 지적 제어를 제공하는 추상화를 개발해야 합니다.

AI는 정기적으로 새로운 아이디어를 제공합니다. 그리고 소프트웨어 엔지니어링에 대한 도전.

소프트웨어 엔지니어링은 이러한 아이디어를 안정적으로 실행에 옮기는 혁신적인 대응을 위해 다양한 엔지니어링 기술 레퍼토리를 정기적으로 활용해 왔습니다. 소프트웨어 엔지니어링이 ML에서도 동일한 역할을 할 것이라고 확신할 수 있습니다.

대규모 데이터 세트
ML 시스템은 자동화된
야생에서 수집한 대량의 데이터를 수집, 분류, 정리, 버전 관리 및 분석할 수 있습니다. 또한 데이터 세트는 정기적으로 업데이트되며, 그 모델은 다

템 관리가 필요합니다. 코드뿐만 아니라 데이터의 양을 늘릴 수 있습니다.

데이터 품질은 잘못 형성된 데이터부터 잘 형성되었지만 잘못된 데이터, 편향되거나 악의적인 데이터에 이르기까지 다양한 위협에 취약하며, 이는 기존 시스템에서도 오랫동안 지속되어 온 문제입니다. ML 시스템에서는 모델 개발의 탐색적 특성으로 인해 이 문제가 더욱 악화됩니다.

비결정론
ML 시스템은 전례가 없는 것으로 간주됩니다.
여러 가지 면에서 불투명하고 신뢰할 수 없습니다. 알고리즘이 불투명하고, 출력과 입력이 어떻게 연관되어 있는지 설명할 수 없습니다. 또한 재학습을 거치면서 예고 없이 동작이 변경될 수 있으며, 숨겨진 종속성과 모델 상호작용이 예측 불가능성을 더합니다. 그러나 기존의 컴포넌트도 실제로는 설명할 수 없고, 타사 컴포넌트도 예고 없이 변경되며, 기능 상호 작용은 다음과 같은 이유로 연구되어 왔습니다.

> 지적 제어를 제공하는 추상화를 개발해야 합니다.
> ML 구성 요소를 통합한 시스템보다 더 높은 성능을 제공합니다.

정기적으로 재연출(재학습)됩니다. 이러한 모델의 추론은 데이터의 품질과 학습 알고리즘의 품질에 따라 달라집니다. 그러나 일반 데이터베이스 시스템에는 코드와 별도로 사람이 관리하며 데이터 품질이 주요 쟁점이 되는 풍부하고 고도로 구조화된 데이터가 포함된 대규모 데이터 세트도 있습니다. ML 데이터 세트에 필요한 관리는 데이터베이스에 필요한 광고 관리와 다르지만, 두 경우 모두 시스

수십 년. ML 구성 요소의 명백한 비결정성은 입력이 잘못 지정된 일반 복잡한 소프트 웨어 시스템, 특히 사이버 물리 시스템에서 발견되는 비결정성과 유사합니다.

신뢰할 수 있는 소프트웨어의 원칙은 모듈화 및 아키텍처 방화벽, 이중화, 런타임 검사 및 피드백, 버전 제어, 체크포인트, 검사 가능한 어설션, 장애 심각도 우선순위 지정, 내결함성, 유예 성능 저하, 확률적 모델링 등과 같은 기술을 사용하여 비결정론적 구성 요소로부터 신뢰할 수 있는 시스템을 개발할 수 있게 해줍니다. 이러한 기능을 통해 측정 불가능하거나 신뢰할 수 없는 구성 요소의 변수를 로컬

라이즈하거나 보완하고, 중간 상태 점검을 제공하고, 허용 가능한 성능의 범위를 정의하고, 현장 가변성을 분석할 수 있으며, 실제로 ML을 사용할 수 있습니다.

시스템에서 이상 징후를 감지하는 기법입니다.[4] 이러한 기법이 ML 시스템에 그대로 적용되지는 않더라도 견고하고 잘 정립된 출발점을 제공합니다.

정확성

ML 구성 요소는 다음과 같은 용도로 사용됩니다.

입력 데이터를 생성한 현상을 모델링합니다. 이러한 현상에 대한 정확한 모델이 이미 있다면 ML을 사용할 필요가 없습니다. 이 경우 결과는 반드시 사전 사양이 아닌 다른 기준에 따라 판단될 것입니다. 그러나 이는 현실 세계의 많은 시스템, 특히 사회 기술 시스템에서도 마찬가지입니다. 이러한 시스템에는 약하거나 부정확하거나 불완전한 사양이 있습니다.

정확한 사양이 없는 경우, ML 시스템은 기대에 부합하는 결과를 어느 정도 제공하는지에 따라 평가될 수 있습니다. 그러나 업무 적합성이 평가의 기준으로 받아들여지는 일반적인 소프트웨어에서도 비슷한 상황이 발생합니다. 더 심각한 문제는 ML 시스템 개발자가 편향된 결과, 즉 데이터를 정확하게 캡처할 수 있지만 사회적 기대치와 일치하지 않는 모델에 대해 적절히 우려해야 한다는 점입니다. 하지만 이러한 상황은 반세기 전 리텔과 웨버[5]가 소개한 "사악한 문제"의 개념입니다.

적절한 추상화

소프트웨어에 ML 통합

시스템에 적합한 추상화를 개발해야 합니다. 구성 요소의 주요 유형은 무엇이며, 이를 알고리즘, 모델, 데이터 세트, 함수 또는 다른 무엇으로 생각합니까? 시스템의 어떤 아키텍처가 데이터 큐레이션, 학습 및 결과 모델 호출에 대한 지적 제어를 제공하며, 이는 소프트웨어 운영에서 서로 다른 시점에 발생합니다.

시스템? 불확실성, 비결정성, 불투명성을 적절한 수준의 디테일로 처리하기 위한 적절한 추상화 방식은 무엇일까요? 추상화 포트폴리오를 확장하는 것은 소프트웨어 엔지니어링의 핵심이므로 이는 자연스러운 진화가 될 것입니다.

따라서 저는 제4회 프로그래밍 언어의 역사 컨퍼런스 기조 강연에서 주장했듯이,[6] ML은 소프트웨어 개발의 기존 규범에 대한 오랜 도전의 역사에서 가장 최근의 것이라고 이 글에서 주장합니다. ML의 문제는 소프트웨어 분야에서 이전에 다루었던 문제와 유사합니다. ML을 기존의 소프트웨어 원칙과 프로세스를 무효화하는 인공지능의 대변혁을 알리는 새로운 도전으로 취급하기보다는 소프트웨어 엔지니어의 확립된 접근 방식을 기반으로 ML의 고유한 특성을 적용해야 합니다. ML이 다른 소프트웨어와 본질적으로 다르다는 주장은 더 이상 유효하지 않습니다.

샤피로와 바리안의 분석7에서 힌트를 얻어야 합니다.

형성 경제: 새로운 기술을 인식하고 이해하되, 새로운 매개변수가 추가된 기존 원칙이 어떻게 여전히 적용되는지 질문합니다. ML 구성 요소의 비규칙성은 기존 구성 요소보다 더 클 수 있지만, 기존 소프트 웨어용으로 개발한 기술을 확장하여 이를 처리할 수 있습니다. 마지막으로, ML은 소프트웨어 엔지니어링에 대한 마지막 도전이 아닙니다. 양자 컴퓨팅은 소프트웨어 엔지니어링 포트폴리오의 새로운 개선을 요구하기 위해 기다리고 있습니다.

모델 및 기술.

승인

카네기멜론대학교의 앨런 펄리스 컴퓨터과학 의장의 지원으로 프로그래밍 언어에 대한 성찰을 제4차 역사에 쓸 수 있었습니다.

이 에세이의 기초를 제공한 프로그래밍 언어6. 이러한 생각을 다듬는 데 도움을 준 많은 동료들에게 감사드립니다.

-메리 쇼

카운터포인트: AI 엔지니어링-혁명은 지금 시작되었습니다.

AI 시스템을 엔지니어링하기 어렵고 도입하기 위험한 **이유는** 무엇일까요? 저는 그 이유가 기존의 소프트웨어 엔지니어링 접근 방식에 새로운 도전 과제로 제시된 AI의 고유한 특성(데이터 기반 ML과 전통적인 AI 모두) 때문이라고 생각합니다. 이러한 새로운 과제는 기존 방법의 확장만으로는 해결할 수 없습니다. 대신 새로운 AI

이러한 새로운 과제는 기존 방법을 확장하는 것만으로는 해결할 수 없습니다.

근본적으로 AI는 인간이 완전히 이해하지 못할 수도 있는 효과적인 솔루션을 통해 자율적으로 문제를 해결합니다. 심지어 AI는 해결해야 할 새로운 문제를 제안할 수도 있습니다. 이러한 패러다임은 인간이 합리적으로 이해하는 접근 방식과 도구로 구축된 소프트웨어를 통해 문제를 식별하고 해결하는 패러다임과는 근본적으로 다릅니다. 기존의 소프트웨어 엔지니어링 방

법은 대부분 효과 패러다임에 맞춰 제작되었습니다.

요구 사항부터 시작하세요. 소프트웨어 엔지니어링 세계는 요구 사항 중심의 규범적 접근 방식에서 데이터 중심의 탐색적 접근 방식으로 전환하고 있지만,[8] 요구 사항은 여전히 발견되고 있습니다.

그리고 사람이 일부 입력/출력 예상을 통해 지정합니다. 반면, ML 기반 접근 방식의 요구사항은 기본적으로 학습 데이터에 숨겨져 있습니다. ML 방식은 사람이 확인해야 하는 명시적 요구 사항을 발견하는 대신 구현 버전, 즉 학습된 모델만 발견합니다. 이러한 학습된 모델은 학습 데이터와 일치하는 것처럼 보일 수 있지만 사람이 이해할 수 없는 경우가 많습니다. 명시적인 요구사항이 누락되면 가치 정렬에 대한 우려와 의도하지 않은 결과를 초래할 수 있습니다. 더욱 우려스러운 점은 AI가 인간의 요구 사항을 조작하고 변경하는 것이 이를 충족하는 솔루션을 찾는 것보다 훨씬 쉽다는 사실을 발견할 수 있다는 것입니다. 대표적인 예로 추천 엔진이

어 AI 시스템에 규범적인 인간의 가치와 윤리를 불어넣으려는 시도를 하고 있습니다. 이러한 설계상의 접근 방식은 품질 관리를 주입할 곳이 거의 없는 AI 생성 블랙박스 솔루션에서는 잘 작동하지 않습니다. 이는 우리가 다음과 같은 것을 기대할 때 거의 불가피합니다.

는 사람이 전작 콘텐츠를 좋아하도록 조작하여 향후 사전 받아쓰기를 더 쉽게 만듭니다.

다음으로 품질을 살펴보겠습니다. 전통적인 소프트웨어 엔지니어링은 소프트웨어에 품질을 명시적으로 구축하고, 비기능적 요구사항에 대해 검증/검증하며, 설계 논리에 의해 뒷받침되는 다양한 방법을 사용합니다. 이러한 접근 방식은 윤리적 AI로 확장되

인간의 편견, 어리석음, 독창성, 구체적이지 않은 목표가 뒤섞인 AI 학습 데이터 세트와 높은 수준의 최적화 목표만 제공함으로써 최고의 결과를 얻을 수 있습니다.

AI 시스템의 고유한 특성은 기존 소프트웨어 엔지니어링 방법과는 다른 여러 가지 방식으로 나타났습니다. 잠재적인 데이터 변화로 인한 시스템 성능 저하는 일반적인 요구 사항의 진화보다 발견하고 수정하기가 더 어렵습니다. 지속적인 학습의 재현성은 엔지니어링/데이터 위생 및 데이터 품질보다는 여전히 과학적 과제입니다. AI가 우리가 완전히 이해하지 못하는 솔루션을 설계할 때 어떻게 설계별 품질 접근 방식과 제어 전술을 적용할 수 있을까요? 큰 빨간 킬 스위치는 위안을 주는 해결책이라기보다는 절망적인 절망에 가깝습니다.

그럼에도 불구하고 저는 전혀 절망하지 않습니다. 소프트웨어 엔지니어들이, 아마도 이번에는 더 강력한 AI 지원 도구를 통해 해답을 찾을 것이라고 믿습니다. 그러기 위해서는 기존의 패러다임을 넘어서는 사고가 필요합니다. 새로운 세대의 AI 엔지니어링 방법은 다음을 달성하는 데 도움이 되는 기능을 갖춰야 합니다:

- 문제, 요구 사항 및 사양을 지정하는 기존의 접근 방식과 달리 결과와 가드레일을 정의합니다. AI는 우리를 위해 자율적으로 문제를 해결하고 때로는 해결해야 할 새로운 문제를 제안하기도 합니다. 엔지니어가 다양한 사용자 및 커뮤니티와 소통하여 문제를 더 잘 선택하고, 결과를 정의하고, AI에 대한 보다 포괄적인 목표와 가드레일을 지정하는 데 사용할 수 있는 새로운 방법이 필요합니다. 나머지는 AI에 맡기되 그 한계를 더 잘 보장할 수 있습니다.
- 문제를 설정하거나 해결하기보다는 AI가 생성한 솔루션을 이해합니다. 소프트웨어 엔지니어가 사용할 수 있는 새로운 방법과 (AI) 도구가 필요합니다.
- 를 사용하여 정당한 신뢰를 바탕으로 암호화된 AI 기반 솔루션을 해석, 설명 및 단순화할 수 있습니다. 더 중요한 것은 이러한 방법이 알고리즘 수준을 뛰어넘어야 한다는 것입니다.

  엔드투엔드 라이프사이클을 포괄하고 비전문가 이해관계자와의 거버넌스 및 커뮤니케이션을 지원합니다.[9]
- 오류 탐지와 진단뿐 아니라 거버넌스를 위한 근본적인 통합 가시성과 모니터링 기능을 도입하세요. 뚫을 수 없는 솔루션과 잠재적/암묵적 요구 사항을 확인해야 하는 상황에서 특히 고위험 AI 시스템의 사후 배포를 위해서는 관찰 가능성 및 모니터링 가능성에 대한 새로운 접근 방식이 필요합니다. 인간이 상징적인 책임 스펀지가 아니라 '의미 있는' 인간 상호 작용을 위한 새로운 방법이 필요합니다. 우리는 새로운 모니터링 방법이 필요합니다.

  기존의 규칙 기반 데이터 유효성 검사 및 데이터 품질과는 다른 AI용 데이터 입력 품질이 필요합니다. 마지막으로, 사회/행동 과학자들이 인간의 뇌를 완전히 이해하지 못한 채 인간과 사회를 연구하는 것처럼, 우리는 외부에서 신비한 인간의 행동을 연구하고 제어하는 새로운 접근 방식을 발명해야 할지도 모릅니다.

이러한 새로운 세대의 AI 엔지니어링 방법을 개발하는 데는 AI 자체가 근본적인 도움을 줄 수도 있습니다. AI는 이미 코드를 작성하고, 인간 엔지니어에게 코드/모델을 설명하고, 다

른 AI 시스템을 모니터링하고 있습니다. AI가 만들어낸 윤리적 트레이드오프는 인간의 블랙박스 두뇌에서 비롯된 윤리적 요구 사항을 명확히 하고 개선하는 데 도움을 주고 있습니다. 하지만 더욱 흥미로운 사실은 이러한 새로운 AI 엔지니어링 방법이 우리 자신의 지능을 이해하는 데 중요한 역할을 할 수 있다는 사실입니다. 소프트웨어를 구축하는 소프트웨어 엔지니어 시스템은 더 이상 인간의 욕구를 충족시키는 데 그치지 않고 인간의 지능을 이해해야 합니다. 용감한 새로운 세상에는 용감한 새로운 방법이 필요합니다.

승인

이 글에 서브스탠셜 의견을 제공해 주신 CSIRO의 Data61의 마크 스테이플스 박사와 첸 왕 박사, 다프니 리 박사에게 감사의 말씀을 전합니다.

-리밍 주

메리 쇼의 답변 소프트웨어 엔지니어링에는 개념, 이론, 방법, 구체적인 구현 기법 등 확립된 지식이 있습니다. 이는 전체 소프트웨어 시스템의 설계 및 통합뿐만 아니라 이러한 시스템의 개별 구성 요소 개발과도 관련이 있습니다. 소프트웨어 엔지니어링은 혁신적인 설계가 필요한 경우와 일상적이고 선례가 있는 설계로 충분한 경우를 인식하여 적절한 구조와 방법을 선택하는 설계적 사고방식을 요구합니다. 소프트웨어 엔지니어링의 지식은 링크 편집된 단일 스레드 코드 구성 요소로 구성된 초기의 단순한 시스템에서 병렬 및 분산 시스템, 정교한 사용자 인터페이스를 갖춘 상호 작용 시스템, 클라우드 기반 웹 시스템, 서비스 구성, 사이버 물리 시스템 등 새로운 유형의 시스템이 등장함에 따라 발전해 왔습니다.

이러한 시스템의 개별 구성 요소는 여러 종류가 있을 수 있습니다. 가장 친숙한 것은 명령형 또는 함수형 프로그래밍 언어로 코드를 컴파일하여 생성된 코드 구성 요소입니다. 그러나 구성 요소는 데이터 베이스, 제약 조건 집합, 동적 데이터 피드, 전처리기의 출력 등 프로그래밍 언어와는 거리가 먼 것일 수도 있습니다. 이 포트폴리오에 가장 최근에 추가된 모델은 ML의 정교한 추론에 의해 유도된 모델입니다.

를 도출할 수 있습니다. 따라서 일상적인 엔지니어링이 아닌 혁신적인 엔지니어링에 적합한 후보임이 분명합니다.

특히 업데이트 주기가 사용되는 시스템 분석에서 동작의 변화를 반영하지 않는 방식으로 자동화되는 경우, 컴포넌트 유형에 따라 통합에 다른 조정이 필요합니다. 이러한 종류의 자동 업데이트는 현재 일반적으로 발생하며, 소프트웨어 관리자가 사용 중인 구성 요소의 버전을 제어한다는 전통적인 개념에서 벗어나 결과 소프트웨어는 "시스템"이라기보다는 "멀티소스 구성 요소의 연합"으로 더 잘 설명될 수 있습니다. ML은 대량의 데이터, 재교육, 모델 업데이트를 처리하는 등 이 분야에서도 혁신을 요구합니다.

수년 동안 AI 시스템은 연구용 시스템으로, 빠른 프로토타이핑과 진화에 대한 공학적 기대는 높았지만 모든 사용 사례에 대한 장기적인 유지보수 및 정확성에 대한 노력은 낮았습니다. 점차 비즈니스 시스템의 구성 요소로 자리잡았지만, 일반 대중에게 눈에 띄게 영향을 미치는 시스템에서는 비교적 최근에야 그 존재감이 두드러졌습니다. 이제 이들은 견고성, 정확성(또는 최소한 예측 가능성), 장기적인 지원 등 프로덕션 시스템에 대한 엔지니어의 기대에 부응해야 하는 상황에 직면해 있습니다. AI와 ML이 주류로 진입함에 따라 이를 도입하는 시스템 또는 연합도 변화해야 합니다. 리밍 주 [illegible] 모델을 좁게 보면 내부 일관성에 초점을 맞출 수 있지만, 대중에게 [illegible] 하기 위한 안전장치가 필요하다는 점에도 동의합니다.

저자 소개

메리 쇼는 미국 펜실베이니아주 피츠버그에 위치한 카네기멜론대학교의 앨런 J. 펄리스 컴퓨터공학 교수입니다. 그녀의 연구는 소프트웨어 엔지니어링, 특히 소프트웨어 아키텍처와 실제 사람들이 사용하는 시스템 설계에 중점을 두고 있습니다. 그녀는 ACM, IEEE, AAAS의 펠로우입니다. 연락처: shaw@cs.cmu.edu로 문의하세요.

리밍 주(LIMING ZHU)는 호주 뉴사우스웨일스주 시드니 뉴사우스웨일스 대학교의 연구 책임자이자 CSIRO의 데이터61 연구 책임자이며 2015년부터 뉴사우스웨일스 대학교의 공동 정교수입니다. 그는 호주표준협회 블록체인 및 분산 원장 위원회와 AI 신뢰성 관련 ISO 위원회의 위원장을 맡고 있습니다. liming.zhu@data61. csiro.au로 문의하세요.

하지만 제 생각에는 이러한 문제는 기존 시스템에서도 발생할 수 있습니다.

다시 한 번 말씀드립니다: AI와 머신러닝은 소프트웨어 시스템에 새로운 종류의 구성 요소를 가져다주지만, 아직은 생소하고 어렵습니다. 그러나 소프트웨어 엔지니어링에는 익숙한 시스템에 제공되는 특정 솔루션보다 더 광범위한 개념과 이론이 있습니다. 이러한 이론은 AI가 가져다주는 새로운 기회를 해결하는 혁신을 위한 강력한 출발점을 제공합니다. 기존 소프트웨어 엔지니어링 지식을 발전시켜온 기존 경험을 바탕으로 혁신을 이룰 수 있는데 왜 처음부터 AI 구성 요소를 위한 새로운 엔지니어링 방법을 찾아야 할까요?

주 리밍의 응답

쇼 교수님의 통찰력 있는 글은 몇 가지 훌륭

한 논거를 제시했습니다. 저는 기존 접근 방식을 확장하는 것이 AI가 제기하는 몇 가지 과제를 해결하는 데 중요한 역할을 한다는 데 동의합니다. 그러나 비AI 소프트웨어와 AI 소프트웨어가 공유하는 몇 가지 과제가 수십 년 전에 확인되었거나 AI가 가깝고, AI 시스템의 미분적 특성이 물리 법칙보다 더 복잡하고 다음과 비슷할 수 있다는 점에 주목해야 합니다.

에서 영감을 받은 많은 소프트웨어 엔지니어링 접근 방식이 오늘날에도 여전히 이러한 과제와 씨름하고 있는 현실을 외면해서는 안 됩니다. 이러한 문제가 해결되지 않았다는 사실은 새로운 사고의 필요성에 무게를 더합니다. 이러한 과제를 단순히 난이도나 새로운 종류의 구성 요소(데이터, 클라우드, 모빌, AI 등)의 문제로만 파악해서는 안 되며, 기존 접근 방식을 확장하여 가치 있는 결과를 도출하는 데 도움이 될 수 있습니다.

저는 여전히 소프트웨어 엔지니어가 지금까지 주로 문제를 설정하고 해결하는 역할에서 도전 과제를 찾아야 한다고 생각합니다. AI 시스템은 우리가 해결해야 할 문제를 제안하고, 종종 우리가 완전히 이해할 수 없을 정도로 좋은 해결책을 제시할 수 있습니다. 이해할 수 없는 AI 시스템을 이해하는 방법은 엔지니어링보다는 과학에 더

생물학적 시스템과 사회 시스템과 유사합니다. 이 모든 것이 AI 시스템을 엔지니어링하는 방식을 바꾸고 새로운 엔지니어링 능력을 과학의 영역으로 끌어올리거나 그 반대의 경우도 마찬가지입니다.

참조

1. A. Horneman, A. Melinger, 및 I. Ozkaya, "AI 엔지니어링: 11가지 기본 관행," Carnegie Mellon Software Engineering Institute, Pitts- burgh, PA, USA, 백서, 2019. Accessed: Dec. 4, 2021. [온라인]. Available: https://resources.sei.cmu.edu/asset_files/WhitePaper/2019_019_001_634648.pdf
2. C. Kästner와 E. Kang, "인공지능 지원 시스템을 위한 소프트웨어 엔지니어링 교육", *Proc. ACM/IEEE 42nd Int. Conf. Softw. Eng., Softw. Eng. Educ. Training (ICSE-SET 20)*. 뉴욕, NY, 미국: 컴퓨팅 기계 협회, 2020, 45-48쪽, 도이치: 10.1145/3377814.3381714.
3. I. Ozkaya, "AI 지원 시스템 엔지니어링에서 실제로 무엇이 다른가?" *IEEE Softw.*, vol. 4, pp. 3-6, Jul./Aug. 2020, 도이: 10.1109/MS.2020.2993662.
4. M. R. Lyu, *소프트웨어 신뢰성 엔지니어링 핸드북*. 미국 뉴저지 주 피스카타웨이: IEEE Press, 1996. [온라인]. 사용 가능: http://www.cse.cuhk.edu.hk/~lyu/book/reliability/index.html
5. H. W. J. Rittel과 M. M. Webber, "일반 계획 이론의 딜레마", *정책 과학*, 4권, 2호, 155쪽-. 169, 1973, 도이: 10.1007/BF01405730.
6. M. Shaw, "신화와 신화 개념: 프로그램밍 언어가 된다는 것은 어떤 의미일까요?" *Proc. ACM Program. Lang.*, vol. HOPL, p. 44, 2022, 도이: 10.1145/3480947.
7. C. Shapiro and H. R. Varian, *정보 규칙: 네트워크 경제를 위한 전략적 가이드*. 미국 매사추세츠주 보스턴: 하버드 비즈니스 스쿨 프레스, 1999.
8. J. Bosch, H. Holmström Ols- son, I. Crnkovic, "엔지니어링 AI 시스템: *스마트 사이버-물리 시스템을 위한 인공 지능 패러다임에서 "엔지니어링 AI 시스템*: 연구 아젠다", A. K. 루하흐와 A. 엘시, Eds. 미국 펜실베이니아주 허시: IGI 글로벌, 2021, pp. 1-19.
9. Q. Lu, L. Zhu, X. Xu, J. Whittle 및 Z. Xing, "Towards a roadmap on software engineering for responsible AI," in *Proc. Conf. AI Eng. - Softw. Eng. AI (Cain)*, 2022, pp. 101-112.

포인트 카운터포인트

IT Professional
TECHNOLOGY SOLUTIONS FOR THE ENTERPRISE

기사 요청

IT Professional은 기업용 기술 솔루션에 대한 독창적인 제출물을 찾습니다. 주제는 다음과 같습니다.

- 떠오르는 기술,
- 클라우드 컴퓨팅,
- 웹 2.0 및 서비스
- 사이버 보안,
- 모바일 컴퓨팅
- 친환경 IT,
- RFID,
- 소셜 소프트웨어,
- 데이터 관리 및 마이닝,
- 시스템 통합,
- 통신 네트워크
- 데이터센터 운영,
- IT 자산 관리, 그리고
- 건강 정보 기술.

웹 기반 데모와 함께 제공되는 기사를 환영합니다. 자세한 내용은 작성자 가이드라인 (www.computer.org/itpro/author.htm)을 참조하세요.

www.computer.org/itpro

IEEE COMPUTER SOCIETY

IEEE