DORA

2024

Google Cloud

Accelerate State of DevOps

골드 스폰서

catchpoint chronosphere DATADOG

Deloitte. Excella Gearset

liatrio Middleware OPSERA

10

DORA와 함께한 10년

목차

핵심 요약

DORA는 10년이 넘는 기간 동안 실적이 뛰어난 기술 기반 팀과 조직의 역량, 관행, 측정값을 연구해 왔습니다. 본 DORA 보고서는 10호입니다. 전 세계 수많은 산업 분야에서 다양한 규모의 조직에 종사하는 39,000여 명의 전문가 의견을 들었습니다. 이 여정에 동참하여 연구에 중요한 역할을 해주셔서 감사합니다.

DORA는 기술 및 관련 직종에 종사하는 전문가를 대상으로 매년 전 세계적으로 실시하는 설문조사를 통해 데이터를 수집합니다. 이 설문조사에는 조직 전반에 걸쳐 해당 조직의 종사자와 관련된 업무 방식 및 성과 관련 질문이 포함됩니다.

DORA는 엄격한 통계적 평가 방법론을 사용하여 이러한 요소 사이의 관계와 각 요소가 팀과 조직의 성공에 기여하는 방식을 이해합니다.

올해에는 보다 심층적인 인사이트를 얻고, 삼각측량을 실시하고, 연구 결과에 대한 추가적인 맥락을 제공하고자 전문가 심층 인터뷰로 설문조사를 보강했습니다. 자세한 내용은 방법론 장을 참고하시기 바랍니다.

올해 조사한 주요 성과 및 결과

번아웃 줄이기	번아웃은 장기간 또는 과도한 스트레스로 인한 정서적, 신체적, 정신적 피로 상태를 말하며, 주로 냉소, 분리감, 성취감 결여가 특징으로 나타납니다.
흐름	흐름은 개발 작업이 진행되는 동안 얼마나 집중하는지를 측정합니다.
직무 만족도	직무 만족도는 직무에 대한 전반적인 느낌을 측정합니다.
조직 실적	수익성, 시장점유율, 총 고객 수, 운영 효율성, 고객 만족도, 제품 및 서비스 품질, 목표 달성 능력 등의 영역에서 조직의 실적을 측정합니다.
제품 실적	제품의 사용성, 기능성, 가치, 가용성, 실적(예: 지연 시간), 보안을 측정합니다.
생산성	생산성은 개인이 업무를 얼마나 효과적이고 효율적으로 수행하여 가치를 창출하고 과제를 달성하는지를 측정합니다.
팀 실적	팀이 협업하고, 혁신하고, 효율적으로 일하고, 서로 의지하고, 적응하는 능력을 측정합니다.

주요 결과

광범위한 영향을 미치는 AI

AI는 소프트웨어 개발 분야에서 패러다임의 변화를 일으키고 있습니다. 조기 도입에 따라 몇 가지 유망한 결과가 나타나고 있지만, 주의가 필요합니다.

AI 도입의 이점

- 흐름
- 생산성
- 직무 만족도
- 코드 품질
- 내부 문서
- 검토 프로세스
- 팀 실적
- 조직 실적

하지만 AI 도입은 몇 가지 부정적인 영향도 있습니다. 소프트웨어 배포 실적 저하가 관찰되었으며 제품 실적에 미치는 영향은 불확실합니다. 또한 AI 도입이 증가하면서 개인은 가치 있는 작업에 투자하는 시간이 줄어들고 있다고 보고하는데, 이러한 흥미로운 결과는 보고서의 뒷부분에서 자세히 살펴보겠습니다.

팀에서는 AI에 대한 의존도 증가가 미치는 영향에 대해 계속 실험하고 더 많은 것을 알아내야 합니다.

AI에 대한 신뢰가 높아질수록 AI 도입 증가

생성형 인공지능(생성형 AI)을 사용하면 개발자의 생산성이 향상되고, 생성형 AI를 신뢰하는 개발자는 이를 더 많이 사용합니다. 이 부분은 개선의 여지가 있습니다. 응답자의 39.2%는 AI를 거의 또는 전혀 신뢰하지 않는다고 답했습니다.

사용자 중심 전략으로 실적 개선

최종 사용자 환경을 우선시하는 조직은 더 좋은 품질의 제품을 생산하며, 개발자의 생산성과 만족도가 높고 번아웃 가능성이 작습니다.

혁신적 리더십의 중요성

혁신적 리더십은 직원의 생산성, 직무 만족도, 팀 실적, 제품 실적, 조직 실적을 개선하는 동시에 직원의 번아웃을 줄이는 데 도움이 됩니다.

안정적인 우선순위 설정으로 생산성 및 웰빙 향상

조직의 우선순위가 불안정하면 강력한 리더, 양질의 내부 문서, 사용자 중심의 소프트웨어 개발 방식을 갖춘 조직이라도 생산성이 현저히 떨어지고 번아웃이 많이 증가합니다.

플랫폼 엔지니어링을 통한 생산성 향상

플랫폼 엔지니어링은 생산성과 조직 실적에 긍정적인 영향을 미치지만 소프트웨어 배포 실적에는 몇 가지 주의해야 할 신호가 있습니다.

클라우드를 통한 인프라 유연성 확보

유연한 인프라는 조직 실적을 향상할 수 있습니다. 그러나 클라우드가 제공하는 유연성을 도입하지 않고 클라우드로 이전하면 데이터 센터에 그대로 있는 것보다 더 해로울 수 있습니다. 성공적인 마이그레이션을 위해서는 접근 방식, 프로세스, 기술을 혁신해야 합니다.

높은 수준의 소프트웨어 배포 실적 달성 가능

실적이 가장 우수한 팀은 4가지 소프트웨어 배포 측정항목(변경 리드 타임, 배포 빈도, 변경 실패율, 배포 실패 복구 시간) 모두에서 우수한 반면, 실적이 가장 낮은 팀은 4가지 모두에서 실적이 저조합니다. 각 실적 클러스터의 모든 업종 카테고리에서 팀을 볼 수 있습니다.

DORA에서 얻은 인사이트 적용

DORA를 통해 팀과 조직을 개선하려면 현재 상황을 평가하고, 투자 및 개선할 영역을 파악하고, 진행 상황을 알려주는 피드백 루프가 있어야 합니다. 지속적 개선의 사고방식과 관행을 도입하는 팀은 가장 많은 이점을 누릴 가능성이 높습니다. 시간이 지나면서 이를 반복하는 데 필요한 조직 역량을 키우는 데 투자하세요.

연구 결과는 사용자의 실험과 가설에 대한 정보를 제공하는 데 도움이 될 수 있습니다. 변경사항의 영향을 실험하고 측정하여 팀과 조직에 무엇이 가장 효과적인지 확인하는 것이 중요합니다. 이렇게 하면 결과를 검증하는 데 도움이 됩니다. 각자의 결과는 다를 수 있으며, 모두가 여러분의 경험을 통해 배울 수 있도록 진행 상황을 공유해 주세요.

개선에 대해 실험적인 접근 방식을 취하는 것이 좋습니다.

1. 개선하고 싶은 영역이나 결과 확인
2. 기준 또는 현재 상태 측정
3. 원하는 상태에 더 근접할 방법에 대한 일련의 가설 수립
4. 개선 계획에 대한 동의 및 약속
5. 작업 수행
6. 진행 상황 측정
7. 위 과정 반복. 개선 작업은 반복적이고 점진적으로 수행됨

DORA COMMUNITY

혼자만으로 이룰 수 있는 개선은 없습니다.

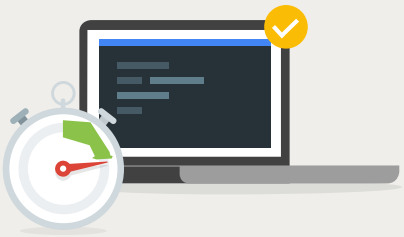
모두 서로의 경험을 통해 배울 수 있습니다. DORA Community 사이트(**https://dora.community**)**에서 개선 이니셔티브에 관한 의견을 나누고 배워보시기 바랍니다.**

소프트웨어 배포 실적

기술 기반 팀은 현재 실적을 평가하고, 개선의 우선순위를 정하고, 진행 상황을 검증할 수 있도록 실적을 측정할 방법이 필요합니다. DORA는 소프트웨어 배포 프로세스의 결과를 효과적으로 측정할 수 있는 4가지 소프트웨어 배포 측정항목(4가지 핵심 측정항목)을 반복해서 검증했습니다.

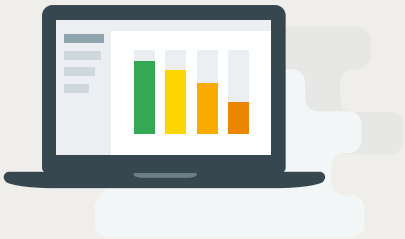
4가지 핵심 측정항목

소프트웨어 변경사항의 처리량과 안정성을 측정하는 데 DORA의 4가지 핵심 측정항목을 사용했습니다. 여기에는 구성 변경과 코드 변경을 포함한 모든 종류의 변경사항이 포함됩니다.

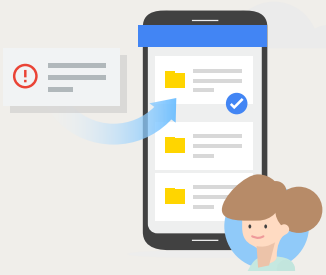


변경 리드 타임:
코드 커밋 또는 변경사항이 프로덕션에 성공적으로 배포되기까지 걸리는 시간

배포 빈도:
애플리케이션 변경사항이 프로덕션에 배포되는 빈도



변경 실패율:
프로덕션에서 오류를 일으켜[1] 핫픽스 또는 롤백이 필요한 배포의 비율



배포 실패 복구 시간:
실패한 배포를 복구하는 데 걸리는 시간

이러한 측정항목은 일반적으로 함께 움직이며, 가장 실적이 좋은 팀은 4가지 측정항목 모두가 양호한 반면 실적이 가장 낮은 팀은 저조한 것으로 나타났습니다.

소프트웨어 배포 실적 측정 방법의 발전

4가지 핵심 측정항목의 분석에는 오랫동안 이상치로 사용된 변경 실패율이 있었습니다.[2] 변경 실패율은 다른 3가지 측정항목과 밀접한 상관관계가 있지만 통계적 테스트와 방법으로는 4가지 모두를 하나의 요소로 결합할 수 없습니다. 응답자들이 변경 실패율 질문에 답하는 방식을 변경하여 연결을 개선했지만 다른 문제가 발생할 수 있다고 생각했습니다.

변경 실패율 측정항목은 팀에 요청되는 재작업의 양을 대신하는 역할을 한다는 오랜 가설이 있습니다. 배포가 실패하면 팀은 다른 변경사항을 도입하여 해당 변경사항을 수정해야 합니다.

이 이론을 테스트하기 위해 올해에는 애플리케이션의 재작업률에 대한 질문을 추가했습니다. "담당하시는 주요 애플리케이션 또는 서비스의 경우, 지난 6개월 동안 계획된 배포는 아니었지만 애플리케이션의 사용자 대상 버그를 해결하기 위해 수행된 배포는 대략 몇 건인가요?"

데이터 분석을 통해 재작업률과 변경 실패율이 관련이 있다는 가설을 확인했습니다. 이 2가지 측정항목을 함께 사용하면 소프트웨어 배포 안정성에 대한 신뢰할 수 있는 요소를 만들 수 있습니다.

이는 소프트웨어 실적 수준 분석에서도 나타납니다. 올해 연구에 참여한 팀 중 절반 이상이 소프트웨어 처리량과 소프트웨어 안정성에서 차이를 보였습니다. 이러한 차이로 인해 2가지 다른 요소를 통해 소프트웨어 배포 실적을 고려했습니다.

개념	
소프트웨어 배포 실적	
요소	
소프트웨어 배포 처리량	소프트웨어 배포 안정성
사용된 측정항목	
• 변경 리드 타임 • 배포 빈도 • 배포 실패 복구 시간	• 변경 실패율 • 재작업률

이 보고서 전반에 적용되는 분석은 소프트웨어 배포 실적 개념과 2가지 요소를 다양한 시점에 활용합니다. 소프트웨어 배포 실적을 설명하기 위해 5가지 측정항목을 모두 고려합니다.

소프트웨어 배포 처리량을 설명할 때 변경 리드 타임, 배포 빈도, 배포 실패 복구 시간을 사용합니다. 이 요소는 모든 종류의 업데이트, 정상적인 변경사항, 실패에 대한 응답으로 변경사항을 만드는 속도를 측정합니다.

변경 실패율과 재작업률은 소프트웨어 배포 안정성을 설명할 때 사용됩니다. 이 요소는 배포가 의도치 않게 즉각적인 추가 작업으로 이어질 가능성을 측정합니다.

실적 수준

매년 설문조사 응답자에게 응답자가 사용하는 주요 애플리케이션 또는 서비스의 소프트웨어 배포 실적에 대해 질문합니다. 그런 다음 클러스터 분석을 사용하여 응답을 분석합니다. 클러스터 분석은 서로 비슷하지만 다른 응답 그룹과 구별되는 응답을 식별하는 통계적 방법입니다.

전년도 클러스터 분석과의 일관성을 유지하기 위해 기존 4가지 소프트웨어 배포 측정항목에 대한 클러스터 분석을 수행했습니다.

소프트웨어 배포 실적을 분석한 결과, 4가지 응답 클러스터가 나타났습니다. 이러한 수준을 미리 설정하지 않고 설문조사 응답에서 자연스럽게 나타나게 합니다. 이를 통해 매년 모든 응답자의 소프트웨어 배포 실적을 한눈에 파악할 수 있습니다.

올해는 아래와 같이 4가지 뚜렷한 클러스터가 데이터에서 나타났습니다.

실적 수준	변경 리드 타임	배포 빈도	변경 실패율	배포 실패 복구 시간	응답자 비율*
최우수	하루 미만	주문형(하루에 여러 번 배포)	5%	1시간 미만	19%(18~20%)
우수	하루에서 일주일 사이	하루 한 번에서 일주일에 한 번 사이	20%	하루 미만	22%(21~23%)
중간	일주일에서 1개월 사이	일주일에 한 번에서 한 달에 한 번 사이	10%	하루 미만	35%(33~36%)
저조	1개월에서 6개월 사이	한 달에 한 번에서 6개월에 한 번 사이	40%	일주일에서 1개월 사이	25%(23~26%)

*89% 불확실성 구간

처리량 또는 안정성

4가지 클러스터 모두에서 처리량과 안정성은 상관관계가 있습니다. 이러한 상관관계는 우수 실적 클러스터(노란색)보다 처리량이 낮고 안정성이 높은 중간 실적 클러스터(주황색)에서도 지속됩니다. 이는 처리량과 안정성 외에 다른 요소가 클러스터 실적에 영향을 미친다는 것을 시사합니다. 예를 들어 중간 실적 클러스터는 변경사항을 더 자주 배포함으로써 이점을 얻을 수 있습니다.

더 자주 배포하는 것과 배포 시 실패를 줄이는 것 중 어느 것이 더 나을까요?

이 질문에 대한 보편적인 해답은 없을 수도 있습니다. 이는 고려 중인 애플리케이션이나 서비스, 해당 애플리케이션을 개발하는 팀의 목표에 따라 다르며, 무엇보다도 애플리케이션 사용자의 기대치에 따라 크게 달라집니다.

더 빠른 팀을 '실적이 우수한 팀', 더 느리지만 안정적인 팀을 '실적이 중간 수준인 팀'이라고 부르기로 했습니다. 이 결정은 이러한 실적 수준을 사용할 때 발생할 수 있는 잠재적 함정 중 하나를 강조합니다. 특정 실적 수준에 도달하는 것보다 개선이 팀에 더 중요하다는 점입니다. 최고의 팀은 최우수 ***실적***을 달성하는 팀이 아니라 뛰어난 ***개선***을 실현하는 팀입니다.

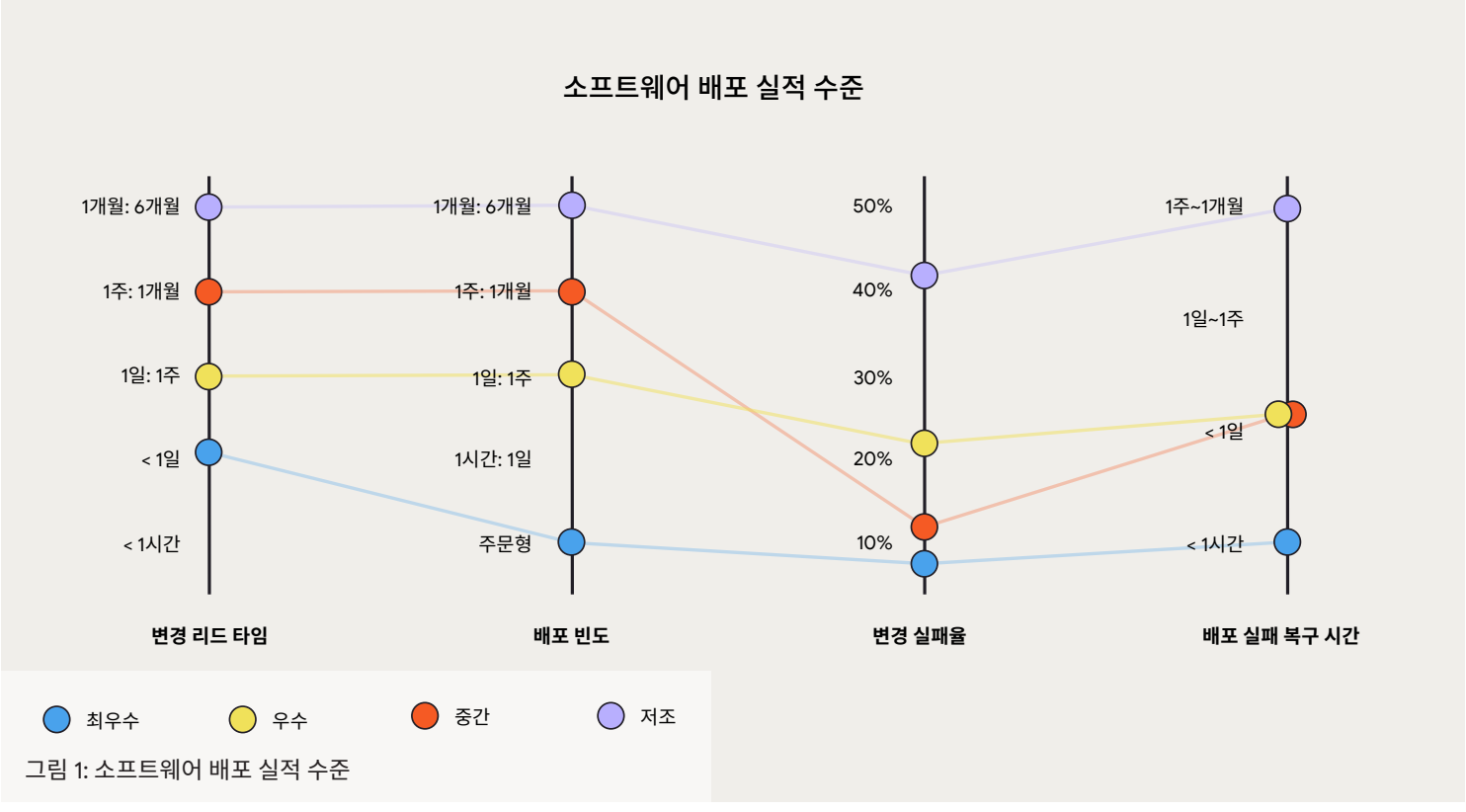


그림 1: 소프트웨어 배포 실적 수준

실적이 낮은 팀과 비교했을 때, 뛰어난 실적을 내는 팀은 다음을 달성합니다.

127배	182배	8배	2,293배
더 빠른 리드 타임	연간 배포 횟수 증가	변경 실패율 감소	배포 실패 복구 시간 단축

실적 클러스터를 사용하는 방법

실적 클러스터는 올해 설문조사 응답자의 소프트웨어 배포 실적을 보여주는 벤치마크 데이터를 제공합니다. 이 클러스터는 뛰어난 실적을 달성하도록 아이디어를 제공하기 위한 것입니다.

특정 실적 수준에 도달하는 것보다 더 중요한 것은 팀이 전반적인 실적을 개선하는 데 집중해야 한다는 것입니다. 최고의 팀은 최우수 ***실적***을 달성하는 팀이 아니라 뛰어난 ***개선***을 실현하는 팀입니다.

업종은 실적 수준에 의미 있는 영향을 미치지 않습니다.

연구 결과, 업종이 소프트웨어 배포 실적의 예측 변수가 되는 경우는 거의 없으며[3], 모든 업종 카테고리에서 높은 실적을 내는 팀을 볼 수 있습니다. 업종별로 고유한 과제가 없는 것은 아니지만, 소프트웨어 배포 실적과 관련하여 어느 한 업종이 고유하게 어려움을 겪거나 고유한 역량을 갖추고 있는 것으로 보이지는 않습니다.

소프트웨어 배포 실적 측정항목 사용

각 애플리케이션이나 서비스에는 고유한 컨텍스트가 있습니다. 이러한 복잡성으로 인해 하나의 변경사항이 시스템의 전체 실적에 어떤 영향을 미칠지 예측하기 어렵습니다. 그 외에도 조직에서 한 번에 한 가지만 변경하는 것은 거의 불가능에 가깝습니다. 이러한 복잡성을 염두에 두고 개선 작업을 진행하는 데 소프트웨어 배포 실적 측정항목을 어떻게 활용할 수 있을까요?

먼저 측정하고 개선하고자 하는 주요 애플리케이션이나 서비스를 식별해야 합니다. 그런 다음 이 애플리케이션을 담당하는 교차 기능 팀을 모아 현재 소프트웨어 배포 실적을 측정하고 합의하는 것이 좋습니다. DORA 빠른 점검(https://dora.dev/quickcheck)은 대화를 안내하고 이 기준 측정을 설정하는 데 도움이 될 수 있습니다. 팀에서는 더 나은 실적을 방해하는 요인이 무엇인지 파악해야 합니다.

이러한 방해 요인을 찾는 효과적인 방법 중 하나는 팀과 함께 가치 흐름 매핑 작업[4]을 완료하는 것입니다.

다음으로 개선 계획을 확인하고 합의합니다. 이 계획은 DORA가 연구한 여러 기능 중 하나를 개선하는 데 초점을 맞출 수도 있고[5], 애플리케이션이나 조직에 고유한 다른 기능에 초점을 맞출 수도 있습니다.

이제 계획을 세웠으니 이제 실행에 옮길 차례입니다. 이 개선 작업에 역량을 집중하고 그 과정에서 얻은 교훈에 주의를 기울여야 합니다.

변경사항을 구현하고 자리를 잡은 후에는 이제 4가지 핵심 측정항목을 다시 평가해야 할 때입니다. 팀이 변경사항을 구현한 후 어떻게 달라졌나요? 어떤 교훈을 얻었나요?

이 프로세스를 반복 수행하면 팀이 지속적 개선에 대한 관행을 구축할 수 있습니다.

변화는 하루아침에 일어나지 않습니다. 학습 분위기, 빠른 흐름, 빠른 피드백을 가능하게 하는 반복적 접근 방식[6]이 필요합니다.

---

1. 최종 사용자가 경험할 수 있는 프로덕션 환경에 배포된 후 문제가 발생하는 경우에만 배포를 변경 실패로 간주합니다. 반대로 프로덕션으로 이동하는 도중에 변경이 중단되면 배포 프로세스의 오류 감지 기능이 성공적으로 입증된 것입니다.
2. 니콜 포스그렌, 제즈 험블, 진 킴. 2018년. Accelerate: The Science Behind DevOps: Building and Scaling High Performing Technology Organizations. IT Revolution Press. pp. 37~38
3. 2019 Accelerate State of DevOps 보고서(32페이지)에 따르면 소매업에서 소프트웨어 배포 실적이 크게 개선되었습니다. https://dora.dev/research/2019/dora-report/2019-dora-accelerate-state-of-devops-report.pdf#page=32
4. https://dora.dev/guides/value-stream-management/
5. https://dora.dev/capabilities
6. https://dora.dev/research

인공지능: 도입 및 태도

핵심 내용

설문조사에 참여한 모든 산업 분야의 대다수 조직은 애플리케이션과 서비스에 AI를 보다 심층적으로 통합하기 위해 우선순위를 바꾸고 있습니다. 이에 따라 대다수의 개발 전문가는 핵심 역할을 수행하는 데 AI를 활용하고 있으며, 그 결과 생산성이 향상되었다고 보고하고 있습니다. 오늘날 시장에서 경쟁력을 유지하기 위해 AI를 사용해야 한다는 개발 전문가의 인식은 보편화되어 있으며, 이는 조직과 개발 전문가 모두에게 AI 도입의 중요한 원동력으로 작용하는 것으로 보입니다.

소개

좋은[1] 영향부터 나쁘고[2] 위험한[3] 영향까지 AI의 영향에 대한 기사가 쏟아지는 것을 보면 올해 AI가 개발 작업 환경에 미친 중대한 영향을 무시하기는 어려울 것입니다. 따라서 2023 Accelerate State of DevOps 보고서[4]에서는 실적에 영향을 미치는 여러 기술 역량 중 하나로만 AI를 논의했지만, 올해는 이 주제를 더 자세히 살펴보고자 합니다.

전문 개발 업무에서 AI 사용이 변방에서 보편화 단계로 빠르게 이동함에 따라 2024 Accelerate State of DevOps 보고서는 업계의 중요한 변곡점에서 개발 전문가의 도입, 사용, 태도를 평가할 중요한 기회라고 생각합니다.

연구 결과

인공지능의 도입

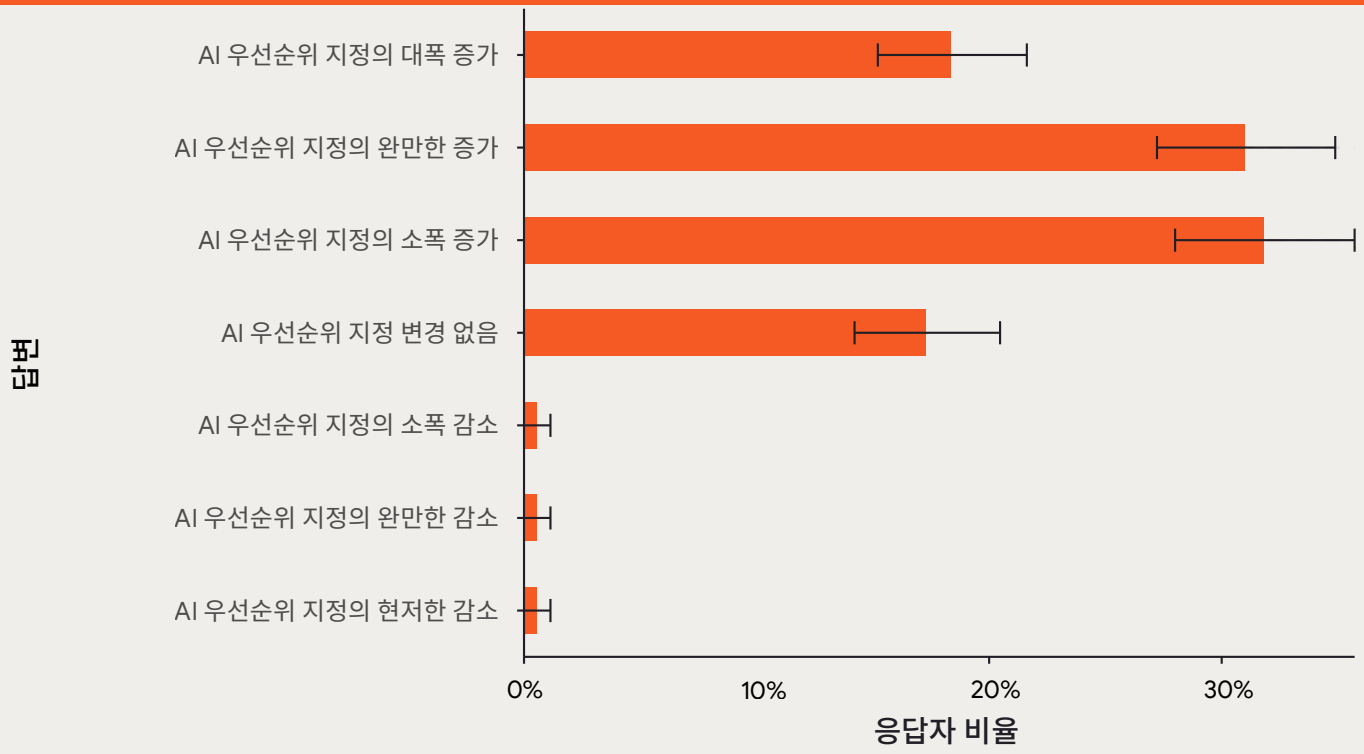
AI 도입에 대한 연구 결과는 AI가 더 이상 '지평선 위에 있는' 것이 아니라 완전히 도래했으며, 앞으로도 계속될 것이라는 인식이 확산되고 있음을 보여줍니다.

조직의 인공지능 도입

응답자의 대다수(81%)는 조직이 애플리케이션과 서비스에서 AI 통합을 확대하기 위해 우선순위를 바꿨다고 답했습니다. 응답자의 49.2%는 이러한 변화의 규모를 '보통' 또는 '상당함'이라고 답했습니다.

특히 응답자의 3%가 소속된 조직에서 AI에 대한 집중도를 낮추고 있다고 답했는데, 이는 설문조사의 오차 범위 내에 있는 수치입니다. 응답자의 78%는 이러한 우선순위의 변화로 인해 조직이 AI 사용 계획에 대해 투명하게 공개할 것으로 믿는다고 답했습니다. 이 데이터는 그림 2에서 확인할 수 있습니다.

AI와 관련된 조직의 우선순위 변화



오차 막대는 89%의 불확실성 구간을 나타냅니다.

그림 2: 조직의 우선순위가 애플리케이션과 서비스에 AI를 통합하는 쪽 또는 통합하지 않는 쪽으로 변화하는 것에 대한 응답자의 인식

설문조사에 참여한 모든 업계 종사자의 일상 업무에서 AI에 대한 의존도가 통계적으로 동일한 수준으로 나타났는데, 이는 AI의 빠른 도입이 모든 산업 분야에 걸쳐 균일하게 확산되고 있음을 시사합니다. 이 결과는 다소 의외였습니다. 개별 산업은 규제 제약 수준과 혁신의 속도에 따라 크게 달라질 수 있으며, 이는 각각 기술 도입률에 영향을 줄 수 있습니다.

그러나 대규모 조직에서 근무하는 응답자가 소규모 조직에서 근무하는 응답자보다 일상 업무에서 AI에 대한 의존도가 낮다고 답했는데, 이는 대기업이 조직의 복잡성과 높은 조정 비용으로 인해 기술 변화에 느리게 적응한다는 기존 연구 결과와 일치하는 결과입니다.[5]

개인의 인공지능 도입

개인 수준에서는 응답자의 75.9%가 일상 업무 중 하나 이상의 업무에서 적어도 부분적으로 AI에 의존하고 있는 것으로 나타났습니다. 다음 업무를 담당하는 응답자 중 대다수는 AI를 활용하고 있었습니다.

1. 코드 작성

2. 정보 요약

3. 생소한 코드 설명

4. 코드 최적화

5. 코드 문서화

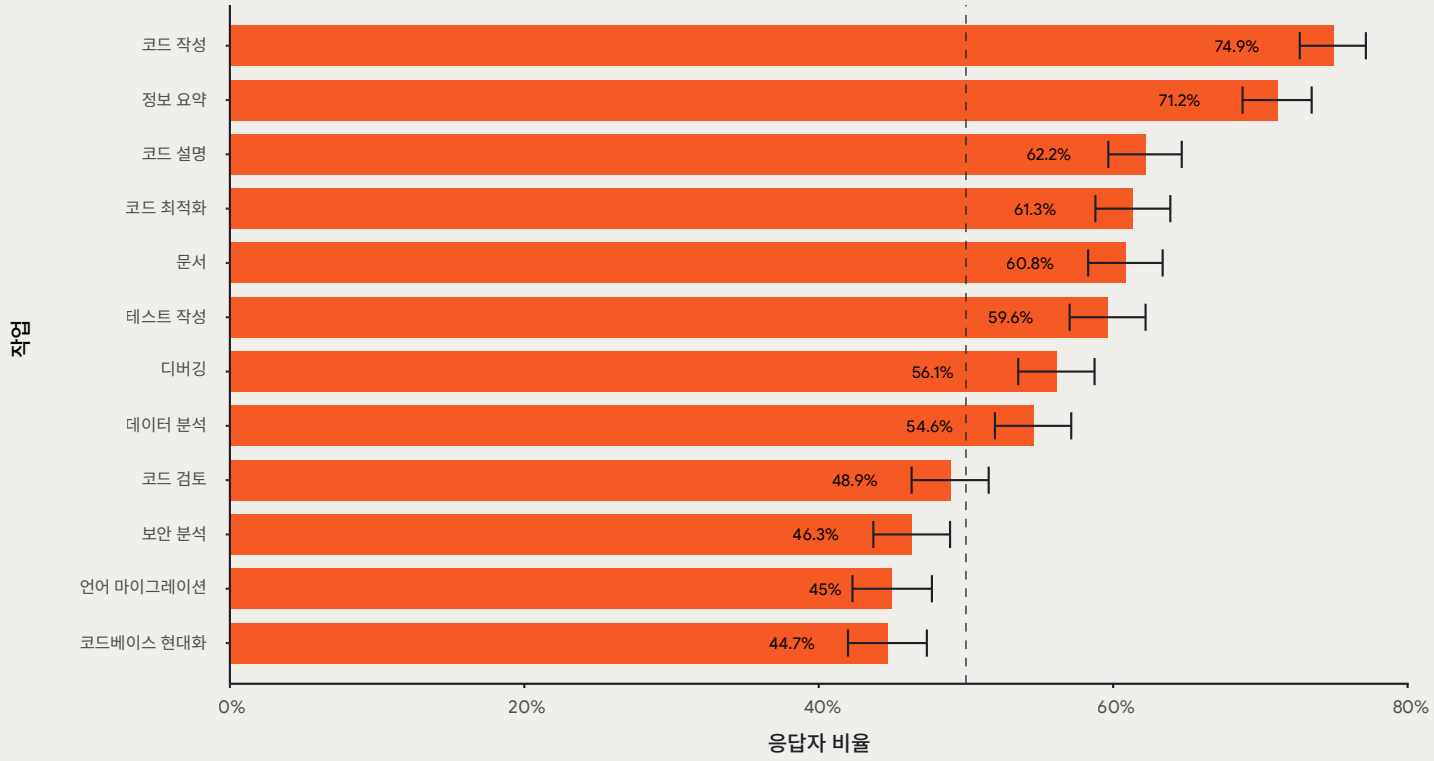
6. 테스트 작성

7. 코드 디버깅

8. 데이터 분석

설문조사 응답에 포함된 모든 업무 중 소프트웨어 개발 업무에서 AI를 가장 많이 활용한 경우는 코드 작성과 정보 요약이었으며, 이러한 업무를 담당하는 응답자의 74.9%와 71.2%가 부분적으로나마 AI에 의존해 업무를 수행하고 있었습니다. 이 데이터는 그림 3에서 확인할 수 있습니다.

AI에 의존하는 작업



오차 막대는 89%의 신뢰도 구간을 나타냅니다.

그림 3: 일반적인 12가지 개발 작업을 수행하기 위해 적어도 부분적으로 AI에 의존하는 응답자 비율

응답자들이 일상 업무에서 AI와 상호작용하는 가장 일반적인 인터페이스는 챗봇(78.2%)이었으며, 외부 웹 인터페이스(73.9%), IDE에 내장된 AI 도구(72.9%) 순으로 나타났습니다. 내부 웹 인터페이스(58.1%)와 자동화된 CI/CD 파이프라인의 일부(50.2%)로 AI를 사용한다는 응답은 상대적으로 적었습니다.

다만, CI/CD 파이프라인과 내부 플랫폼에서 사용되는 AI에 대한 응답자의 인식은 해당 기술을 접하는 빈도에 따라 달라질 수 있습니다. 따라서 이러한 수치는 인위적으로 낮을 수 있습니다.

데이터 과학자와 머신러닝 전문가가 다른 모든 직무를 맡고 있는 응답자보다 AI에 의존할 가능성이 더 높은 것으로 나타났습니다. 반대로 다른 모든 직무의 응답자보다 하드웨어 엔지니어는 AI에 의존하는 비율이 낮았는데, 이는 하드웨어 엔지니어의 업무가 AI가 일반적으로 사용되는 위의 업무와 다르기 때문으로 보입니다.

인공지능 도입의 원동력

인터뷰 참가자들은 AI 도입을 결정한 이유를 경쟁의 압박과 조직과 개발자 모두가 업계 표준을 따라가야 한다는 필요성 때문이라고 답했으며, 이는 점점 더 AI에 대한 숙련도가 중요해지는 것으로 인식되고 있습니다.

여러 참가자의 조직에서는 AI를 사용하는 것 자체가 경쟁사와 차별화할 수 있는 '중요한 마케팅 포인트(P3)'[6]로 간주했습니다. 경쟁사들이 자체 프로세스에 AI를 도입하기 시작했다는 사실을 인지한 한 기업은 “경쟁사가 먼저 이러한 조치를 취하면 어떻게 될까?”라는 의문을 제기하며 새로운 기술 도입에 수반되는 전형적인 '거대한 관료주의'를 포기하고 AI 도입의 절박함을 느낀 적도 있습니다(P11).

개인적 수준에서 많은 참가자는 자신의 AI 도입을 소프트웨어 개발에서 AI 사용 숙련도가 '엔지니어로서 진입하기 위한 새로운 기준'이라는 생각과 연결했습니다(P9). 몇몇 참가자는 “이 분야에서 너무 많은 일이 일어나고 있어 따라잡기 힘들다 AI를 사용하지 않으면 곧 뒤처질 것 같다”(P4)며 동료 개발자들이 개발 워크플로에 AI를 빠르게 도입해야 한다고 제안했습니다.

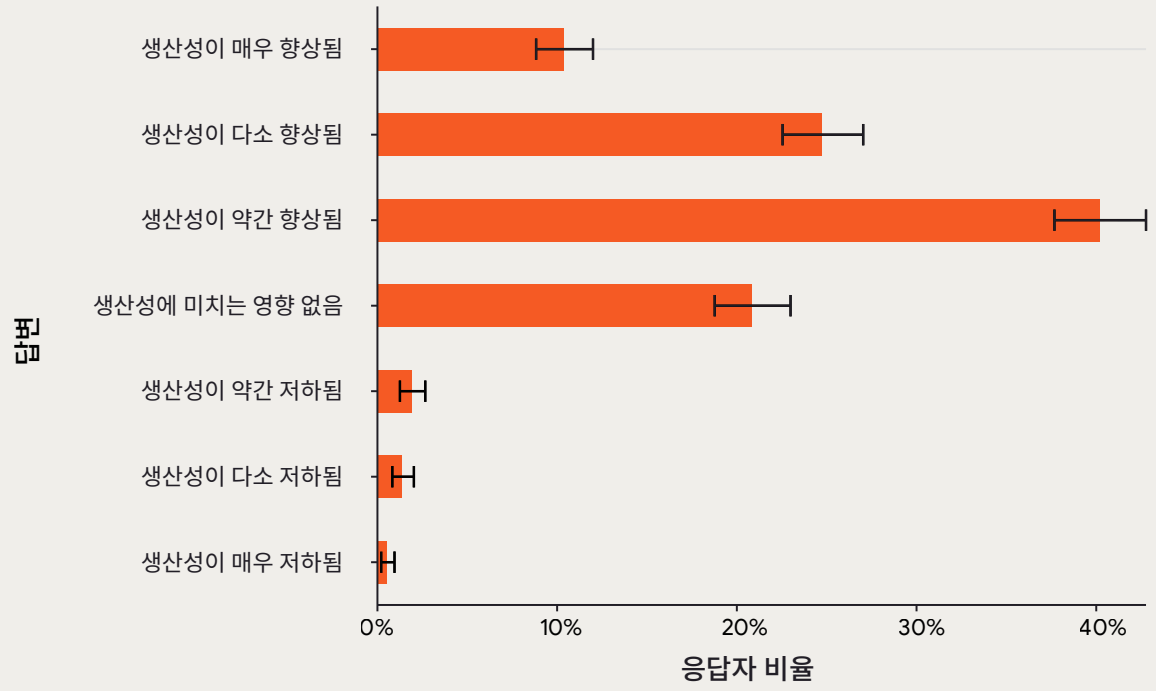
인공지능에 대한 인식

인공지능을 통한 실적 개선

AI를 도입하고 있는 많은 조직과 개발자가 개발 업무에 AI를 사용함으로써 얻는 이점은 상당한 것으로 보입니다. 응답자의 75%는 2024년 초에 실시된 설문조사 직전 3개월 동안 AI를 통해 긍정적인 생산성 향상을 경험했다고 답했습니다.

특히 3분의 1이 넘는 응답자가 관찰된 생산성 향상의 정도가 보통(25%) 또는 매우 높음(10%)이라고 답했습니다. 응답자의 10% 미만이 AI로 인해 생산성에 조금이라도 부정적인 영향을 받았다고 답했습니다. 이 데이터는 그림 4에서 확인할 수 있습니다.

AI로 인한 생산성 변화에 대한 인식



오차 막대는 89%의 불확실성 구간을 나타냅니다.

그림 4: AI가 생산성에 미치는 영향에 대한 응답자의 인식

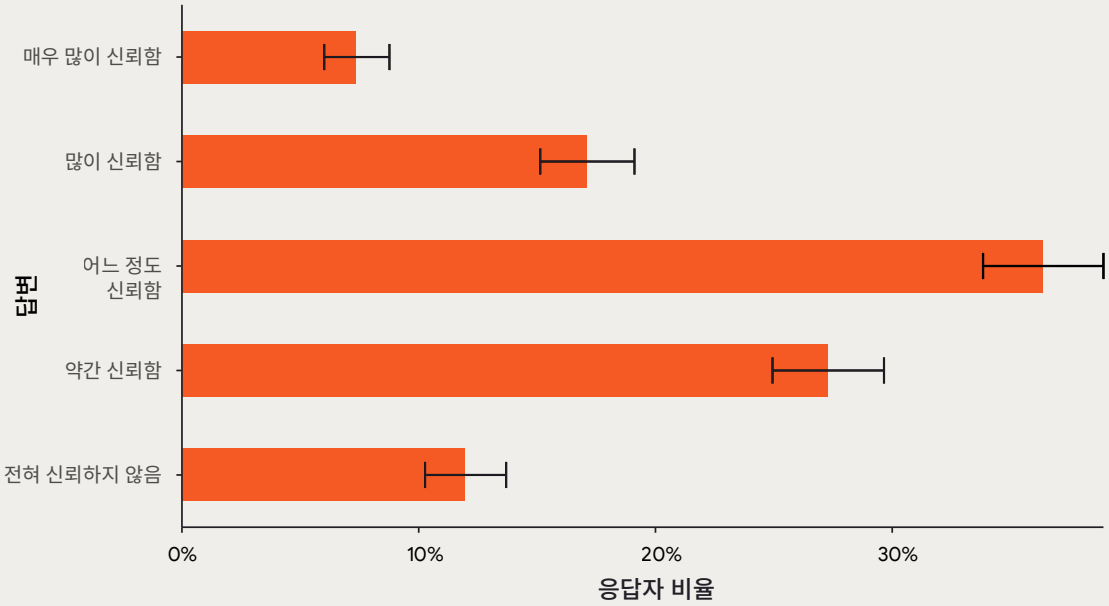
모든 직군에서 AI를 통해 생산성이 가장 크게 향상되었다고 답한 응답자는 보안 전문가, 시스템 관리자, 풀 스택 개발자였습니다. 모바일 개발자, 사이트 안정성 엔지니어, 프로젝트 매니저 역시 긍정적인 생산성 향상이 있었다고 보고했지만, 다른 모든 직군에 비해 생산성 향상 효과는 낮았다고 답했습니다.

개발 작업에서 AI의 참신함과 그에 따른 학습 곡선이 개발자의 코드 작성 능력을 저해할 수 있다고 추측했지만, 연구 결과는 이러한 가설을 뒷받침하지 못했습니다. 응답자의 5%만이 AI가 코드 작성 능력을 어느 정도 저해했다고 답했습니다. 실제로 응답자의 67%는 AI 지원 코딩 도구 덕분에 코드 작성 능력이 어느 정도 향상되었다고 답했으며, 약 10%는 AI 덕분에 코드 작성 능력이 '매우' 향상되었다고 답했습니다.

AI 생성 코드에 대한 신뢰도

개발 작업에 사용되는 AI 생성 코드의 신뢰도에 대한 참가자의 인식은 복합적이었습니다. 응답자의 대다수(87.9%)가 AI 생성 코드의 품질을 어느 정도 신뢰한다고 답했지만, 응답자의 39.2%가 거의 신뢰하지 않거나(27.3%) 전혀 신뢰하지 않는(11.9%) 등 AI 생성 코드의 품질을 신뢰하는 정도는 전반적으로 낮았습니다. 이 데이터는 그림 5에서 확인할 수 있습니다.

AI 생성 코드의 품질에 대한 신뢰도



오차 막대는 89%의 불확실성 구간을 나타냅니다.

그림 5: 응답자들은 AI 생성 코드의 품질을 신뢰한다고 답했습니다.

개발자들이 AI를 빠르게 도입하고, AI에 의존하며, AI가 성과에 긍정적인 기여를 한다고 인식하고 있다는 설문조사 결과를 고려할 때, AI에 대한 전반적인 신뢰도가 낮다는 것은 놀라운 결과입니다. 한 가지 주목할 점은 인터뷰 중 많은 참가자가 자신의 전문 업무에 사용한 AI 생성 코드의 결과물을 수정할 의향이 있거나 수정할 것으로 예상한다고 답했다는 점입니다.

한 참가자는 AI 생성 코드의 결과물을 평가하고 수정해야 하는 상황을 “Stack Overflow의 초창기에는 사용자들이 경험이 많아서 무엇을 해야 할지 정확히 알고 있다고 생각했습니다. 그렇게 복사하여 붙여넣기만 하다가 보면 일이 갑자기 폭발적으로 늘어나죠.”(P2)라고 설명했습니다.

새로운 문제가 아니기 때문인지, P3와 같은 참가자들은 '누군가가 Copilot이나 ChatGPT에서 코드를 복사하여 붙여넣는 것에 대해 걱정'하지 않는다며, 기존의 코드 품질 보증 프로세스를 통해 '이를 확인할 수 있는 계층이 많기 때문'이라고 말했습니다.

개발자가 AI 생성 코드의 정확성에 대해 절대적인 신뢰를 기대하는 것은 아니며, 개발자가 AI 생성 코드를 유용하다고 생각하기 위해 절대적인 신뢰가 필요한 것 같지도 않다는 가설을 세웠습니다. 오히려 약간의 조정으로 완성도를 높일 수 있는 거의 정확한 AI 생성 코드는 널리 도입하고 사용하도록 장려할 가치가 충분하며 기존 품질 보증 프로세스와도 호환되는 것으로 보입니다.

AI의 미래에 대한 기대치

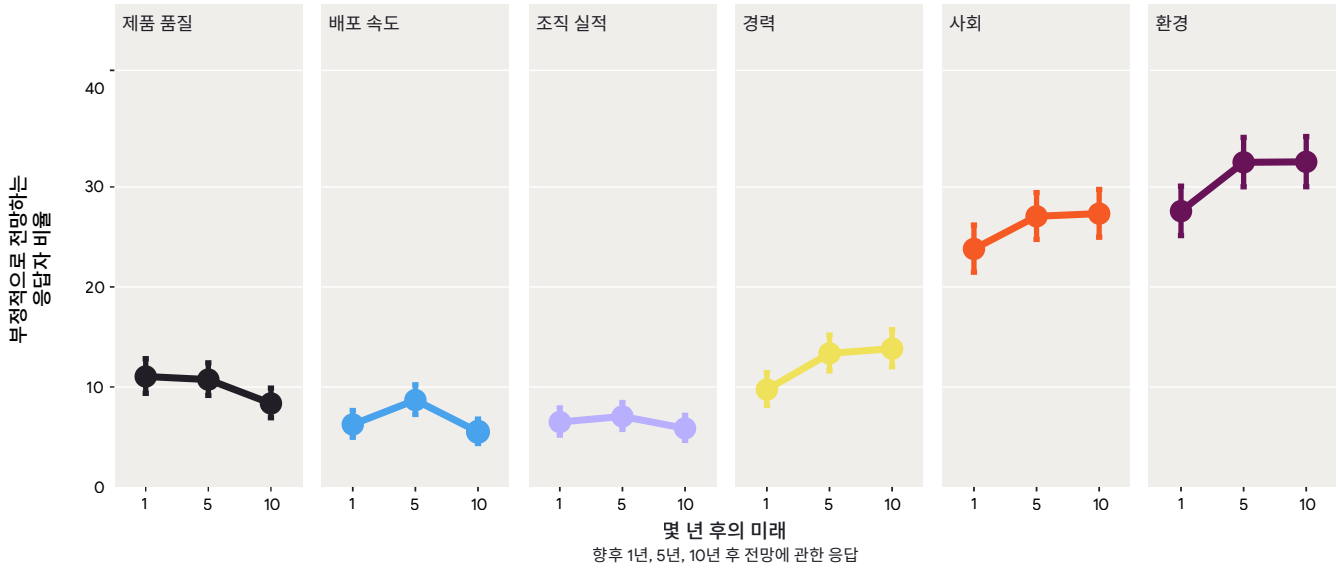
연구 결과에 따르면 전반적으로 AI는 이미 개발 전문가의 업무에 막대한 영향을 미치고 있으며, 이러한 추세는 계속 증가할 것으로 예상됩니다. 앞으로 AI가 개발, 더 나아가 세상에 어떤 영향을 미칠지 정확히 예측하는 것은 불가능하지만, 응답자에게 향후 1년, 5년, 10년 후 AI가 미칠 영향에 대해 추측하고 기대치를 공유해 달라고 요청했습니다.

응답자들은 최근 경험을 바탕으로 AI가 개발 업무에 미친 영향에 대해 상당히 긍정적인 반응을 보였지만, 향후 AI의 영향에 대한 예측은 그다지 희망적이지 않았습니다.

낙관적으로는 AI가 개발 전문가의 실적에 긍정적인 영향을 미쳤다는 연구 결과와 일관되게 응답자는 향후 1년, 5년, 10년 동안 AI를 통해 제품의 품질이 지속해서 향상될 것으로 기대한다고 답했습니다.

하지만 응답자들은 AI가 자신의 직업, 환경, 사회 전반에 순전히 부정적인 영향을 미칠 것이며, 이러한 부정적인 영향은 약 5년 후에 완전히 실현될 것이라고 예상했습니다. 이 데이터는 그림 6에서 확인할 수 있습니다.

AI로 인해 예상되는 부정적인 영향



오차 막대는 89%의 신뢰도 구간을 나타냅니다.

그림 6: 향후 1년, 5년, 10년 후 AI가 미칠 부정적인 영향에 대한 응답자의 기대치

인터뷰 참가자들은 설문조사 응답자들과 마찬가지로 AI가 미래에 미칠 영향에 대한 생각이 다양했습니다. 일부 참가자는 아직 미비한 규제 환경에서의 향후 법적 조치에 대해 궁금해하면서 '법적 조치가 결정되면 반대편에 서게 될지도 모른다'고 우려했습니다(P3).

다른 참가자는 “AI가 인간을 대체하게 될까요? 미래는 알 수 없지만 아마도 그렇겠지요.” 라며 오랫동안 품어온 불안감을 드러냈습니다(P2). 반면 동료들은 “사람들이 'Y2K가 온다. 모든 게 망할 거야’라고 말하던 시절이 있었죠. 어쩌고저쩌고 그 당시에는 새로운 것이었으니까요. [하지만] 아무것도 바뀌지 않았습니다. 오히려 더 많은 일자리가 창출되었죠. AI도 마찬가지라고 생각합니다(P1).”라며 Y2K 시절과 비교하며 두려운 감정을 일축하기도 했습니다.

AI가 앞으로 세상에 미칠 영향은 아직 불분명합니다. 하지만 올해 설문조사는 AI가 소프트웨어 개발 분야에서 무시할 수 없는 패러다임의 변화를 일으켰음을 여실히 보여주고 있습니다. 지금까지 이러한 변화는 개발 전문가로부터 호평을 받고 있습니다.

1. https://www.sciencedaily.com/releases/2024/03/240306144729.htm
2. https://tech.co/news/list-ai-failures-mistakes-errors
3. https://klyker.com/absurd-yoga-poses-generated-by-ai/
4. https://dora.dev/dora-report-2023
5. 에버렛 M. 로저스, 아르빈드 싱할, 마가렛 M. 퀸란. 'Diffusion of innovations.' An Integrated Approach to Communication Theory and Research. Routledge, 2014년. 432-44, Tornatzky, L. G., & Fleischer, M. (1990). The processes of technological innovation. Lexington, MA: Lexington Books
6. (P[N]), 예를 들어 (P1)은 인터뷰 참가자의 가명을 나타냅니다.

AI의 다운스트림 영향 살펴보기

핵심 내용

이 장에서는 개인 개발자부터 전체 조직에 이르기까지 전반적으로 AI 도입이 미치는 영향에 대해 살펴봅니다. 연구 결과, 분명한 이점과 예상치 못한 문제점이 모두 포함된 복잡한 그림이 드러났습니다. AI를 도입하면 개인의 생산성, 업무 흐름, 업무 만족도가 향상되는 반면, 가치 있는 작업에 소요되는 시간은 줄어들 수도 있습니다.

마찬가지로 AI는 코드 품질, 문서화, 검토 프로세스에 긍정적인 영향을 미치지만, 놀랍게도 이러한 이점이 소프트웨어 배포 실적 개선으로 이어지지는 않습니다. 실제로 AI 도입은 이러한 영역에서 부정적인 영향을 주는 것으로 보이며, 제품 실적에 미치는 영향은 미미한 수준에 머물러 있습니다.

이러한 어려움에도 불구하고 AI 도입은 팀과 조직의 실적 향상과 관련이 있습니다. 이 장에서는 소프트웨어 개발에서 AI의 역할을 냉정하게 평가하고 이점을 극대화하고 예기치 않은 결과를 완화하기 위해 AI를 선제적으로 적용해야 한다는 내용으로 마무리합니다.

AI의 순간과 DORA

주요 기술 대기업들은 향후 5년 동안 AI 개발에 약 1조 달러를 투자할 것으로 추산됩니다.[1] 이는 '인공지능: 도입과 태도' 장에 제시된 통계와 잘 맞아떨어지는데, 응답자의 81%가 자신의 회사가 AI 개발에 리소스를 투입했다고 답했습니다.

AI의 환경적 영향은 비용을 더욱 가중시킵니다. 일부 추정치에 따르면 2030년까지 AI로 인해 데이터 센터의 전력 수요가 160% 증가할 것으로 예상됩니다.[2] AI 모델의 학습은 대략 '미국 1,000여 가구의 연간 전력 소비량'에 달할 수 있습니다.[3] 30%가 넘는 응답자가 환경에 AI가 해로울 것으로 생각한다는 점은 놀라운 일이 아닙니다.

개발 및 환경 비용 외에 도입 비용이 발생할 가능성도 있습니다.

이는 생산성 저하부터 전문가 채용에 이르기까지 다양한 형태로 나타날 수 있습니다. 도입 비용은 사회적 차원에서도 발생할 수 있습니다. 응답자의 3분의 1 이상은 향후 10년 이내에 AI가 사회에 해를 끼칠 것으로 생각합니다. 이러한 비용을 고려할 때 수익률에 대한 사람들의 호기심이 커지는 것은 당연해 보입니다.

이러한 호기심은 적어도 어느 정도는 감정과 데이터가 혼재된 수많은 미디어, 기사, 연구 결과에서 나타났습니다.

어떤 사람들은 AI가 인류의 능력을 극적으로 개선했다고 믿지만,[4] 어떤 사람들은 AI가 숙제를 도와주는 단순한 도구에 불과하다고 말하며,[5] 어떤 사람들은 AI로 인해 인류가 멸망할 것이라고 우려합니다.[6]

특정 작업을 성공적으로 완료하는 능력과 같은 가까운 성과에 대한 증거는 대체로 긍정적입니다.[7] 팀의 코드베이스와 같이 성과가 좀 더 먼 곳에 있게 되면 결과는 조금 덜 명확해지고 덜 긍정적으로 됩니다. 예를 들어 일부 연구에 따르면 코드 이탈이 2021년 이전 기준선보다 두 배로 증가할 수 있다고 합니다.[8]

이러한 다운스트림 효과를 이해하는 것은 어려운 일이 아닙니다. 결과가 원인에서 멀어질수록 연관성이 덜 뚜렷하고 덜 명확해집니다.

AI의 다운스트림 효과를 평가하는 것은 호수에 던진 돌의 효과를 정량화하는 것과 유사합니다. 물속 바위의 충격 지점에서 가장 가까운 물결을 가장 쉽게 파악할 수 있지만, 진입 지점에서 멀어질수록 바위의 영향이 덜 뚜렷해지고 그곳에 나타난 파장이 바위의 영향이라고 말하기는 더 어렵습니다.

AI는 본질적으로 다른 프로세스와 역학의 폭풍우가 몰아치는 바다에 던져진 돌멩이입니다. AI(또는 모든 기술이나 관행)로 인한 파장의 정도를 파악하는 것은 어려운 일입니다. 이는 업계가 AI의 영향을 이해하기 위해 측정 및 분석 프레임워크를 원칙적으로 도입하는 데 어려움을 겪어온 이유 중 하나일 수 있습니다.[9]

Google의 접근 방식은 이러한 유형의 과제에 유용하게 사용할 수 있도록 특별히 설계되었습니다. DORA는 관행의 유용성 또는 비유용성을 이해하도록 설계되었습니다. 지난 10년 동안 보안 관행, 혁신적 리더십, 생성적 문화, 문서 관행, 지속적 통합, 지속적 배포, 사용자 중심 전략 등 수많은 관행의 다운스트림 영향을 살펴봤습니다.[10]

특히 다양한 결과물에서 AI의 영향을 탐구하는 과정에서 DORA의 접근 방식[11]이 AI의 영향에 대해 학습하는 데 도움이 될 수 있다고 믿습니다.

AI 도입 측정하기

AI 도입의 영향을 파악하기 위한 첫 번째 과제는 AI 도입을 측정하는 것입니다. 사용 빈도를 측정하는 것은 개발 워크플로에서 AI의 비중을 이해하는 데 있어 의존도를 측정하는 것만큼의 의미는 없을 수 있다고 판단했습니다. 코드 검토나 문서 작성을 한 달에 몇 번 또는 두 달에 한 번만 할 수도 있지만, 이러한 작업은 업무에 매우 중요하다고 생각할 수 있습니다.

반대로, AI를 자주 사용한다고 해서 자신의 역할에서 중요하거나 핵심적이라고 생각하는 업무에 AI를 사용하고 있다는 의미는 아닙니다.

이러한 점을 고려하여 응답자들에게 일반적인 업무와 특정 업무에서 AI에 대한 의존도에 대해 질문했습니다. 이전 장에서는 설문조사의 결과와 그 해석에 대해 자세히 설명했습니다.

요인 분석을 통해 '일반' AI 의존도 설문조사 항목이 아래 작업에서 보고된 AI 의존도와 중복되는 부분이 많다는 것을 발견했습니다.

- 코드 작성
- 정보 요약
- 코드 설명
- 코드 최적화
- 문서
- 테스트 작성

이 7가지 항목 간의 강한 공통성과 공분산은 AI 도입이라는 근본적인 요인을 제시합니다.

AI가 개인에게 미치는 영향과 관련된 분명한 이점과 몇 가지 잠재적인 절충안

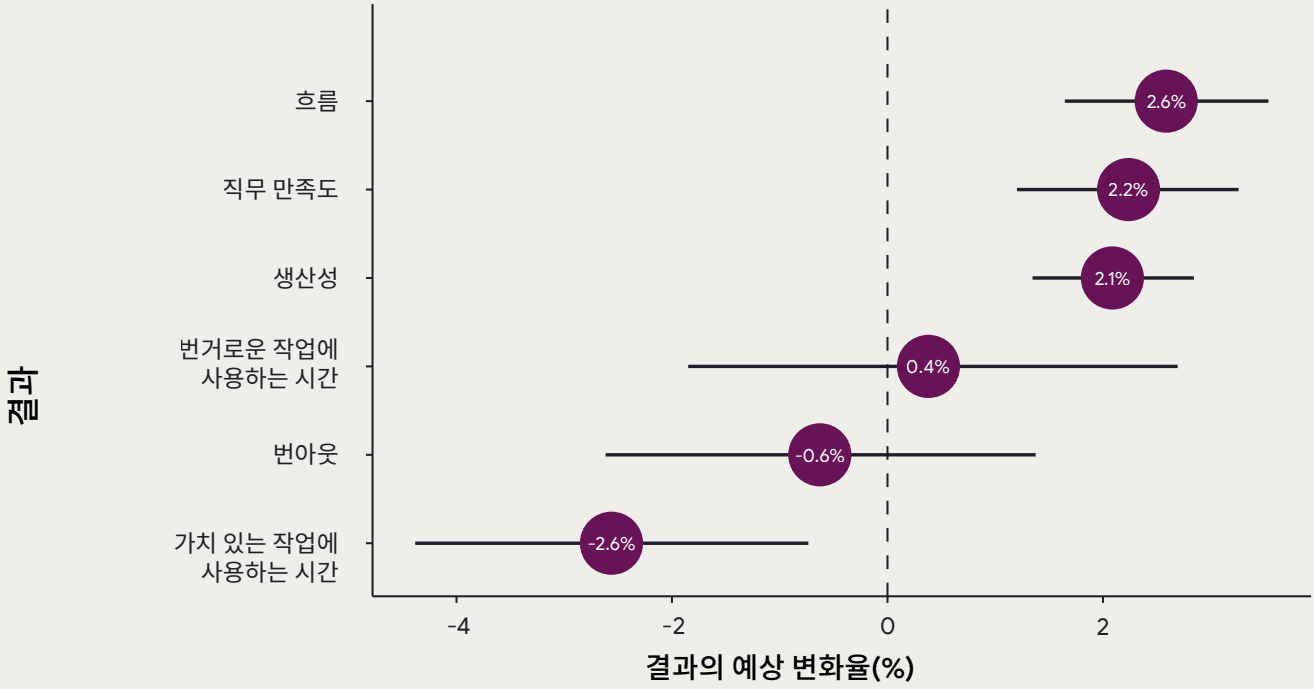
매년 그랬던 것처럼 개인의 성공과 웰빙과 관련된 다양한 구성 요소를 측정했습니다.

직무 만족도	직업에 대한 전반적인 느낌을 파악하기 위해 고안된 단일 항목입니다.
번아웃	번아웃의 신체적, 정서적, 심리적 차원은 물론 개인 생활에 미치는 영향까지 포괄하는 다면적인 특성을 요약하는 요소입니다.
흐름	개발 작업을 수행하는 동안 얼마나 집중하는 경향이 있는지 파악하기 위해 고안된 단일 항목입니다.
생산성	개인이 업무에서 효과적이고 효율적이라고 느끼는 정도, 가치를 창출하고 과제를 달성하는 정도를 측정하기 위해 고안된 요인 점수입니다.
번거로운 작업에 사용하는 시간	개인이 장기적인 가치가 거의 없는 반복적인 수작업에 소비하는 시간 비율을 측정하는 단일 항목입니다.
가치 있는 작업에 사용하는 시간	개인이 가치 있다고 생각하는 작업에 소비하는 시간의 비율을 측정하는 단일 항목입니다.

응답자들이 이러한 질문에 답하는 방식이 AI 도입에 따라 달라지는지 알아보고 싶었습니다. 결과를 보면 그런 경우가 많다는 것을 알 수 있습니다.

그림 7은 AI 도입이 개인의 성공과 웰빙에 미치는 영향에 대한 최선의 추정치를 시각화해서 보여줍니다.

개인이 AI 도입을 25% 늘린다면…



포인트 = 예상값
오차 막대 = 89% 불확실성 구간
그림 7: AI 도입이 개인의 성공과 웰빙에 미치는 영향

확실한 이점

개인을 위한 AI 도입의 이점에 대한 이야기는 대체로 호의적이지만, 여느 좋은 이야기와 마찬가지로 몇 가지 아쉬운 점이 있습니다. **분명한 것은 AI가 흐름, 생산성, 업무 만족도에 실질적이고 유익한 영향을 미친다는 점입니다 (그림 7 참조)**.

예를 들어 개인의 AI 도입률이 25% 증가하면 생산성은 약 2.1% 증가할 수 있습니다(그림 7 참조). 사소해 보일 수 있지만 이는 개인 차원의 수치입니다. 이 패턴이 수십 명의 개발자 또는 수만 명의 개발자에게 확대 적용된다고 상상해 보세요.

이러한 패턴은 Google이 예상한 것입니다. 이는 부분적으로는 서로 다른 정보 소스를 종합하여 한 곳에서 고도로 맞춤형 답변을 제공하는 AI의 능력 덕분이라고 생각합니다. 혼자서 이 작업을 수행하려면 시간이 오래 걸리고 컨텍스트 전환이 많으며 흐름이 원활하지 않을 수 있습니다.

생산성과 흐름이 직무 만족도와 밀접한 관련이 있다는 점을 고려할 때, AI 도입이 직무 만족도 향상으로 이어지는 것은 놀라운 일이 아닙니다.

잠재적인 장단점

여기서부터 이야기가 조금 복잡해집니다. AI 도입의 한 가지 가치 제안은 사람들이 가치 있는 작업에 더 많은 시간을 할애할 수 있도록 도와준다는 점입니다. 즉, 수동적이고 반복적이며 번거로운 작업을 자동화함으로써 응답자가 '더 나은 일'에 시간을 자유롭게 사용할 수 있을 것으로 기대합니다. 그러나 데이터에 따르면 AI 도입이 증가하면 가치 있는 작업에 소요되는 시간은 줄어드는 반면, 번거로운 업무에 소요되는 시간은 영향을 받지 않는 것으로 나타났습니다.

업무 흐름, 업무 만족도, 생산성 등 응답자의 웰빙을 나타내는 지표는 지금까지 가치 있는 작업을 하는 데 소요되는 시간과 관련이 있었습니다. 따라서 가치 있는 작업에 소요되는 시간의 감소와는 별개로 이러한 지표의 증가가 관찰된 것은 놀라운 일입니다.

이러한 패턴을 잘 설명하려면 부조화처럼 보이는 이 문제와 씨름해야 합니다. 좋은 영화 설명은 설명과 모순되는 장면을 무시할 수 없습니다. 좋은 책 설명은 설명과 잘 들어맞지 않는 장을 무시할 수 없습니다. 마찬가지로, 이러한 패턴에 대한 좋은 설명은 단순한 이야기를 들려주는 패턴의 하위 집합에만 집중해서는 안 됩니다.

데이터에 맞는 가설은 무수히 많지만 가치 있는 작업에 소요되는 시간은 줄어들고 반복 업무는 그대로인 반면, 업무 흐름, 생산성, 직무 만족도는 AI로 인해 향상된다는 가설이 가장 합리적이라고 생각했습니다.

이 가설을 진공 가설(vacuum hypothesis)이라고 부릅니다. AI는 생산성과 흐름을 개선하여 사람들이 보다 효율적으로 일할 수 있도록 지원합니다. 이러한 효율성은 사람들이 가치 있다고 생각하는 작업을 더 빨리 마무리하는 데 도움이 됩니다.

여기서 진공 상태가 만들어지며 여분의 시간이 생깁니다. AI는 응답자의 작업에서 가치를 훔치는 것이 아니라 작업의 실현을 가속화합니다.

가치 있는 작업이란?

이러한 이해하기 어려운 결과를 이해하기 위해 응답자들이 가치 있는 일과 번거로운 일이라고 판단하는 일의 유형을 더 자세히 연구했습니다.

전통적인 관념, 과거 보고서, 인터뷰의 정성적 데이터에 따르면 응답자들은 코딩과 같은 개발 관련 업무를 가치 있는 작업으로 여기는 반면, 회의 참석과 같은 조직의 조정과 관련된 업무는 덜 가치 있고 번거로운 일로 여기는 것으로 나타났습니다. 이 분류 체계 내에서 응답자들이 정의한 것에 따르면 AI는 '번거로운' 업무보다 '가치 있는' 업무를 지원하는 것이 더 적합합니다.

인터뷰의 정성적 데이터를 살펴본 결과, 참가자들은 자신의 업무가 '의미 있는 일'이라고 생각하는지에 대한 운영자의 질문에 답할 때 자신의 업무가 다른 사람에게 미치는 영향과 관련하여 업무의 가치를 측정하는 경우가 많다는 사실을 확인했습니다.

이는 지난 2년간 사용자 중심 전략이 직무 만족도에 미치는 매우 유익한 영향에 대한 DORA의 연구 결과를 통해 더욱 확고해졌습니다.

예를 들어 최근의 역할 변경을 설명할 때 P10[12]은 "더 많은 사람과 더 많은 부분에 영향을 미칠 수 있게 되어" 결정을 내렸다고 답했습니다. 마찬가지로 P11은 “처음부터 무언가를 빌드하고 그것이 소비자나 고객에게 배포되는 것을 보면 성취감을 느낄 수 있고 스스로 '그래, 내가 이걸 배포했으니 사람들이 사용하는구나'라고 느낄 수 있습니다."라고 말했습니다.

개발 작업의 '유의미성'은 코드 작성에서 직접적으로 파생되는 것이 아니라 개발된 솔루션의 영향력에서 파생된다는 점을 이해하면 응답자들이 가치 있는 작업에 더 적은 시간을 소비하면서도 업무에 더 만족하는 이유를 설명하는 데 도움이 됩니다.

AI는 사람들이 가치 있게 여기는 작업을 더 쉽고 빠르게 만들어 주지만, 사람들이 좋아하지 않는 작업에는 큰 도움이 되지 못합니다. AI 도입에도 불구하고 반복 업무와 번아웃은 변하지 않고 그대로인 상태에서 이런 일이 발생한다는 것은 AI가 회의, 관료주의 및 기타 여러 번거로운 업무를 피할 수 있도록 도와주는 역할을 하지 못했음을 강조합니다(그림 8).

좋은 소식은 AI로 인해 상황이 악화되거나 응답자의 웰빙에 부정적인 영향을 미치지는 않았다는 것입니다.

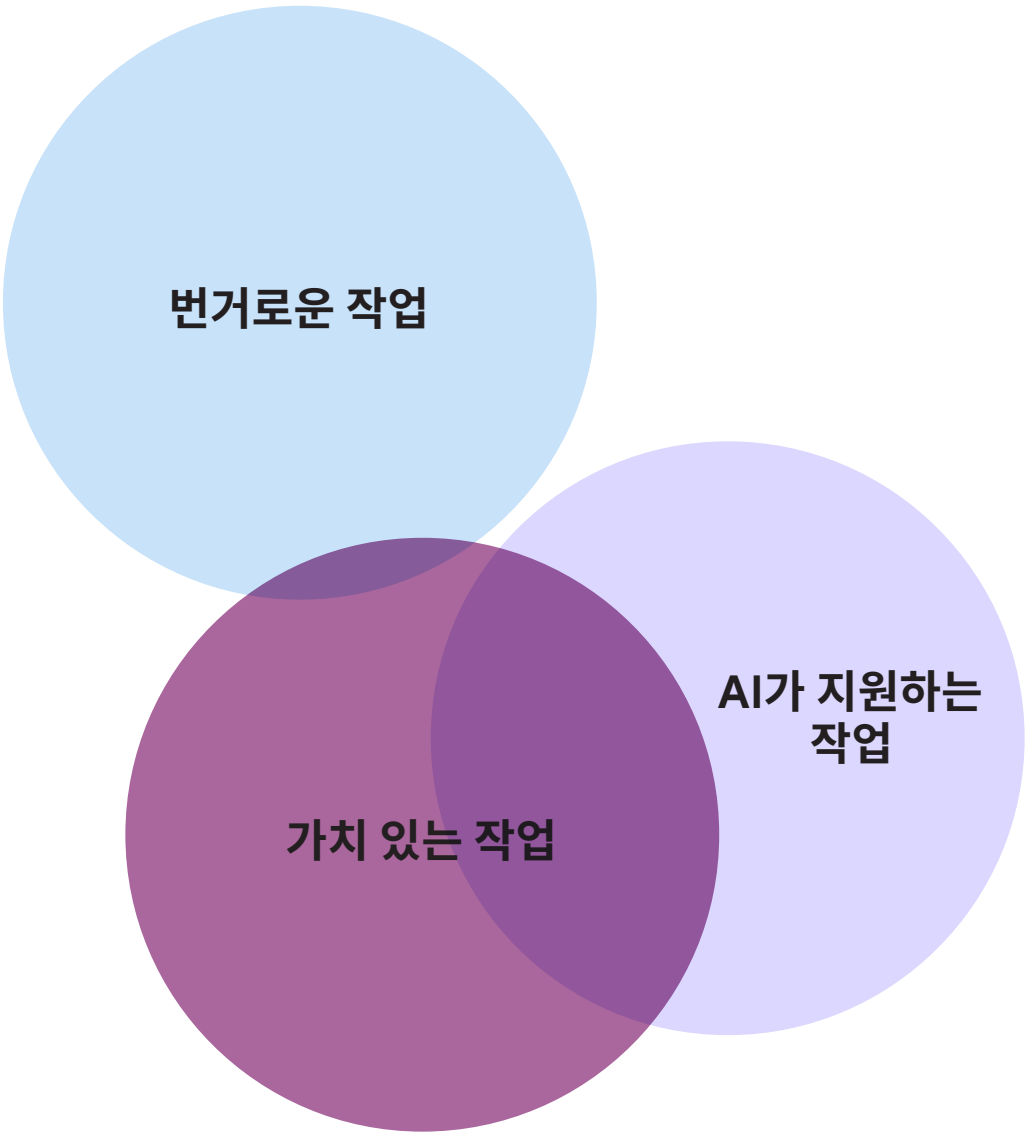


그림 8: 데이터가 아니라 'AI는 가치 있는 작업에는 도움을 주지만 반복 업무를 덜어주지는 않는다'는 가설을 시각화한 것입니다.

개발 워크플로에 대한 AI의 잠재적 영향

지난 섹션에서는 개인에 초점을 맞춘 결과를 살펴보았습니다. 다음 결과에서는 프로세스, 코드베이스, 팀 조정을 살펴보는 데 초점을 맞춥니다. 다음은 측정한 결과 목록입니다.

코드 복잡성	코드의 복잡성과 정교함이 생산성을 저해하는 정도입니다.
기술 부채	지난 6개월 동안 주요 애플리케이션 또는 서비스 내의 기존 기술 부채가 생산성을 저해한 정도입니다.
코드 검토 속도	기본 애플리케이션 또는 서비스에 대한 코드 검토를 완료하는 데 필요한 평균 시간입니다.
승인 속도	코드 변경을 제안하고 기본 애플리케이션 또는 서비스에서 프로덕션 사용 승인을 받기까지 걸리는 일반적인 시간입니다.
교차 기능 팀(XFN) 조정	"지난 3개월 동안 부서 간 팀원들과 효과적으로 협업할 수 있었습니다." 라는 진술에 대한 동의 수준입니다.
코드 품질	지난 6개월 동안 기본 서비스 또는 애플리케이션의 기반이 되는 코드 품질에 대한 만족 또는 불만족 수준입니다.
문서 품질	신뢰성, 검색성, 최신성, 지원 제공 능력 측면에서 내부 문서(매뉴얼, 리드미, 코드 주석)에 대한 인식입니다.

이전과 마찬가지로, 목표는 이러한 측면이 AI 도입에 따라 달라지는지 이해하는 것입니다. 그림 9는 AI 도입률 25% 증가와 관련하여 이러한 결과의 변화에 대한 최선의 추정치를 시각화한 것입니다.

전반적으로 이 패턴은 AI에 대한 매우 설득력 있는 설명을 제시합니다. 이 섹션의 실제 결과는 다음과 같습니다.

AI 도입이 25% 증가하면 다음과 같은 이점이 있습니다.

문서 품질 7.5% 향상

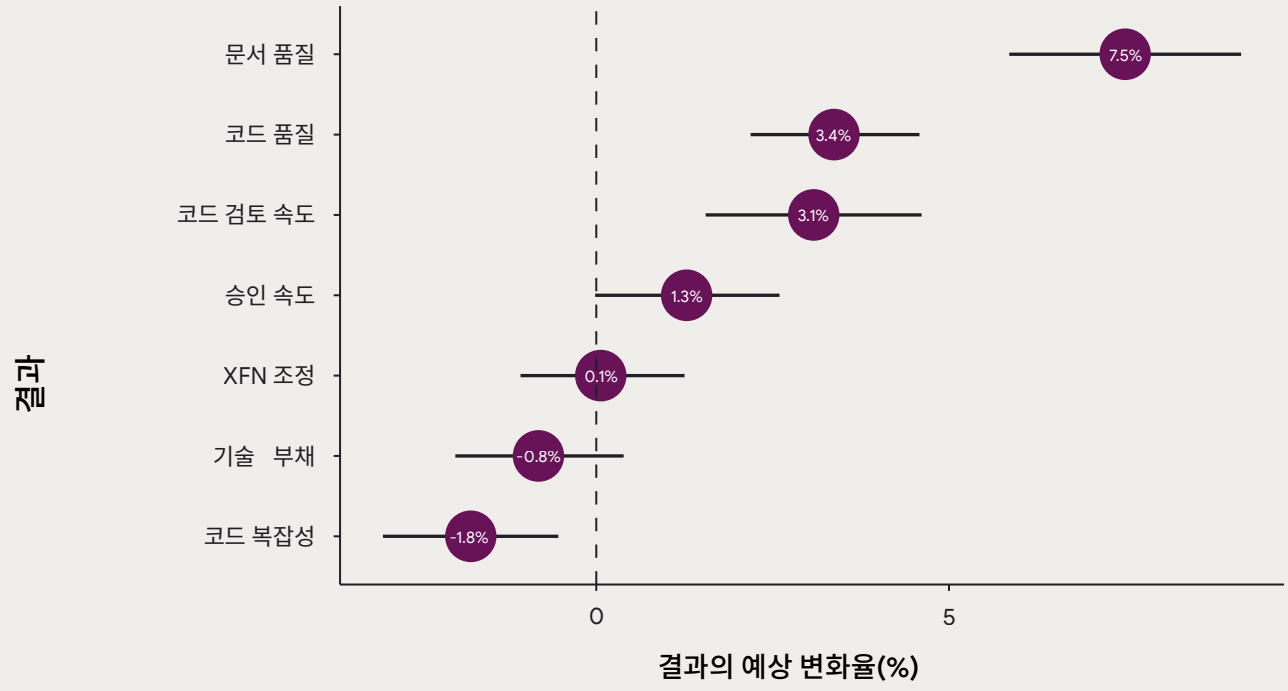
코드 품질 3.4% 향상

코드 검토 속도 3.1% 증가

승인 속도 1.3% 증가

코드 복잡성 1.8% 감소

AI 도입률이 25% 증가하면…



포인트 = 예상값
오차 막대 = 89% 불확실성 구간
그림 9: AI 도입이 조직에 미치는 영향

'인공지능: 도입 및 태도' 장에 제시된 데이터에 따르면 AI의 가장 일반적인 용도는 코드 작성입니다. 응답자의 67%는 AI가 코드 개선에 도움이 된다고 답했습니다. 여기서 이러한 정서를 더욱 확인할 수 있습니다. AI는 코드 품질을 개선하고 코드 복잡성을 감소시키는 것으로 보입니다(그림 9). 오래된 코드의 잠재적인 리팩터링과 결합하면 고품질의 AI 생성 코드가 전반적으로 더 나은 코드베이스로 이어질 수 있습니다. 이 코드베이스는 사람들이 AI를 사용하여 생성하는 양질의 문서에 더 잘 액세스함으로써 더욱 개선될 수 있습니다(인공지능: 도입 및 태도 참조).

더 좋은 코드는 더 쉽게 검토하고 승인할 수 있습니다. AI 지원 코드 검토와 결합하면 더 빠르게 검토하고 승인할 수 있으며, 이는 데이터에서 명확하게 드러난 패턴입니다(그림 9).

물론 코드 검토 및 승인이 빠르다고 해서 코드 검토 프로세스 및 승인 절차가 더 철저하고 더 나은 것은 아닙니다. 프로세스 지원을 위해 AI에 과도하게 의존하거나 AI가 생성한 코드를 과신하면서 속도를 높이고 있는 것일지도 모릅니다. 이 결과는 그림 9의 패턴과 상충되는 것은 아니지만, 그렇다고 해서 분명한 결론도 아닙니다.

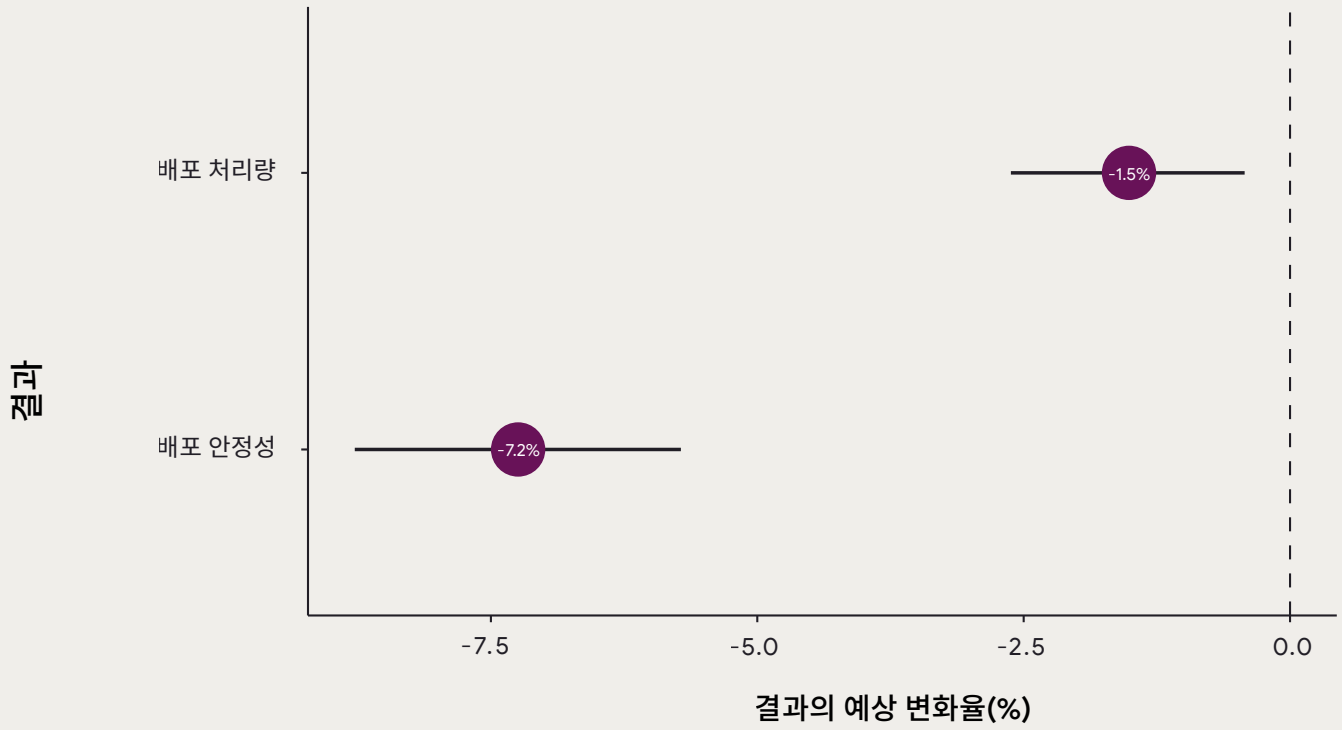
또한, AI가 코드를 생성하기 때문에 코드의 품질과 문서의 품질이 향상되는 것인지, 아니면 저품질 코드와 문서로 간주되던 것에서 가치 있는 정보를 얻을 수 있는 능력이 AI를 통해 향상되었는지는 분명하지 않습니다. 만약 AI의 강력한 성능으로 인해 양질의 코드와 문서라고 생각하는 것의 기준점이 AI를 사용할 때 조금만 낮아진다면 어떨까요? 이러한 패턴을 이해하는 이 2가지 방법은 상호 배타적인 해석이 아니며, 2가지 모두 이러한 패턴에 기여할 수 있습니다.

이러한 패턴에서 분명히 알 수 있는 것은 AI가 사람들이 의존하는 문서와 작업하는 코드베이스에서 더 많은 것을 얻을 수 있도록 도와준다는 것입니다. 또한 AI는 코드 검토 및 승인 프로세스에서 비용이 많이 드는 병목 현상을 줄이는 데 도움이 됩니다. AI가 정확히 어떻게 이러한 작업을 수행하는지, 이러한 이점이 소프트웨어 배포 개선과 같은 추가적인 다운스트림 이점으로 이어지는지 여부는 명확하지 않습니다.

배포 실적을 저해하는 AI

지난 몇 년 동안 소프트웨어 배포 처리량과 소프트웨어 배포 안정성 지표가 서로 어느 정도 독립성을 보이기 시작했다는 것을 확인했습니다. 처리량과 안정성 사이의 기존 상관관계는 지속되어 왔지만, 새로운 증거에 따르면 이러한 요소는 별도의 고려가 필요할 만큼 충분히 독립적으로 작동합니다.

AI 도입률이 25% 증가하면…



포인트 = 예상값
오차 막대 = 89% 불확실성 구간
그림 10: AI 도입이 배포 처리량 및 안정성에 미치는 영향

기대와는 달리, 연구 결과 AI 도입이 소프트웨어 배포 실적에 부정적인 영향을 미치고 있는 것으로 나타났습니다. 배포 처리량에 미치는 영향은 적지만 부정적일 가능성이 높습니다(AI 도입률이 25% 증가할 때마다 1.5% 감소하는 것으로 추정됨). 배포 안정성에 미치는 부정적인 영향은 더 큽니다 (AI 도입이 25% 증가할 때마다 7.2% 감소하는 것으로 추정됨). 이 데이터는 그림 10에서 확인할 수 있습니다.

지금까지의 연구에 따르면 문서 품질, 코드 품질, 코드 검토 속도, 승인 속도, 코드 복잡성 감소 등 소프트웨어 개발 프로세스의 개선이 소프트웨어 배포의 개선으로 이어진다는 사실이 밝혀졌습니다. 그래서 AI가 이러한 프로세스 지표를 개선하는 동시에 배포 처리량과 안정성이라는 실적 측정값을 떨어뜨리는 것처럼 보이는 것을 확인하고 놀랐습니다.

지난 몇 년간의 연구 결과를 바탕으로, 응답자의 생산성과 코드 생성 속도 측면에서 AI가 가져온 근본적인 패러다임의 변화로 인해 현장에서 DORA의 가장 기본적인 원칙 중 하나인 소규모 일괄 처리의 중요성을 간과했을 수 있다는 가설을 세웠습니다. 즉, AI를 사용하면 응답자가 같은 시간에 훨씬 더 많은 양의 코드를 작성할 수 있기 때문에 변경 목록의 규모가 커질 수도 있고, 실제로 커질 가능성이 높습니다. DORA는 큰 변화는 더 천천히 일어나고 불안정성을 초래하기 쉽다는 것을 일관되게 보여줬습니다.

데이터를 종합적으로 고려하면 개발 프로세스를 개선한다고 해서 소프트웨어 배포가 자동으로 개선되는 것은 아니며, 적어도 소규모 일괄 처리와 강력한 테스트 메커니즘과 같은 성공적인 소프트웨어 배포를 위한 기본사항을 제대로 준수하지 않는다면 더욱 그러합니다.

AI가 높은 소프트웨어 배포 실적을 위한 조건을 형성하는 여러 중요한 개인 및 조직적 요인에 미치는 유익한 영향은 긍정적으로 볼 수 있는 이유입니다. 하지만 AI가 만병통치약은 아닙니다.

실적이 우수한 팀과 조직은 AI를 사용하지만 제품은 그 혜택을 누리지 못하는 것 같습니다.

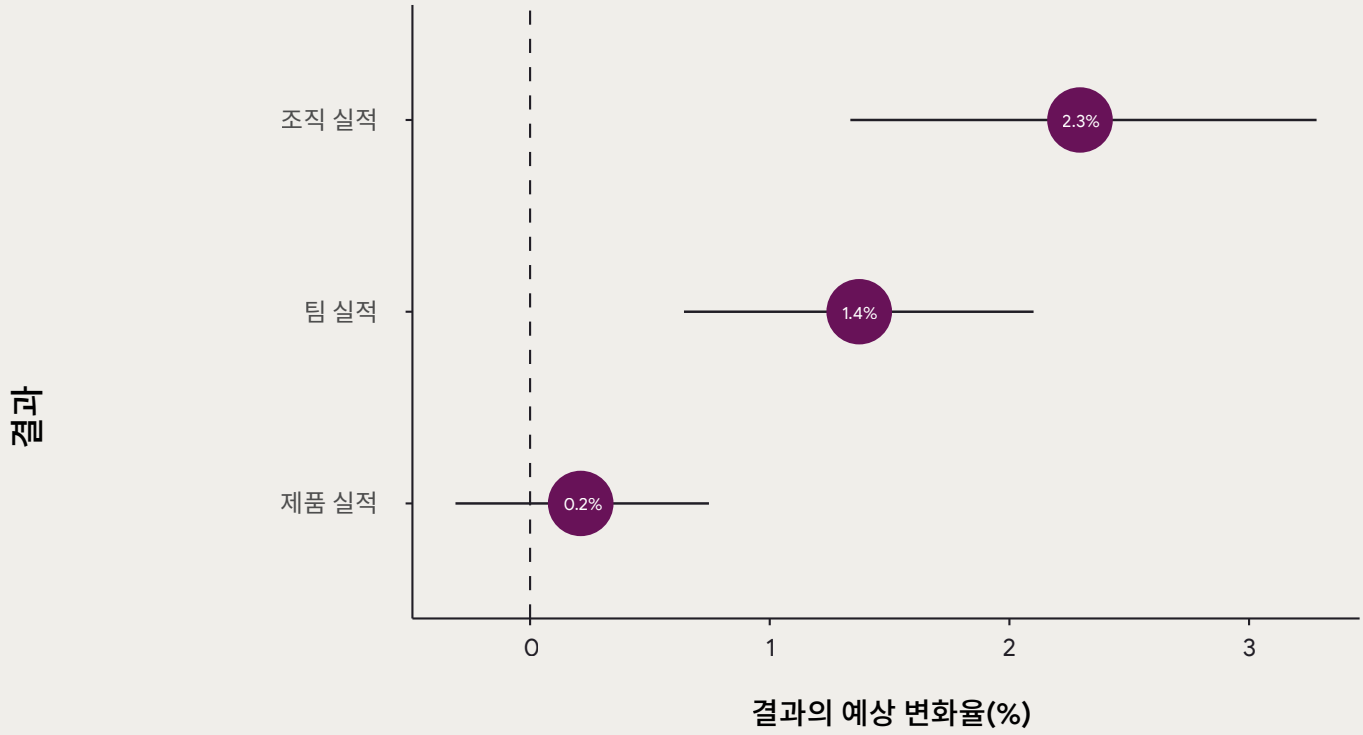
AI와 대부분의 다운스트림 결과물과의 관계는 다음과 같습니다.

조직 실적	조직의 전반적인 실적, 수익성, 시장점유율, 총 고객 수, 운영 효율성, 고객 만족도, 제품/서비스 품질, 목표 달성 능력 등을 고려한 요소 점수입니다.
팀 실적	팀의 협업, 혁신, 효율적 업무, 서로 의지하고 적응하는 능력을 설명하는 요소 점수입니다.
제품 실적	제품의 사용성, 기능, 가치, 가용성, 성능(예: 지연 시간) 및 보안을 설명하는 요소 점수입니다.

이러한 결과를 AI를 도입하는 개인과 연관 짓는 것은 어렵고 복잡합니다. 때로는 오늘 점심으로 먹은 음식이 올해 조직 성과에 미치는 영향을 분석하는 것처럼 느껴질 수도 있습니다.

미시적 수준(예: 개인)에서 거시적 수준(예: 조직)으로 넘어가는 데에는 논리가 있습니다. 방법론 장에서 이러한 추론적 도약에 대해 논의합니다. 지금은 연관성만 살펴보겠습니다.

AI 도입률이 25% 증가하면…



포인트 = 예상값
오차 막대 = 89% 불확실성 구간
그림 11. AI 도입이 조직, 팀, 제품 실적에 미치는 영향

조직 수준의 실적(AI 도입이 25% 증가할 때마다 약 2.3% 증가)과 팀 수준의 실적(AI 도입이 25% 증가할 때마다 약 1.4% 증가)은 AI 도입으로 인해 개선되는 것으로 보입니다(그림 11). 그러나 제품 실적은 AI 도입과 뚜렷한 연관성이 없는 것으로 보입니다. 이제 이러한 효과의 근본적인 원인을 이해하는 단계로 넘어갈 수 있습니다.

강력한 팀 및 조직 실적에 기여하는 요인은 제품 실적에 영향을 미치는 요인과는 다르다는 가설을 세웠습니다.

팀과 조직은 커뮤니케이션, 지식 공유, 의사 결정, 건강한 문화에 많은 부분을 의존합니다. AI는 이러한 영역에서 일부 병목 현상을 완화하여 팀과 조직에 긍정적인 영향을 미칠 수 있습니다.

하지만 제품의 성공에는 추가적인 요소가 작용할 수 있습니다. 좋은 제품에는 분명 실적이 우수한 팀 및 조직과 유사한 근본적인 원인이 있지만, 개발 워크플로 및 소프트웨어 배포와 더 밀접하고 직접적인 관련이 있을 수 있으며, 이 둘은 AI 도입 후에도 여전히 안정화되는 중일 수 있습니다.

기술적 측면의 고유한 중요성은 좋은 제품의 근간을 이루는 부분이지만, 훌륭한 제품의 근간에는 예술과 공감이라는 측면도 있습니다. 계산으로 모든 문제를 해결할 수 있다고 생각하는 사람들에게는 믿기 어려운 부분일 수 있지만, 창의성이나 사용자 환경 디자인과 같은 특정한 제품 개발 요소는 여전히 (또는 영원히) 인간의 직관과 전문 지식에 크게 의존해야 할 수도 있습니다.

조직, 팀, 제품 실적은 분명 서로 연관되어 있다는 것은 부인할 수 없는 사실입니다. 이변량 상관관계(Pearson)를 살펴보면 제품 실적은 팀 실적($r = 0.56$, 95% 신뢰 구간 = 0.51~0.60) 및 조직 실적($r = 0.47$, 95% 신뢰 구간 = 0.41~0.53) 모두와 중간 정도의 상관관계가 있는 것으로 나타났습니다.

이러한 결과는 서로 영향을 주고받으며 명확한 상호 종속성을 만들어냅니다. 실적이 우수한 팀은 더 좋은 제품을 개발하는 경향이 있지만, 수준 이하의 제품을 상속하면 성공에 방해가 될 수 있습니다. 마찬가지로 실적이 우수한 조직은 리소스와 프로세스를 통해 실적이 우수한 팀을 양성하지만, 조직의 어려움으로 인해 팀 실적이 저하될 수 있습니다. 따라서 AI 도입이 팀과 조직에 상당한 이점을 가져다준다면 제품에도 도움이 될 것으로 기대하는 것이 합리적입니다.

AI의 도입은 이제 막 시작되었습니다. AI의 본질적인 영향력이나 효과적인 활용과 관련된 학습 곡선으로 인해 일부 이점과 단점이 실현되기까지 시간이 걸릴 수 있습니다.

제품 혁신과 개발을 위한 AI의 잠재력을 완전히 실현하기 전에 조직과 팀에 AI가 어떻게 도움이 될 수 있는지 파악하고 있다는 이야기일 수도 있습니다. 그림 12는 이러한 상황이 어떻게 전개될 수 있는지 시각적으로 보여줍니다.

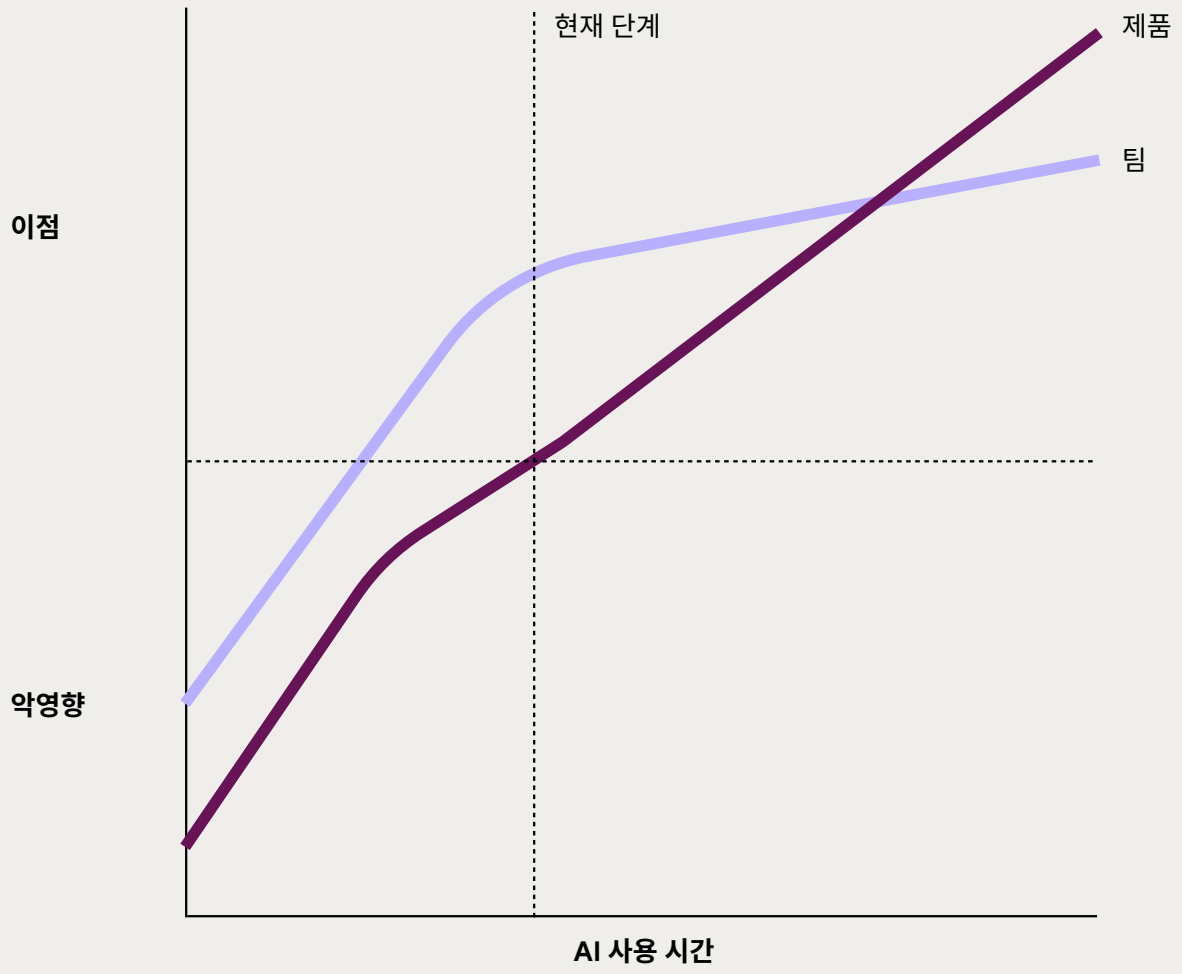


그림 12: 다양한 학습 곡선. 설명 목적으로 추상화한 것이며, 실제 데이터에서 도출한 것이 아닙니다.

이제 어떻게 해야 할까요?

현재 개인, 팀, 조직에 도움을 줄 수 있는 AI의 잠재력을 이해하고 싶었습니다. 현재 나타나고 있는 패턴은 이것이 모두 허풍이 아니라 실제로 무언가 일어나고 있다는 것을 강조합니다.

AI 도입의 필요성에 대한 명확한 증거가 있습니다. 하지만 잠재적인 장애물과 성장통도 있을 수 있으며 AI가 여러 방식으로 해로운 영향을 미칠 수 있다는 점도 분명합니다.

대규모로 AI를 도입하는 것은 재생을 누르는 것만큼 쉽지 않을 수 있습니다. 측정 가능하고 투명하며 적응 가능한 전략은 상당한 이점을 제공할 수 있습니다. 이 전략은 리더, 팀, 조직, 연구원, AI를 개발하는 사람들이 공동으로 발전해 나가야 합니다.

임원진과 조직은 직원을 가장 잘 지원할 수 있는 영역에서 우선순위를 정하여 AI를 도입할 방법을 찾아야 합니다.

다음은 AI 도입 전략의 방향을 잡는 방법에 대한 몇 가지 아이디어입니다.

명확한 AI 미션과 정책을 규정하여 조직과 팀의 역량을 강화합니다.

직원들에게 회사의 AI 미션, 목표, AI 도입 계획에 대한 투명한 정보를 제공합니다. 허용된 코드 배치 및 사용 가능한 도구와 같은 절차적 문제를 해결하는 등 중요한 비전과 구체적인 정책을 모두 명시함으로써 불안감을 완화하고 모두가 더 가치 있고 만족스럽고 창의적인 작업에 집중할 수 있도록 돕는 수단으로 AI를 활용할 수 있습니다.

AI를 지속적으로 학습시키고 실험하는 문화를 조성합니다.

개인과 팀이 유익한 사용 사례를 발견할 수 있도록 시간을 할애하고 스스로 진로를 계획할 수 있는 자율성을 부여하여 AI 도구를 지속적으로 탐색할 수 있는 환경을 조성합니다. 샌드박스 또는 저위험 환경에서의 실제 경험을 통해 AI 기술에 대한 신뢰를 구축합니다. 강력한 테스트 자동화를 개발하는 데 집중하여 위험을 더욱 완화합니다. 단순한 도입이 아닌 의미 있는 다운스트림 영향, 즉 직원의 성장을 돕고, 제품에 의존하는 사용자에게 혜택을 제공하며, 팀의 잠재력을 실현하는 방법을 기준으로 AI를 평가하는 측정 프레임워크를 구현합니다.

경쟁력을 확보하기 위해 AI의 장단점을 파악하고 활용합니다.

가치 있는 작업에 소비하는 시간 감소, AI에 대한 과도한 의존, 한 영역에서 얻은 이점이 다른 영역에서 문제로 이어질 가능성, 소프트웨어 배포 안정성과 처리량에 미치는 영향 등 잠재적인 단점을 인식함으로써 함정을 피하고 조직과 팀에서 AI의 궤적을 긍정적으로 형성할 기회를 파악할 수 있습니다. AI가 어떻게 유익할 수 있는지뿐만 아니라 어떻게 해로울 수 있는지를 이해하면 학습 곡선을 앞당기고, 탐색을 지원하고, 학습한 내용을 행동으로 전환하여 실질적인 경쟁력을 확보할 수 있습니다.

기대할 것도 많고 배워야 할 것도 많다는 것은 분명합니다. DORA는 지난 10년 동안 그래왔던 것처럼 앞으로도 정직하고 정확하며 유용한 관점을 제공하기 위해 최선을 다할 것입니다.

1. https://www.goldmansachs.com/insights/top-of-mind/gen-ai-too-much-spend-too-little-benefit
2. https://www.goldmansachs.com/insights/articles/AI-poised-to-drive-160-increase-in-power-demand
3. https://www.washington.edu/news/2023/07/27/how-much-energy-does-chatgpt-use/
4. https://www.gatesnotes.com/The-Age-of-AI-Has-Begun
5. https://www.businessinsider.com/ai-chatgpt-homework-cheating-machine-sam-altman-openai-2024-8
6. https://www.safe.ai/work/statement-on-ai-risk
7. https://github.blog/news-insights/research/research-quantifying-github-copilots-impact-on-developer-productivity-and-happiness/
8. https://www.gitclear.com/coding_on_copilot_data_shows_ais_downward_pressure_on_code_quality
9. https://www.nytimes.com/2024/04/15/technology/ai-models-measurement.html
10. https://dora.dev/capabilities
11. 이는 특별한 접근 방식은 아니지만 이 분야에서는 다소 특별한 접근 방식이라는 점을 분명히 합니다.
12. (P[N]), 예를 들어 (P1)은 인터뷰 참가자의 가명을 나타냅니다.

플랫폼 엔지니어링

소개

플랫폼 엔지니어링은 업계 전반에서 관심과 모멘텀을 얻고 있는 떠오르는 엔지니어링 분야입니다. Spotify, Netflix와 같은 업계 리더와 *팀 토폴로지*[1]와 같은 책이 대중의 관심을 불러일으켰습니다.

플랫폼 엔지니어링은 엔지니어가 여러 팀 간의 사회적 상호작용과 자동화, 셀프 서비스, 프로세스의 반복성과 같은 기술적 측면의 교차점에 초점을 맞추는 사회기술 분야입니다. 플랫폼 엔지니어링의 개념은 DORA를 포함하여 수년 동안 연구되어 왔습니다.

일반적으로 Google의 연구는 외부 사용자에게 소프트웨어를 제공하는 방법에 초점을 맞추는 반면, 플랫폼 팀의 결과물은 일반적으로 소프트웨어 개발 및 운영 수명 주기를 지원하도록 설계된 내부 중심의 API, 도구, 서비스 세트입니다.

플랫폼 엔지니어링에서는 플랫폼 사용자가 애플리케이션을 배포하고 운영하는 데 필요한 리소스와 상호작용할 때 사용하는 고도로 자동화된 셀프 서비스 워크플로인 최적의 경로를 구축하여 개발자 경험을 개선하는 데 많은 에너지와 집중력을 쏟고 있습니다. 목적은 개발자가 코드에만 신경 쓰면 되도록 소프트웨어 빌드 및 배포의 복잡성을 없애는 것입니다.

최적의 경로를 통해 자동화된 작업의 몇 가지 예로는 새로운 애플리케이션 프로비저닝, 데이터베이스 프로비저닝, 스키마 관리, 테스트 실행, 빌드 및 배포 인프라 프로비저닝, DNS 관리가 있습니다.

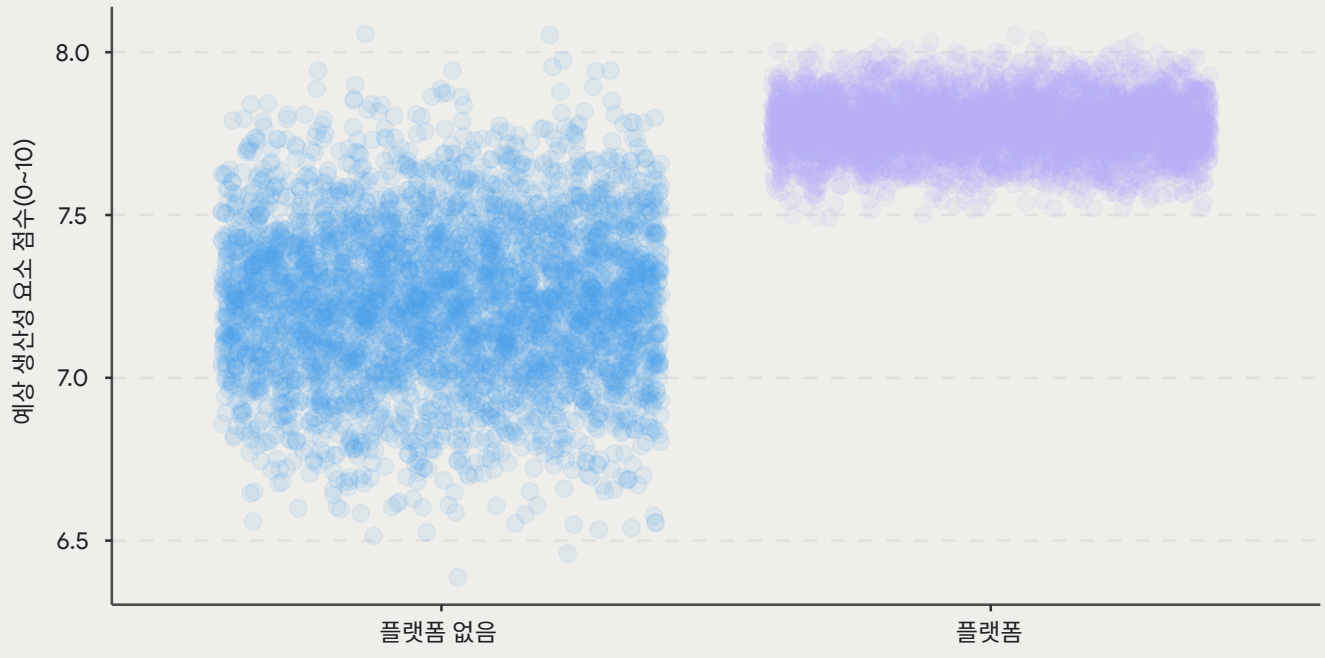
공유 시스템으로 기능을 아래로 이동('시프트 다운'이라고도 함)하는 등의 플랫폼 엔지니어링 개념[2]은 '빌드하면 실행한다'는 접근 방식에 반하는 것처럼 보일 수 있습니다. 하지만 플랫폼 엔지니어링을 조직 전체에 걸쳐 이러한 관행의 도입을 확장하는 방법으로 생각하는 이유는 플랫폼에 기능이 추가되면 팀은 기본적으로 플랫폼 도입을 통해 해당 기능을 무료로 얻을 수 있기 때문입니다.

예를 들어 플랫폼에는 단위 테스트를 실행하고 결과를 개발팀에 직접 보고하는 기능이 있지만 해당 팀이 테스트 실행 환경을 구축하고 관리할 필요가 없는 경우, 지속적 통합 플랫폼 기능을 사용하면 팀은 고품질 테스트 작성에 집중할 수 있습니다. 이 예에서 지속적 통합 기능은 대규모 조직 전체에 걸쳐 확장할 수 있으며 여러 팀이 지속적인 테스트[3] 및 테스트 자동화[4]를 통해 역량을 쉽게 개선할 수 있도록 지원합니다.

성공의 핵심 요소는 사용자 중심 전략(내부 개발자 플랫폼의 맥락에서 사용자는 곧 개발자임), 개발자 독립성, 제품 사고방식으로 플랫폼 엔지니어링에 접근하는 것입니다. 사용자 중심 전략이 올해와 지난해에 조직 실적 개선의 핵심 요소로 확인되었다는 점을 고려하면 이는 그리 놀라운 일이 아닙니다[5]. 사용자 중심의 접근 방식이 없다면 플랫폼은 도움이 되기보다는 오히려 방해될 것입니다.

올해 보고서에서는 플랫폼과 소프트웨어 배포 및 운영 실적 간의 관계를 테스트하고자 했습니다. 몇 가지 긍정적인 결과를 확인했습니다. 내부 개발자 플랫폼 사용자의 경우 개인 생산성은 8%, 팀 실적은 10% 더 높았습니다. 또한 플랫폼을 사용하면 조직의 소프트웨어 배포 및 운영 실적이 6% 향상됩니다. 이러한 이점이 있지만 몇 가지 단점도 있습니다. 처리량과 변경 안정성은 각각 8%와 14% 감소했는데, 이는 놀라운 결과였습니다.

다음 섹션에서는 이번 설문조사에서 밝혀진 수치와 뉘앙스, 몇 가지 놀라운 데이터에 대해 자세히 살펴보겠습니다. 플랫폼 엔지니어링 이니셔티브를 이제 막 시작했든 수년 동안 진행했든 상관없이, 주요 결과를 적용하면 플랫폼의 성공률을 높이는 데 도움이 될 수 있습니다.



각 점은 가장 타당한 평균 생산성 점수에 대한 8,000개의 추정치 중 하나입니다.
그림 13: 내부 개발자 플랫폼을 사용하거나 사용하지 않을 때의 개인별 생산성 요소

플랫폼 엔지니어링의 가능성

내부 개발자 플랫폼은 관행에서 얻을 수 있는 잠재적인 효율성과 생산성 향상으로 인해 소프트웨어 개발자 및 IT 업계의 많은 분야에서 관심을 받고 있습니다. 올해 설문조사에서는 내부 개발자 플랫폼의 정의를 상당히 폭넓게 설정했으며[6], 응답자의 89%가 내부 개발자 플랫폼을 사용하고 있는 것으로 나타났습니다. 상호작용 모델은 해당 집단 전체에서 매우 다양합니다.

이러한 데이터 포인트는 플랫폼 엔지니어링에 대한 업계의 광범위한 관심 수준과 이 분야의 새로운 특성에 부합합니다.

전반적으로 플랫폼의 영향은 긍정적이었으며, 내부 개발자 플랫폼을 사용할 때 개인 생산성은 8%, 팀 생산성은 10% 더 높았습니다.

생산성 외에도 플랫폼을 사용하면 조직의 전반적인 성과가 6% 향상되는 것으로 나타났습니다. 전반적으로 조직은 플랫폼 덕분에 소프트웨어를 신속하게 배포하고, 사용자 니즈를 충족하며, 비즈니스 가치를 창출할 수 있습니다.

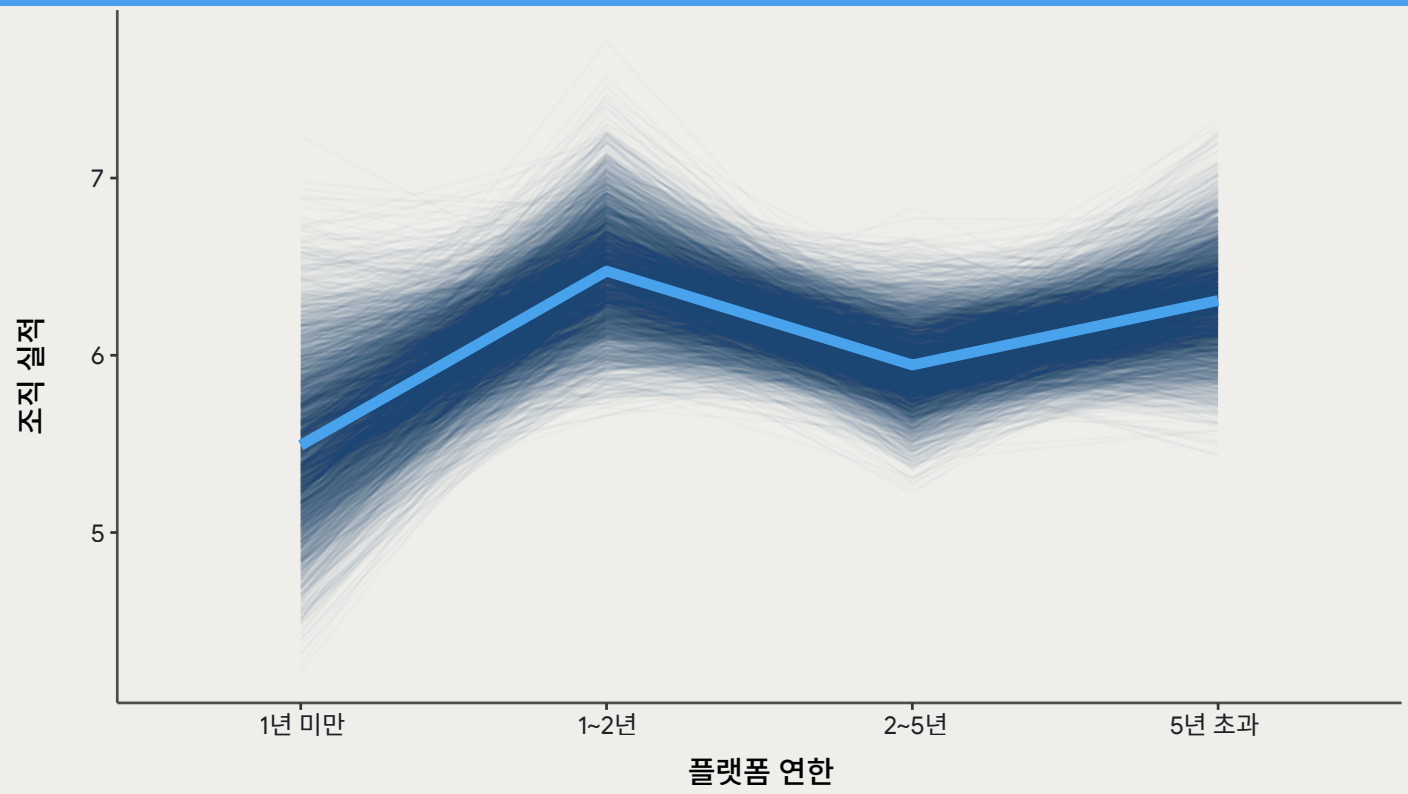
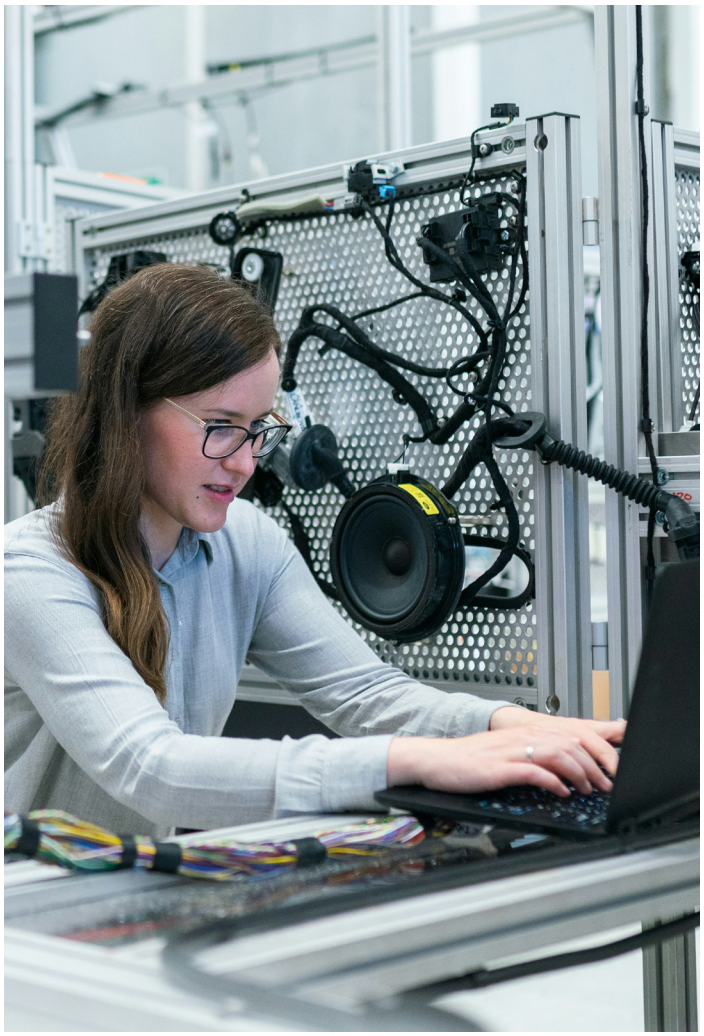


그림 14: 내부 개발자 플랫폼을 사용할 때의 조직 실적 변화와 플랫폼의 도입 시기 비교

생산성과 함께 플랫폼의 연한을 고려할 때, 플랫폼 엔지니어링 이니셔티브가 시작될 때 초기에는 실적이 향상되다가 플랫폼이 노후화되고 성숙해지면서 실적이 감소하고 회복되는 것을 볼 수 있습니다. 이러한 패턴은 혁신 이니셔티브에서 흔히 볼 수 있는 것으로, 초기에는 이익을 얻지만 실현된 후에는 난관에 부딪히게 됩니다.

장기적으로는 소프트웨어 배포 및 운영 프로세스에서 내부 개발자 플랫폼의 역할에 대한 전반적인 잠재력을 보여주면서 생산성 향상이 유지됩니다.



주요 결과 - 개발자 독립성이 미치는 영향

내부 개발자 플랫폼을 사용하여 소프트웨어를 배포할 때 개발자 독립성은 개인과 팀 수준 모두에서 생산성 수준에 상당한 영향을 미쳤습니다. 개발자 독립성이란 '개발자가 지원팀에 의존하지 않고 전체 애플리케이션 수명 주기 동안 자신의 작업을 수행할 수 있는 능력'으로 정의됩니다.

팀과 개인 수준 모두에서 플랫폼 사용자가 지원팀의 개입 없이도 작업을 완료할 수 있을 때 생산성이 5% 향상되는 것을 확인했습니다. 이러한 연구 결과는 플랫폼 엔지니어링의 핵심 원칙 중 하나인 셀프 서비스 워크플로를 사용하는 데 초점을 맞추고 있습니다.

플랫폼 팀에 있어 이는 플랫폼 엔지니어링 프로세스의 중요한 부분인 사용자 피드백 수집을 의미하므로 매우 중요합니다. 설문조사 응답에서는 어떤 형태의 피드백이 가장 효과적인지는 나타나지 않았지만, 비공식적인 대화와 문제 추적기가 가장 일반적인 방법이며, 지속적인 공동 개발, 설문조사, 원격 분석, 인터뷰가 그 뒤를 따르고 있습니다.

이러한 모든 방법은 사용자가 독립적으로 작업을 완료할 수 있는지 파악하는 데 효과적일 수 있습니다. 또한 설문조사 데이터에 따르면 플랫폼에 대한 피드백을 수집하지 않는 것은 부정적인 영향을 미치는 것으로 나타났습니다.

부가적 연구 결과 - 플랫폼 전담팀의 영향력

흥미롭게도 플랫폼 전담팀이 생산성에 미치는 영향은 개인에게는 미미한 것으로 나타났습니다. 하지만 팀 수준의 생산성은 6% 향상되었습니다. 이 결과는 플랫폼 전담팀이 개인에게는 유용한 정도지만 팀 전체적으로는 더 큰 영향력을 미친는 것을 시사하는 것으로, 그 영향력이 균등하지 않다는 점에서 놀랍습니다.

팀에는 책임과 기술이 다른 여러 명의 개발자가 있기 때문에 개별 엔지니어에 비해 당연히 더 다양한 작업을 수행하게 됩니다. 전담 플랫폼 엔지니어링팀을 보유하면 팀이 담당하는 업무의 다양성을 플랫폼에서 더 잘 지원할 수 있습니다.

전반적으로는 내부 개발자 플랫폼이 생산성에 긍정적인 영향을 미치는 것으로 나타났습니다.

주요 요소는 다음과 같습니다.

셀프 서비스 및 자율적으로 완료할 수 있는 워크플로를 통해 개발자의 독립성을 지원하는 사용자 중심 접근 방식입니다. 플랫폼의 맥락에서 사용자는 내부 엔지니어링 및 개발팀이라는 점을 기억하시기 바랍니다.

다른 혁신과 마찬가지로 플랫폼 엔지니어링에도 'J 곡선'이 적용되므로 지속적인 개선을 통해 생산성 향상은 안정화될 것입니다.

예상치 못한 단점

플랫폼 엔지니어링은 팀과 개인의 생산성이 향상되고 조직 실적이 개선된다는 측면에서 분명한 이점을 제공하지만, 예상치 못한 단점도 있었습니다. 또한 처리량과 변경 안정성이 감소한 것으로 나타났습니다.

뜻밖에도 변경사항 불안정성과 번아웃 사이에 매우 흥미로운 연관성을 발견했습니다.

처리량

처리량의 경우, 플랫폼을 사용하지 않는 사용자보다 약 8% 감소한 것으로 나타났습니다. 근본적인 원인이 무엇인지에 대해 가설을 세웠습니다.

첫째, 변경사항이 프로덕션에 배포되기 전에 통과해야 하는 기계가 추가되면 전체 변경사항 처리량이 감소합니다. 일반적으로 내부 개발자 플랫폼을 사용하여 소프트웨어를 빌드하고 배포하는 경우 일반적으로 시스템 간 그리고 암시적으로 팀 간 '핸드오프' 횟수가 증가합니다.

예를 들어 코드가 소스 제어에 커밋되면 테스트, 보안 검사, 배포, 모니터링을 위해 여러 시스템에서 자동으로 선택됩니다.

이러한 각 핸드오프는 전체 프로세스에 시간을 투입할 기회이므로 처리량은 감소하지만 업무 처리 능력은 순증가합니다.

둘째, “전체 앱 수명 주기 동안 작업을 수행하기 위해 플랫폼을 독점적으로 사용할 필요가 있다”고 답한 응답자의 경우 처리량이 6% 감소했습니다. 확실한 연관성은 아니지만 첫 번째 가설과도 관련이 있을 수 있습니다.

플랫폼의 존재로 인해 소프트웨어 개발 및 출시와 관련된 시스템과 도구가 증가한다면, 목적에 맞지 않을 수 있는 플랫폼을 사용해야 하거나 프로세스에서 자연스럽게 지연 시간이 늘어나는 것은 독점과 생산성 저하 사이의 관계를 설명할 수 있습니다.

이에 대응하기 위해서는 플랫폼 엔지니어링 이니셔티브에서 사용자 중심을 지향하고 사용자 독립성을 확보하는 것이 중요합니다.

변경사항 불안정성 및 번아웃

내부 개발자 플랫폼을 사용할 때 개발 및 운영 중인 애플리케이션의 변경사항의 안정성을 고려했을 때 변경 안정성이 놀랍게도 14% 감소한 것으로 나타났습니다. 이는 플랫폼을 사용할 때 변경 실패율과 재작업 비율이 많이 증가한다는 것을 나타냅니다.

더욱 흥미로운 점은 플랫폼과 결합된 불안정성이 더 높은 수준의 번아웃과 관련이 있다는 사실을 발견했다는 점입니다. 플랫폼이 번아웃을 유발한다는 것은 아니지만, 불안정성과 플랫폼의 조합은 특히 번아웃을 유발할 때 문제가 됩니다. 처리량 감소와 마찬가지로 번아웃의 변화도 왜 발생하는지 완전히 알 수는 없지만 몇 가지 가설이 있습니다.

첫째, 이 플랫폼을 통해 개발자와 팀은 변경사항이 잘못되었을 경우 신속하게 수정할 수 있다는 강한 확신을 가지고 변경사항을 내보낼 수 있습니다. 이 경우 플랫폼이 팀에 실험과 변경사항을 배포할 수 있는 권한을 부여함으로써 변경 실패와 재작업이 증가하므로 불안정성이 높다고 해서 반드시 나쁜 것은 아닙니다.

두 번째 아이디어는 플랫폼이 프로덕션에 대한 변경사항 및/또는 배포의 품질을 보장하는 데 효과적이지 않다는 것입니다.

또한 플랫폼이 애플리케이션에 포함된 모든 테스트를 실행하는 자동화된 테스트 기능을 제공할 수도 있습니다. 그러나 애플리케이션 팀은 품질보다 처리량을 우선시하고 테스트를 개선하지 않으면서 이러한 기능을 충분히 활용하지 못하고 있습니다. 두 시나리오 모두 실제로는 잘못된 변경사항이 프로세스에 적용되어 재작업이 발생하게 됩니다.

세 번째 가능성은 변경사항 불안정성과 번아웃 수준이 높은 팀이 안정성을 개선하고 번아웃을 줄이기 위해 플랫폼을 만드는 경향이 있다는 것입니다. 플랫폼 엔지니어링은 주로 번아웃을 줄이고 작은 변경사항을 일관되게 출시하는 능력을 개선하는 관행으로 간주되기 때문에 이는 의미가 있습니다. 이 가설에 따르면 플랫폼 엔지니어링은 번아웃과 변경사항 불안정성이 있는 조직에서 흔히 나타납니다.

처음 2가지 시나리오에서는 플랫폼에서 허용하는 재작업이 부담스러워 번아웃을 높일 수 있습니다. 특히 플랫폼이 잘못된 변경사항을 가능하게 하는 두 번째 시나리오는 번아웃에 더 큰 영향을 미칠 수 있지만, 두 시나리오 모두에서 팀이나 개인은 변경사항과 기능을 내보낼 수 있는 능력 때문에 여전히 생산적이라고 느낄 수 있습니다. 세 번째 시나리오에서는 변경사항 불안정성과 번아웃이 플랫폼 엔지니어링 이니셔티브를 예측하며, 플랫폼은 이러한 문제에 대한 해결책으로 간주됩니다.

장단점 균형 맞추기

플랫폼 엔지니어링이 만병통치약은 아니지만, 전반적인 소프트웨어 개발 및 운영 프로세스에 있어 강력한 분야가 될 수 있는 잠재력이 있습니다. 다른 분야와 마찬가지로 플랫폼 엔지니어링에도 장단점이 있습니다.

연구에 따르면 플랫폼 엔지니어링 이니셔티브를 시작할 때 장단점의 균형을 맞추기 위해 취할 수 있는 몇 가지 조치가 있습니다. 이렇게 하면 조직이 플랫폼 엔지니어링의 장점을 달성하는 동시에 잠재적인 단점을 모니터링하고 관리할 수 있습니다.

먼저, 개발자의 독립성과 셀프 서비스 기능을 지원하는 플랫폼 기능을 우선적으로 고려합니다. 이 경우 애플리케이션 수명 주기의 모든 측면에 플랫폼을 독점적으로 사용하도록 요구하는 것은 개발자의 독립성을 저해할 수 있으므로 균형을 고려해야 합니다.

플랫폼은 플랫폼 사용자가 플랫폼에서 제공하는 도구와 자동화에서 벗어날 수 있는 방법을 제공해야 독립성에 기여할 수 있지만, 이는 복잡성이라는 대가를 치르게 됩니다. 플랫폼 사용자와 적극적으로 협업하고 피드백을 수집하는 전담 플랫폼 팀을 통해 이러한 절충사항을 완화할 수 있습니다.

협업과 피드백은 플랫폼 이니셔티브의 사용자 중심 전략을 향상하고 플랫폼의 장기적인 성공에 기여할 것입니다. 데이터에서 보았듯이 피드백 수집에는 다양한 방법이 사용되므로 피드백 수집을 극대화하려면 2가지 이상의 접근 방식을 사용해야 합니다.

둘째, 애플리케이션 변경사항의 불안정성을 주의 깊게 모니터링하고 불안정성이 의도적인 것인지 아닌지 파악합니다. 플랫폼은 불안정성 측면에서 실험을 가능하게 하고, 생산성을 높이며, 대규모로 실적을 개선할 수 있는 잠재력이 있습니다.

그러나 이와 같은 불안정성은 불안정성과 번아웃이라는 대가를 치르게 할 수도 있으므로 플랫폼 엔지니어링 여정 전반에 걸쳐 신중하게 모니터링하고 고려해야 합니다. 그렇게 할 때는 불안정성에 대한 자신의 성향을 이해하는 것이 중요합니다. 사이트 안정성 엔지니어링(SRE)의 서비스 수준 목표(SLO)와 오류 예산을 사용하면 안전하게 실험을 지원하는 과정에서 플랫폼의 위험 허용 범위와 효과를 측정하는 데 도움이 됩니다.

내부 개발자 플랫폼은 개발자 경험을 상당히 강조하지만, 소프트웨어를 효과적으로 배포하고 운영하는 데 필요한 다른 팀(데이터베이스 관리자, 보안, 운영 등)도 많습니다.

플랫폼 엔지니어링 이니셔티브에서 모든 팀에 걸쳐 사용자 중심 전략을 강조하고 조직의 목표에 맞춰 지속적으로 개선하는 문화를 조성합니다.

이렇게 하면 소프트웨어와 비즈니스 가치를 제공하기 위해 노력하는 개인과 팀의 필요에 가장 잘 부합하도록 플랫폼의 기능, 서비스, API를 조정할 수 있습니다.

1. 매튜 스켈튼 및 마누엘 파이스. 2019년. Team Topologies: Organizing Business and Technology Teams for Fast Flow. IT Revolution Press. https://teamtopologies.com/
2. https://cloud.google.com/blog/products/application-development/richard-seroter-on-shifting-down-vs-shifting-left
3. https://dora.dev/capabilities/continuous-integration/
4. https://dora.dev/capabilities/test-automation/
5. https://dora.dev/research/2023/, https://dora.dev/research/2016/
6. https://dora.dev/research/2024/questions/#platform-engineering

개발자 환경

핵심 내용

소프트웨어는 저절로 빌드되지 않습니다. AI의 도움을 받더라도 사람들이 소프트웨어를 빌드하며, 직장에서의 경험은 성공적인 조직의 기본 구성요소입니다.

올해 보고서에서는 사용자의 니즈에 따라 개발자가 빌드해야 직원과 조직이 성공할 수 있다는 사실을 다시 한번 확인했습니다. 개발자가 사용자 중심적인 사고방식으로 소프트웨어를 빌드할 때 생산성이 향상되고 번아웃을 덜 경험하며 우수한 품질의 제품을 빌드할 가능성이 높아집니다.

궁극적으로 소프트웨어는 사용자를 위해 빌드되므로 개발자가 사용자 경험을 개선하는 소프트웨어를 빌드하는 데 집중할 수 있는 환경을 조성하는 것은 조직의 책임입니다. 또한 우선순위가 지속적으로 바뀌지 않는 안정적인 환경에서는 생산성이 작지만 의미 있게 증가하고 직원의 번아웃이 유의미하게 감소하는 것으로 나타났습니다.

환경적 요인은 개발된 제품의 품질과 이러한 제품을 빌드하는 개발자의 전반적인 경험에 상당한 영향을 미칩니다.

사용자를 최우선으로 생각하면 (거의) 모든 것이 제자리를 찾습니다.

개발자라는 직업이 멋진 직업이라고 생각합니다. 개발자는 기술 발전의 최전선에 있으며 사람들이 생활하고, 일하고, 세상과 소통하는 방식을 형성하는 데 기여하고 있습니다.

개발자의 업무는 근본적으로 소프트웨어와 애플리케이션의 사용자, 즉 사람과 연결되어 있습니다. 하지만 개발자는 기능과 혁신을 우선시하는 환경에서 작업하는 경우가 많습니다. 이러한 기능이 제품 사용자에게 가치를 제공하는지 여부를 파악하는 데 중점을 두지 않습니다.

여기에서는 최종 사용자를 우선시하는 소프트웨어 개발 접근 방식이 직원과 조직 모두에게 긍정적인 영향을 미친다는 강력한 증거를 제시합니다.

올해에는 개발자가 다음에 해당하는지 이해하는 데 초점을 맞춘 질문을 던졌습니다.

1. 사용자 피드백을 통합하여 기능을 재검토하고 우선순위를 재조정합니다.

2. 사용자가 특정 애플리케이션/서비스로 달성하고자 하는 목표를 파악합니다.

3. 사용자에게 집중하는 것이 비즈니스 성공의 핵심이라고 믿습니다.

4. 사용자 환경이 비즈니스의 최우선 과제라고 믿습니다.

연구 결과와 그 의미

데이터는 사용자의 필요와 과제를 기준으로 삼는 조직이 더 나은 제품을 만든다는 사실을 강력하게 시사합니다.

사용자에게 집중하면 생산성과 업무 만족도가 높아지는 동시에 번아웃의 위험도 줄어드는 것으로 나타났습니다.

중요한 점은 이러한 이점이 직원 개인을 넘어 조직 전체로 확대된다는 점입니다. 지난 몇 년 동안 성과가 우수한 조직이 소프트웨어를 빠르고 안정적으로 제공한다는 점을 강조해 왔습니다. 이는 소프트웨어 배포 성과가 성공의 필수 요건이라는 것을 의미합니다.

하지만 데이터에 따르면 성공에 이르는 또 다른 길이 있음을 알 수 있습니다.

개발자와 고용주, 일반적으로 조직은 소프트웨어 개발에 대한 사용자 중심적인 접근 방식을 수립할 수 있습니다.

조직이 사용자의 필요를 파악하고 이해하면 소프트웨어 배포의 안정성과 처리량은 제품 품질에 대한 요건이 아니라는 것을 알게 되었습니다. 사용자 환경이 최우선시되는 한 제품의 품질은 높아질 것입니다.

조직이 사용자에 초점을 맞추지 않고 사용자 피드백을 개발 프로세스에 통합하지 않는다면 안정적이고 빠른 배포에 집중하는 것만이 제품 품질을 개선하는 유일한 방법입니다(그림 15 참조).

일부 조직이 기능 개발과 기술 혁신에 집중해야 하는 경향을 잘 알고 있습니다. 액면 그대로 보면 이 접근 방식이 합리적입니다. 결국 개발자는 일반 사용자보다 기술의 세부사항을 훨씬 더 잘 알고 있습니다.

그러나 사용자 환경에 대한 가정을 기반으로 소프트웨어를 개발하면 개발자가 겉보기에는 화려하지만 거의 사용되지 않는 기능을 만들 가능성이 높아집니다[1].

조직과 직원이 사용자가 세상을 경험하는 방식을 이해하면 사용자의 실제 필요를 충족하는 기능을 구축할 가능성이 커집니다. 실제 사용자 니즈를 해결하면 해당 기능이 실제로 사용될 가능성이 높아집니다.

사용자를 위한 빌드에 집중하면 만족스러운 제품을 만들 수 있습니다.

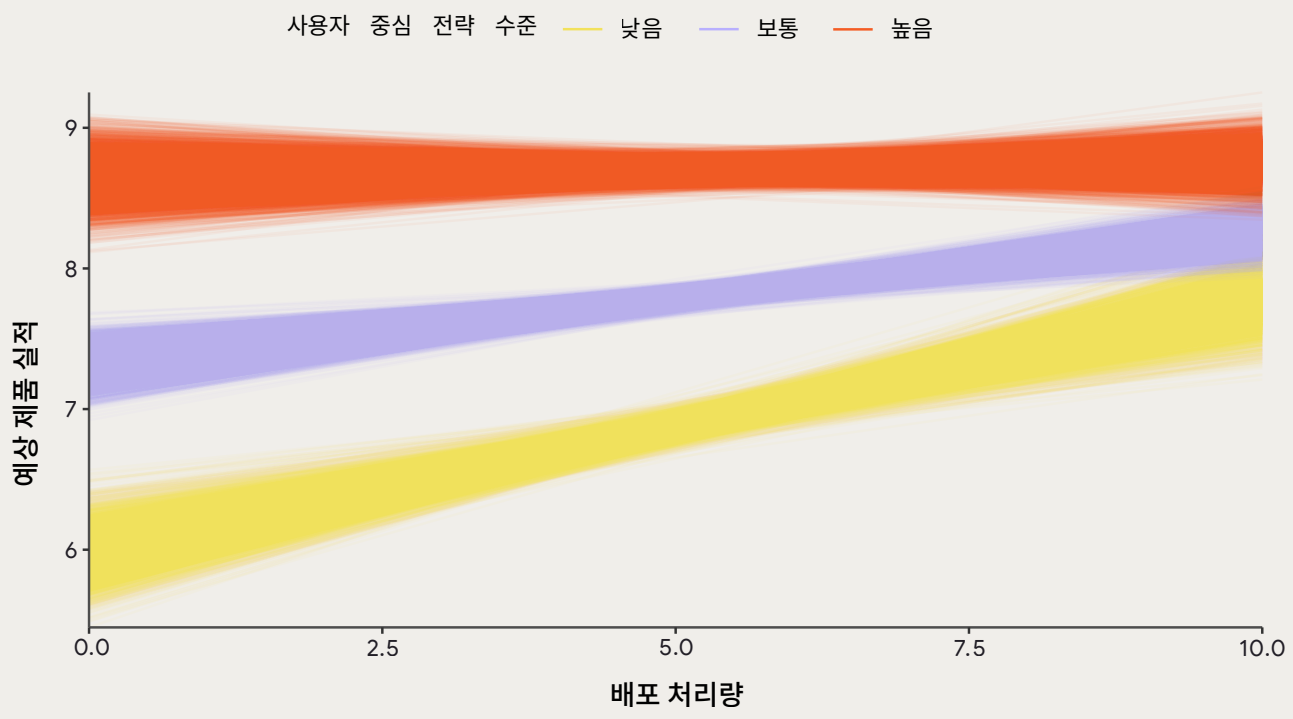


그림 15: 3가지 수준의 사용자 중심 전략에 따른 제품 실적 및 배포 처리량

소프트웨어 개발에 대한 사용자 중심적인 접근 방식이 강력한 철학이자 관행인 이유

학계 연구에 따르면 작업에서 목적의식을 이끌어내는 것이 직원과 조직에 도움이 된다고 합니다.[2,3]

예를 들어 최근 설문조사에 따르면 직장인의 93%가 자신이 하는 일이 의미 있다고 느끼는 직업을 갖는 것이 중요하다고 답했습니다.[4] 비슷한 맥락에서 또 다른 설문조사에 따르면 응답자들은 언제든 의미 있는 직업을 가질 수 있다면 평균적으로 전체 미래 수입의 23%를 포기할 의향이 있는 것으로 나타났습니다.[5]

이것이 직원들이 기꺼이 감수할 수 있는 절충점이라는 점은 놀랍습니다. 이는 사람들에게 동기를 부여하는 요소에 대해 알려주며, 사람들은 의미 있는 일에 시간을 투자하고 싶어 한다는 사실을 보여줍니다.

"모든 사람이 회사 외부의 개인이나 지역사회에 긍정적인 영향을 미치는 회사에서 일할 수 있다면 정말 멋질 것입니다. 언제나 그런 것은 아닙니다. 항상 가능한 것도 아닙니다. 자율 주행의 원대한 비전 중 하나는 고속도로를 달리는 동안에도 잠을 자면서 운전할 수 있게 되는 것입니다. 제가 여기 온 이유는 그 때문이 아닙니다. 저는 운전을 할 수 없는 사람이 원하는 곳 어디든 자유롭게 이동할 수 있도록 돕고 싶습니다."(P2)[6]

명확한 방향성 제공

소프트웨어 개발에 대한 사용자 중심적인 접근 방식은 개발자가 자신의 작업을 바라보는 시각을 근본적으로 바꿀 수 있습니다. 개발자는 임의의 기능을 배포하고 사용자가 사용할지 여부를 추측하는 대신 사용자 피드백을 바탕으로 빌드할 기능의 우선순위를 정할 수 있습니다.

이러한 접근 방식을 통해 개발자는 자신이 작업 중인 기능에 존재 이유가 있다는 확신을 가질 수 있습니다. 사람들이 제품과 서비스를 사용할 때 최상의 경험을 제공한다는 점에서 이들의 업무는 의미가 있습니다. 더 이상 개발된 소프트웨어와 그 소프트웨어가 존재하는 세상 사이에 단절은 존재하지 않습니다.

개발자는 자신이 만든 소프트웨어를 통해 작업의 직접적인 영향을 확인할 수 있습니다.

"회사의 입장으로 서비스를 배포해야 한다는 압박을 받고 있습니다. 따라서 반짝이는 멋진 것들, 개선 방안에 대한 논의 사항 등 모든 것이 최근 조직 구조의 변화로 인해 품질이 아닌 배포에 초점을 맞추고 있는데, 개인적으로 이는 큰 고민거리입니다." (P9)

부서 간 협업 강화

아무리 유능한 개발자라도 혼자서 소프트웨어를 빌드하지는 않습니다. 우수한 품질의 제품을 빌드하려면 서로 다르지만 상호 보완적인 재능이 있는 다수의 사람과 협업해야 합니다.

사용자 중심적인 개발 접근 방식을 통해 개발자는 조직 전체에서 부서 간 협업에 참여할 수 있습니다. 이 과정에서 이들의 책임은 단순히 소프트웨어를 배포하는 것 이상으로 확장됩니다. 이제 개발자는 사용자를 위한 놀라운 경험을 만들기 위해 노력하는 팀의 일원이 됩니다.

소프트웨어 개발에 대한 이러한 접근 방식은 개발자가 고립된 환경에서 벗어나 협업을 추구하고 팀워크를 강화하며 다른 사람들로부터 더 많은 것을 배울 기회를 마련하는 데 도움이 될 수 있습니다. 문제 해결은 다른 형태로 이루어집니다. 기술적인 문제를 해결하는 방법뿐만 아니라 사용자에게 가장 적합한 방식으로 문제를 해결하는 방법도 중요합니다.

이러한 접근 방식은 직원 참여를 향상하고 지적인 자극을 주는 환경을 조성하여 번아웃과 관련된 정체감을 방지하는 데 도움이 될 수 있습니다.

그렇다면 조직은 무엇을 할 수 있을까요?

연구 결과를 바탕으로 조직은 사용자를 파악하는 데 시간과 리소스를 투자할 것이 좋습니다. 빌드 대상과 그들이 겪는 문제를 이해하는 데 집중하세요. 이것이 가치 있는 투자라고 굳게 믿습니다.

사용자에 대한 가정을 하고 싶은 유혹을 뿌리치세요. 사용자의 환경에서 사용자를 관찰하고, 사용자에게 질문하고, 사용자가 말하는 것에 따라 방향을 전환할 수 있을 만큼 겸손해야 합니다. 이렇게 하면 개발자의 생산성이 향상되고 번아웃을 덜 겪으면서도 더 우수한 품질의 제품을 배포할 수 있습니다.

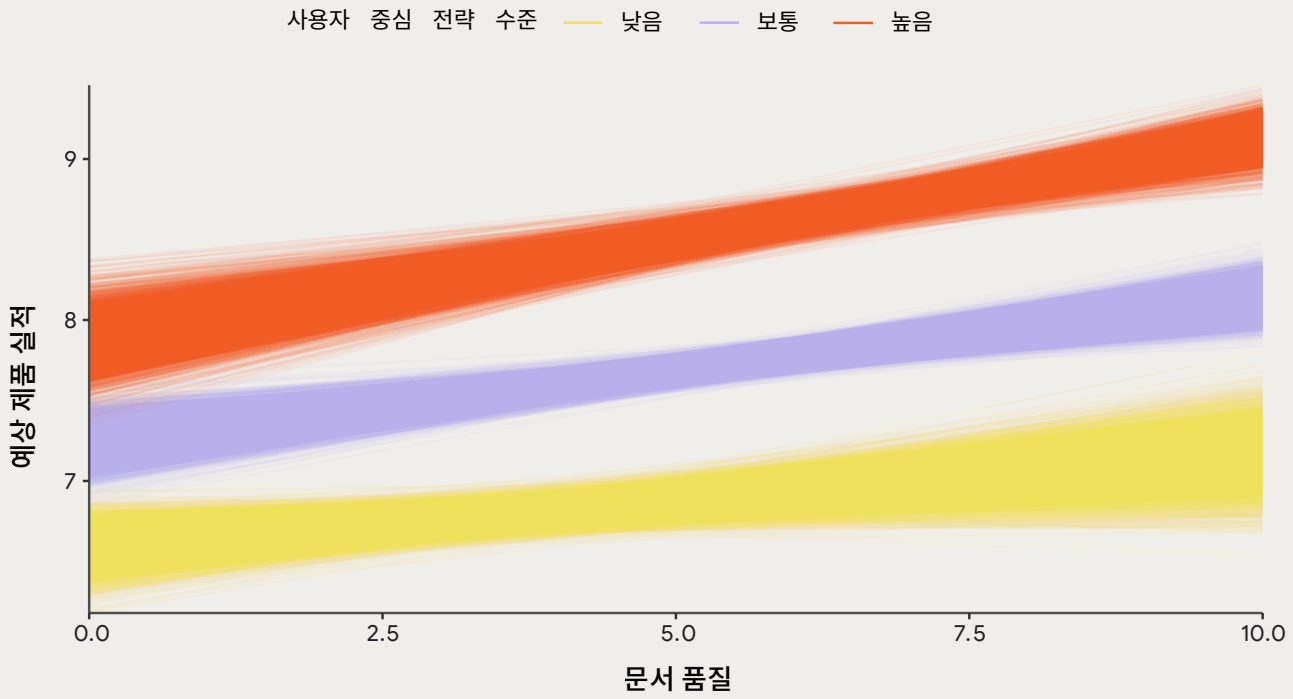
좋은 문서와 사용자 중심적인 소프트웨어 개발 접근 방식은 강력한 조합입니다.

사용자에게 집중하는 팀은 제품 실적이 향상되는 것을 볼 수 있습니다. 이러한 사용자에 대한 집중이 양질의 내부 문서 환경과 결합되면 제품 실적은 더욱 향상됩니다 (그림 16 참조). 이러한 결과는 문서가 기술 역량이 조직 실적에 미치는 영향을 증폭시키는 경우에서 볼 수 있는 행동과 유사합니다.[7]

문서는 사용자 신호와 피드백을 팀 전체와 제품 자체에 전파하는 데 도움이 됩니다.

내부 문서는 사용자 신호가 없으면 예측된 제품 실적에 의미 있는 영향을 미치지 못한다는 것을 알 수 있습니다. 그러나 팀에 양질의 내부 문서가 있다면 그 문서에 포함된 사용자 신호가 제품 실적에 더 큰 영향을 미칩니다.

Google은 2021년부터 문서를 연구하기 시작했으며, 매년 양질의 문서가 미치는 광범위한 영향력을 지속적으로 발견하고 있습니다. 올해의 연구 결과에는 내부 문서가 예측된 제품 실적에 미치는 영향이 목록에 추가되었습니다.



그래프는 가장 개연성 있는 패턴을 추정하기 위해 시뮬레이션에서 얻은 12,000개의 라인을 합성한 것입니다.
그림 16: 3가지 수준의 사용자 중심 전략에 따른 제품 실적 및 문서 품질

문서 문화

애자일 선언은 '종합적인 문서보다 작동하는 소프트웨어'를 지지합니다.[7] 하지만 양질의 문서가 업무용 소프트웨어의 핵심 요소라는 사실이 계속 확인되고 있습니다.

'종합적인 문서'는 문서가 포함된 비정상적 관행을 대신하는 문구일 수 있습니다. 문제가 되는 문서에는 관료적 목적으로만 작성되거나 경영진과 직원 간의 불신을 덮기 위해 작성된 문서가 있습니다. 비정상적인 문서 문화에는 문서를 작성하지만 문서를 관리하거나 통합하지 않는 것도 포함될 수 있습니다.

이러한 경우 문서 품질 측정에서 낮은 점수를 받을 가능성이 높습니다. 이러한 유형의 콘텐츠는 잘못된 대상을 위해 작성된 것이므로 업무 중 사용하려고 할 때 실적이 좋지 않게 됩니다. 문서가 너무 많으면 문서가 부족한 것만큼이나 문제가 될 수 있습니다.

우수한 문서에 대한 기준에는 문서의 검색성 및 신뢰성과 같은 속성이 포함됩니다. 내부 문서의 경우 주요 대상은 동료 또는 특정 작업을 수행하려는 미래의 자신이라는 점을 기억하세요.[8] 건강한 문서 문화를 가진 팀은 이러한 독자를 지원하는 데 중점을 둡니다. 이는 사용자에게 집중하는 또 다른 방법입니다.

다음과 같이 우수한 품질의 문서를 작성하기 위해 확인된 관행을 따르면 팀에서 건강한 문서 작성 문화를 조성할 수 있습니다.

- 중요한 사용 사례 문서화하기
- 기술 문서 작성 교육 수강하기
- 소유권 및 문서 업데이트 프로세스 정의하기
- 팀 내에서 문서 작업 배포하기
- 문서화를 소프트웨어 개발 수명 주기의 일부로 관리하기
- 오래되거나 중복된 문서 삭제하기
- 실적 검토 및 승진에서 문서 작업 인정하기

끊임없이 변화하는 우선순위의 위험성

모두 그 기분을 잘 알고 있습니다. 여러분은 지난 몇 달 동안 새로운 기능을 개발하기 위해 노력해 왔습니다. 사용자를 위해 빌드하는 것이 옳다는 것을 알고 있고, 집중하고 의욕이 넘칩니다. (겉으로 보기에) 갑자기 경영진이 조직의 우선순위를 바꾸기로 합니다. 이제 프로젝트가 일시 중지될지, 폐기될지, 프랑켄슈타인화될지, 돌연변이가 될지 불분명합니다.

이러한 보편적인 경험은 직원과 조직에 큰 영향을 줄 수 있습니다. 이제 조직이 지속적으로 우선순위를 바꿀 때 어떤 일이 발생하는지 살펴보겠습니다.

연구 결과와 그 의미

전반적으로 연구 결과, 조직의 우선순위가 불안정할 경우 생산성이 작지만 의미 있는 수준으로 감소하고 번아웃이 많이 증가하는 것으로 나타났습니다.

데이터에 따르면 이러한 번아웃의 증가를 완화하는 것은 어려운 일입니다. 강력한 리더, 좋은 내부 문서, 사용자 중심적인 소프트웨어 개발 접근 방식이 번아웃에 영향을 미치는 우선순위 변화에 대응하는 데 도움이 되는지 연구했습니다.

정답은 '도움이 되지 않는다'입니다. 조직이 이러한 긍정적인 특성을 모두 가지고 있어도 우선순위가 불안정하면 직원들은 여전히 번아웃을 경험할 위험이 있습니다.

불안정한 조직 우선순위가 직원의 웰빙에 악영향을 미치는 이유

Google은 조직의 우선순위가 불안정하면 기대치가 불분명해지고, 직원의 통제감이 떨어지며, 업무량이 증가하여 직원의 번아웃이 증가한다는 가설을 세웠습니다.

분명한 것은 우선순위 변경 자체에 문제가 있는 것이 아니라는 점입니다. 비즈니스 목표와 제품 방향은 항상 바뀝니다. 조직의 우선순위를 유연하게 조정하는 것은 좋을 수 있습니다.

우선순위가 자주 바뀌는 것이 직원의 웰빙에 부정적인 영향을 미친다고 생각합니다. 불안정한 우선순위에 수반되는 불확실성은 우선순위가 자주 바뀌는 고질적인 문제를 시사합니다.

수십 년에 걸친 학술 연구에 따르면 만성 스트레스가 건강과 웰빙에 미치는 해로운 영향이 밝혀졌습니다.[9] 만성 스트레스에 대한 연구와 Google의 연구 결과 사이에는 유사점이 있습니다. 만성적인 불안정성은 불확실성을 높이고 인지된 통제력을 감소시킵니다. 이 조합은 번아웃에 대한 훌륭한 처방전입니다.

우선순위가 안정화되면 발생하는 일

이번 연구 결과는 다소 의아합니다. 우선순위가 안정화되면 소프트웨어 배포 실적이 감소하는 것으로 나타났습니다. 배포 속도가 느려지고 안정성이 떨어집니다.

이는 안정적인 우선순위를 가진 조직은 일반적으로 양호한 상태의 제품과 서비스를 보유하고 있어 변경사항이 덜 자주 실행되기 때문일 수 있다는 가설을 세우고 있습니다. 또한 우선순위의 안정성으로 인해 권장되는 것보다 적은 양을 더 많이 배포할 수도 있습니다.

하지만 이는 예상치 못한 결과입니다. 조직의 우선순위를 안정화하면 소프트웨어 배포의 속도와 안정성이 떨어지는 이유는 무엇이라고 생각하시나요?

최종 사용자를 위해 AI를 구축하면 우선순위의 안정성은 확보할 수 있지만 배포의 안정성은 확보할 수 없습니다.

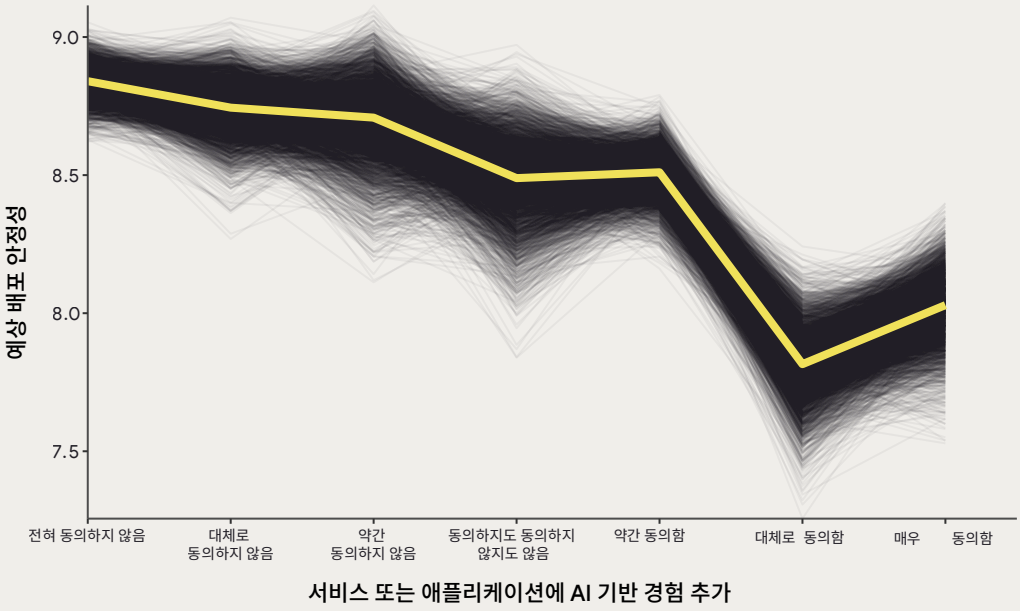
최종 사용자를 위한 AI 기반 경험을 통합하면 조직의 우선순위가 안정화됩니다. 이는 AI에 대한 화려한 찬사처럼 들립니다. 하지만 이 결과가 AI 자체에 대해 의미 있는 내용을 담고 있는 것이라고 해석하지는 않습니다.

대신, AI 빌드를 위한 노력으로 전환하는 것이 조직이 따라야 할 명확성과 방향성을 제공한다고 믿습니다. AI가 아닌 이러한 명확성이 조직의 우선순위를 안정화하는 원동력입니다.

이는 새로운 기술이 등장할 때 조직에 어떤 일이 일어나는지에 대해 알려주기 때문에 강조할 만한 가치가 있습니다. 새로운 기술이 변화를 일으키면 조직은 적응할 시간이 필요합니다. 이 시기에는 리더가 조직을 위한 최선의 방법을 찾으려고 노력하면서 우선순위가 불안정해질 수 있습니다. 혼란이 진정되고 조직이 다음 단계를 명확히 하면 우선순위가 안정화되기 시작합니다.

하지만 우선순위가 안정화된다고 해서 소프트웨어 배포 프로세스가 바로 안정화되는 것은 아닙니다. 분석 결과, 서비스나 애플리케이션에 AI 기반 경험을 추가하는 전환에는 어려움과 성장통이 수반되는 것으로 나타났습니다.

전환한 팀은 그렇지 않은 팀에 비해 소프트웨어 배포 안정성이 10% 정도 현저히 떨어지는 것으로 나타났습니다. 다음은 이 문제를 시각화한 것입니다.



*각 라인은 가장 개연성 있는 패턴을 추정하기 위해 시도한 4,000개의 시뮬레이션 중 하나입니다.

그림 17: 서비스 또는 애플리케이션에 AI 기반 경험을 추가하는 기능으로서의 소프트웨어 배포 안정성

그렇다면 조직은 무엇을 할 수 있을까요?

답은 쉽지만 그렇게 간단하지 않을 수도 있습니다. 연구 결과를 바탕으로 조직은 우선순위를 안정화하는 데 집중하는 것이 좋습니다. 이는 불안정한 우선순위가 직원의 번아웃에 미치는 부정적인 영향에 대응할 수 있는 확실한 방법 중 하나입니다.

연구 결과, 불안정한 우선순위로 인한 부정적인 영향은 훌륭한 리더, 좋은 문서화, 사용자 중심의 소프트웨어 개발 접근 방식을 유지하는 데 방해가 되는 것으로 나타났습니다. 따라서 (1) 우선순위를 안정시키고 (2) 직원이 지속적인 우선순위 변화로 인해 일상에 영향을 받지 않도록 보호하는 방법을 찾는 것 외에는 번아웃을 방지하기 위해 조직이 할 수 있는 일이 많지 않다는 결론에 도달합니다.

1. https://www.nngroup.com/articles/bridging-the-designer-user-gap/
2. https://executiveeducation.wharton.upenn.edu/thought-leadership/wharton-at-work/2024/03/creating-meaning-at-work/
3. https://www.apa.org/pubs/reports/work-in-america/2023-workplace-health-well-being
4. https://bigthink.com/the-present/harvard-business-review-americans-meaningful-work/
5. https://hbr.org/2018/11/9-out-of-10-people-are-willing-to-earn-less-money-to-do-more-meaningful-work
6. (P[N]), 예를 들어 (P1)은 인터뷰 참가자의 가명을 나타냅니다.
7. https://cloud.google.com/blog/products/devops-sre/deep-dive-into-2022-state-of-devops-report-on-documentation 및 2023 Accelerate State of DevOps 보고서 - https://dora.dev/research/2023/dora-report
8. https://agilemanifesto.org/
9. 경영진, 규제 기관 또는 감사자와 같은 다른 대상도 마찬가지임
10. 코헨 S, 자니키-데버트 D, 밀러 GE. Psychological Stress and Disease. JAMA. 2007년;298(14):1685~1687.doi:10.1001/jama.298.14.1685

최고 수준의 혁신

혁신이 효과를 발휘하려면 많은 것을 갖추어야 합니다. 올해 Google은 안정성을 우선시하고, 사용자에게 집중하며, 훌륭한 리더를 보유하고, 양질의 문서를 작성하는 팀이 실적이 우수한 팀이라는 사실을 확인했습니다. Google의 연구는 성공적인 전환을 위한 과정을 계획하는 데 도움이 되는 몇 가지 유용한 경로를 제시합니다.

지속적인 개선이라는 마음가짐으로 혁신에 접근하는 것이 성공의 열쇠라는 것을 알게 되었습니다. 연구 결과, 실적이 우수한 기업은 자신을 방해하는 변수를 파악하고 DORA 측정항목을 기준으로 삼아 체계적이고 지속적으로 개선하고 있습니다. 장기적인 성공을 위해서는 모든 요소에서 우수성을 갖춰야 하지만, 10년간의 DORA 연구는 조직에서 변화를 주도할 수 있는 구체적이고 영향력 있는 4가지 방법을 제시했습니다.

혁신적인 리더십

혁신적인 리더십은 리더가 직원들의 가치와 목적의식에 호소하여 더 높은 실적을 달성하도록 영감을 주고 동기를 부여하여 조직의 광범위한 변화를 촉진하는 모델입니다.

이러한 리더는 다음과 같은 기준을 통해 팀이 공동의 목표를 향해 나아가도록 독려합니다.[1]

비전	팀과 조직이 나아갈 방향에 대한 명확한 비전을 가지고 있습니다.
영감을 주는 커뮤니케이션	팀에 대해 긍정적인 피드백을 제공하고, 직원이 조직의 일원이 된 데에 대한 자부심을 불어넣으며, 변화하는 상황에서 기회를 볼 수 있도록 유도합니다.
지적인 자극	팀원이 오래된 문제를 새로운 방식으로 생각하고 업무에 대한 기본적인 가정을 재고하도록 요구합니다.
지원적 리더십	행동하기 전에 다른 사람의 감정을 고려하고, 다른 사람의 개인적인 요구를 염두에 두고 행동합니다.
개인 인식	팀원이 평균 이상의 성과를 내거나 업무 품질이 향상될 때 이를 인정합니다.

올해 혁신적인 리더십이 직원 생산성 향상으로 이어진다는 사실을 확인했습니다. 혁신적인 리더십을 25% 높이면 직원 생산성이 9% 향상되는 것으로 나타났습니다.

혁신적인 리더십은 생산성 향상 그 이상의 효과를 가져올 수 있습니다. 훌륭한 리더가 있으면 다음과 같은 결과를 얻을 수 있습니다.

- 직원 번아웃 감소
- 업무 만족도 향상
- 팀 실적 향상
- 제품 실적 개선
- 조직 실적 개선

연구 결과, 2017년에는 위의 리더십 자질과 IT 실적 간에 통계적으로 유의미한 관계가 있는 것으로 나타났습니다. 실적이 우수한 팀에는 리더가 5가지 특성 모두에서 높은 점수를 받았고, 실적이 저조한 팀에는 리더가 가장 낮은 점수를 받았습니다. 또한, 회사에서 근무할 것을 추천할 가능성을 나타내는 직원 순 추천 고객 지수(eNPS)와 혁신적인 리더십 사이에는 밀접한 상관관계가 있음을 확인했습니다.

다만, 혁신적인 리더십은 그 자체로 높은 실적을 이끌어내는 것이 아니라 조력자 역할을 하는 것으로 보아야 합니다.

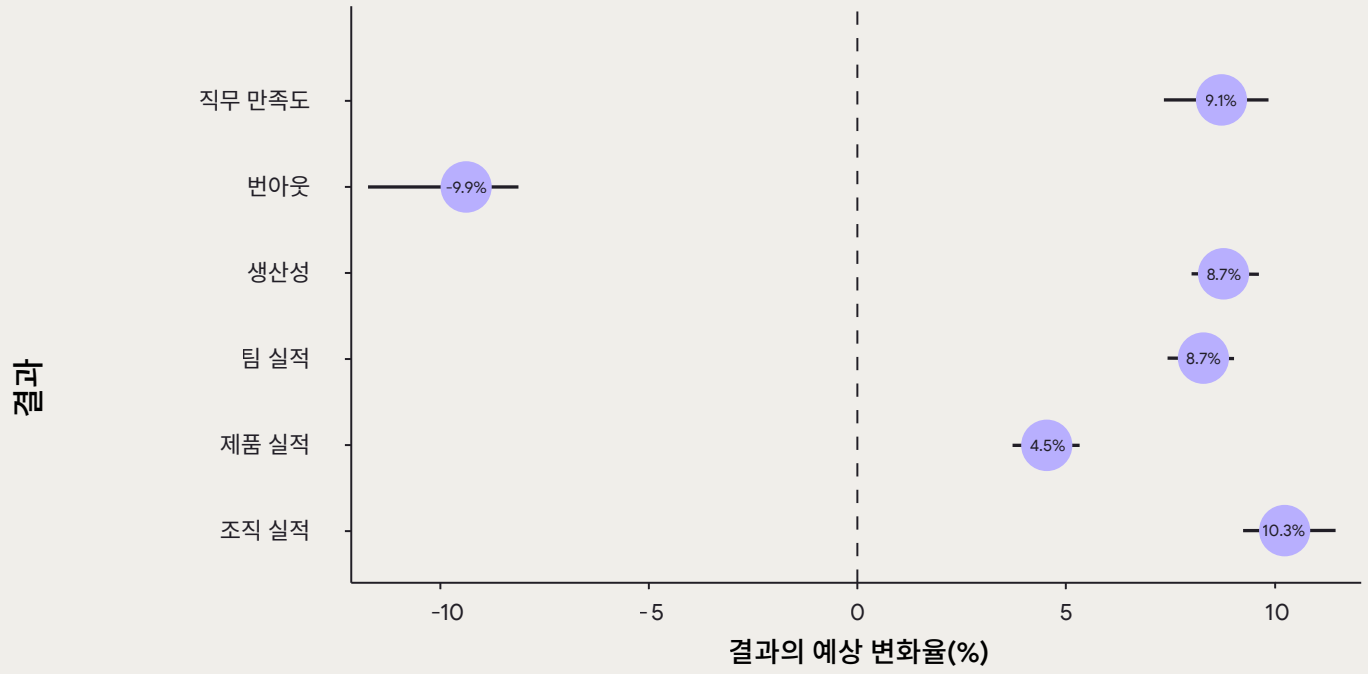
혁신적인 리더십은 기술 및 제품 관리 역량과 관행을 도입하는 데 핵심적인 역할을 합니다. 이는 (1) 팀에 권한과 자율성을 위임하고, (2) 문제 해결에 필요한 측정항목과 비즈니스 인텔리전스를 제공하며, (3) 기능 제공이 아닌 가치 제공을 중심으로 인센티브 구조를 만들면 가능해집니다.

혁신에는 시간이 걸리고 도구가 필요합니다. 리더십은 개선 작업을 위해 리소스를 구체적으로 할당해야 합니다. 훌륭한 리더는 개선에 필요한 시간과 자금을 팀에 제공하는 면에서 핵심적인 역할을 합니다. 엔지니어에게 쉬는 시간에 새로운 것을 배우고 자동화하는 것을 기대해서는 안 되며, 이는 엔지니어의 일정에 포함해야 합니다.

연구 결과는 IT가 비용의 중심이라는 관점을 비즈니스 성공을 이끄는 투자라는 관점으로 전환하는 데 도움이 되었습니다. Google은 2020년에 작성한 DevOps의 ROI 백서[2]에서 IT 개선에 투자함으로써 창출되는 잠재적 가치를 명확히 설명하는 데 사용할 수 있는 계산 방법을 포함했습니다.

금전적 수익은 이 투자에서 기대할 수 있는 수익 중 하나에 불과합니다. 2015년에 실시한 연구에 따르면 "DevOps에 대한 조직의 투자는 조직 문화, 개발, 운영 및 정보보안 팀의 윈윈 성과 달성 능력, 낮은 수준의 번아웃, 보다 효과적인 리더십, 지속적 배포 및 린 관리 관행의 효과적인 구현과 밀접한 상관관계가 있습니다."라고 합니다.[3] 개선을 위해 일정 용량을 특별히 할당하는 것이 좋습니다.

혁신적인 리더십이 25% 증가하면...



포인트 = 예상값
오차 막대 = 89% 불확실성 구간
그림 18: 혁신적인 리더십이 다양한 결과에 미치는 영향

지속적인 사용자 중심 전략

올해의 연구에 따르면 리더가 강력하고 사용자 니즈를 충족하는 소프트웨어 빌드에 집중하는 조직이 더 좋은 제품을 개발하는 것으로 나타났습니다. 이는 강력한 조합입니다. 사용자가 소프트웨어 개발의 중심에 있을 때 리더는 비전을 명확하게 제시할 수 있습니다.

궁극적인 목표는 사용자가 우리가 만든 제품을 좋아하게 만드는 것입니다. 개발자 환경 장에서 설명했듯이 사용자에게 초점을 맞추면 제품 기능의 존재 이유를 알 수 있습니다. 개발자는 사용자 환경을 개선하는 데 도움이 된다는 사실을 인지하고 이러한 기능을 자신 있게 빌드할 수 있습니다.

사용자의 니즈를 깊이 이해하고 이에 부합하고자 하는 열망이 강하며 사용자 피드백을 수집, 추적, 대응하는 메커니즘을 갖춘 팀은 조직 실적도 가장 높다는 것을 알 수 있습니다. 사실, 사용자 중심적인 조직이라면 높은 수준의 소프트웨어 속도와 안정성이 없어도 성공할 수 있습니다. 2023년에는 사용자 중심적인 팀이 그렇지 않은 팀에 비해 40% 더 높은 수준의 조직 실적을 보였으며[4], 2016년에도 사용자 중심 팀의 조직 실적이 더 우수하다는 것을 확인했습니다.

올해의 연구는 이전 연구 결과를 반영합니다. 사용자에게 집중하는 팀이 더 좋은 제품을 만듭니다.

제품이 개선될 뿐만 아니라 직원의 업무 만족도가 높아지고 번아웃을 경험할 가능성도 줄어듭니다.

빠르고 안정적인 소프트웨어 배포를 통해 조직은 실험하고 배울 기회가 더 많아집니다. 이러한 실험과 반복은 사용자 피드백을 기반으로 하는 것이 가장 이상적입니다. 소프트웨어를 빠르고 안정적으로 배포하면 실험하고, 사용자 니즈를 더 잘 이해하고, 이러한 니즈가 충족되지 않을 경우 신속하게 대응할 수 있습니다.

또한, 빠르고 안정적으로 배포하면 시장 변화나 경쟁에 더 쉽게 적응할 수 있습니다.

내부 개발자도 사용자라는 사실을 기억해야 합니다. 내부 개발자 플랫폼(IDP)을 통해 조직이 개발자에게 가치를 제공하고, 개발자는 다시 외부 사용자나 다른 내부 사용자에게 가치를 제공할 수 있습니다.

연구에 따르면 성공적인 IDP는 개발자가 독립적으로 작업할 수 있는 환경을 제공하기 위해 사용자 중심 전략에 중점을 두고 제품 형태로 개발됩니다. 이러한 방식으로 배포된 IDP는 개인 생산성, 팀 생산성, 조직 실적 향상으로 이어집니다.

데이터에 기반한 조직으로 거듭나기

성공을 향한 진행 상황을 시각화하는 능력은 매우 중요합니다. 지난 10년 동안 데이터에 기반한 조직이 되기 위해 최선을 다해 왔습니다. DORA의 4가지 핵심 측정항목[5]은 소프트웨어 배포 실적을 측정하는 글로벌 표준이 되었지만 이는 이야기의 일부에 불과합니다. 조직 개선을 추진하는 데 사용할 수 있는 30가지가 넘는 기능과 프로세스[6]를 확인했습니다.

측정항목의 가치는 개선되고 있는지 알려주는 측정항목의 기능에 달려 있습니다. 4가지 핵심 측정항목은 조직이나 비즈니스 라인 수준이 아닌 애플리케이션 및 서비스 수준에서 사용해야 합니다. 측정항목은 지속적인 개선 노력을 시각화하는 데 사용해야 하며, 팀을 비교하거나 개인을 비교하는 데 사용해서는 안 됩니다.

또한 이 측정항목을 애플리케이션 또는 서비스 팀의 성숙도 모델로 사용해서는 안 됩니다. 저조한 성과자, 중간 성과자, 우수한 성과자 또는 최우수 성과자라는 말은 흥미롭지만 이러한 명칭은 혁신 과정의 맥락에서 별다른 가치가 없으므로 주의가 필요합니다.

Google의 연구가 진행되고 발전하는 지금, 4가지 핵심 측정항목 그 이상을 생각하세요. 사용자 피드백 측정항목이 4가지 핵심 측정항목만큼이나 중요하다는 것이 분명해졌습니다. 이는 대부분의 팀이 속도와 안정성을 개선하기 위해 실행 가능한 솔루션을 마련했기 때문이라고 생각합니다. 그 결과, 높은 실적이 보편화되면서 속도와 안정성으로 얻게 되는 이점이 줄어들게 됩니다.

혁신을 전체적으로 고려할 때는 기술 측정항목(예: 4가지 핵심 및 신뢰성 측정항목)과 비즈니스 측정항목을 모두 결합한 대시보드와 시각화를 만드는 것이 좋습니다. 이는 하향식 혁신 노력과 상향식 혁신 노력 간의 격차를 해소하는 데 도움이 됩니다. 이는 또한 지향점, OKR, 직원 목표를 IT에 대한 투자와 연결하는 데 도움이 됩니다. 이들은 ROI를 정량화할 수 있습니다.

측정항목은 탁월한 성과를 위한 필수 요건이라고 생각합니다. 측정항목은 의사 결정을 용이하게 합니다. 정량적, 정성적 측정항목을 많이 수집할수록 더 많은 정보를 바탕으로 더 나은 의사 결정을 내릴 수 있습니다. 데이터의 가치나 의미에 대해서는 저마다의 의견이 있기 마련이지만, 의사 결정의 근거로 데이터를 사용하는 것이 의견이나 직관에 의존하는 것보다 더 나은 경우가 많습니다.

클라우드에 올인하거나 데이터 센터에 머물기

2018년부터 NIST에서 정의한 클라우드 컴퓨팅의 5가지 특성[7](주문형 셀프 서비스, 광범위한 네트워크 액세스, 리소스 풀링, 빠른 탄력성, 유연한 인프라라고도 하는 측정 서비스)과 조직의 실적 간 관계를 연구해 왔습니다. 성공적인 팀은 덜 성공적인 팀보다 유연한 인프라를 활용할 가능성이 높다는 것을 알 수 있습니다.

작년에 연구를 시작한 이후 지금까지 이 주제에 관한 가장 놀라운 정보를 얻었습니다. 5가지 특성을 활용하지 않고 클라우드를 사용하면 조직의 실적이 저하될 수 있습니다. 애플리케이션이나 서비스를 근본적으로 혁신할 의향이 없는 조직은 데이터 센터에 머무는 것이 더 나을 수 있습니다. 물론 이를 위해서는 단순히 도구나 기술을 도입하는 데 그치지 않고 애플리케이션을 설계, 구축, 배포, 실행하는 데 있어 완전히 새로운 패러다임이 필요한 경우가 많습니다. 대규모 변경사항은 소수의 서비스로 시작할 때 더 쉽게 적용할 수 있으며, 팀과 조직이 앞으로 나아가면서 학습하고 개선하는 데 도움이 되는 반복적인 접근 방식을 취하는 것이 좋습니다.

요약

지난 10년 동안 일관되게 확인된 것은 혁신이 성공의 필수 요건이라는 사실입니다. 많은 조직이 오해하는 것은 혁신은 목적지가 아니라 지속적인 개선의 여정이라는 점입니다.[8] 연구 결과는 분명합니다. 지속적으로 개선하지 않는 기업은 결국 뒤처지게 됩니다. 반대로 지속적인 개선이라는 사고방식을 도입한 기업은 가장 높은 수준의 성공을 거둡니다.

이 여정에서 약간의 고통과 불편함을 겪을 수 있다는 점에 유의하세요. 연구에 따르면 DevOps[9], SRE[10], 올해는 플랫폼 엔지니어링을 통해 초기 실적 하락에 이어 큰 폭의 실적 상승('J 곡선'이라고도 함)이 나타났습니다. 이는 정상적인 현상이며, 지속적으로 개선해 나간다면 상황이 점점 나아지고 시작했을 때보다 훨씬 더 나은 상태로 끝날 수 있습니다.

끝이 없는 여정이라고 생각하면 부담스러울 수 있습니다. 완벽한 혁신을 계획하거나 디자인할 때 막히기 쉽습니다. 성공의 열쇠는 팔을 걷어붙이고 일을 시작하는 것입니다. 조직과 팀의 목표는 단순히 어제보다 조금 더 나아지는 것이어야 합니다. 지난 10년간의 연구와 향후 10년에 대한 목표는 여러분이 더욱 발전할 수 있도록 돕는 것입니다.

1. Dimensions of transformational leadership: Conceptual and empirical extensions - A. E. 라퍼티 및 M. 그리핀 A.
2. The ROI of DevOps Transformation - https://dora.dev/research/2020/
3. 2015 State of DevOps 보고서 https://dora.dev/research/2015/2015-state-of-devops-report.pdf#page=25
4. 2023 Accelerate State of DevOps 보고서 - https://dora.dev/research/2023/dora-report/2023-dora-accelerate-state-of-devops-report.pdf#page=17
5. DORA의 4가지 핵심 측정항목 https://dora.dev/guides/dora-metrics-four-keys/
6. DORA의 기능 및 프로세스 https://dora.dev/capabilities/
7. NIST에서 정의한 클라우드 컴퓨팅의 5가지 특성 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf
8. 지속적 개선의 여정 https://cloud.google.com/transform/moving-shields-into-position-organizing-security-for-digital-transformation
9. 2018 Accelerate State of DevOps 보고서 https://dora.dev/research/2018/dora-report/
10. 2022 State of DevOps 보고서 https://dora.dev/research/2022/dora-report/

DORA와 함께한 10년

연혁

DevOps 방식은 2009년에 주제는 관련이 있지만 조율은 없었던 2가지 이벤트에서 탄생했습니다. 존 올스포와 폴 해먼드는 그해 6월 벨로시티 콘퍼런스에서 '하루에 10번 배포: Flickr에서 Dev와 Ops의 협력' 이라는 주제로 강연을 진행했습니다.[1] 팩트릭 드보아는 몇 달 후 자발적인 조직자 팀을 이끌고 벨기에 겐트에서 첫 번째 DevOpsDays 이벤트를 주최했습니다.[2]

오래지 않아 DevOps 커뮤니티는 DevOps 가 어떻게 발전하고 있는지 자세히 알아보고 싶어졌습니다. Puppet Labs에서 일하던 앨레나 브라운은 2011년에 DevOps에 대해 자세히 알아보기 위해 설문조사를 실시했습니다. 이번 설문조사를 통해 'DevOps' 방식으로 작업하는 것이 IT의 새로운 비즈니스 방식으로 부상하고 있다는 사실을 확인할 수 있었습니다.

이러한 움직임이 새로운 산업과 조직으로 계속 확대됨에 따라 앨레나는 이러한 성공을 바탕으로 2012년 IT Revolution Press와 협력하여 또 다른 설문조사를 실시했고, 그 결과를 2013 State of DevOps 보고서에 발표했습니다.[3]

이듬해에는 니콜 포스그렌 박사가 연구팀에 합류하여 프로그램에 과학적 정확성을 강화했습니다. 2014 State of DevOps 보고서[4]에 따르면 소프트웨어 배포 실적과 조직 실적 간의 연관성을 연구한 결과, "IT 팀의 실적이 우수한 상장 기업은 IT 조직의 실적이 저조한 기업보다 3년 동안 시가총액 성장률이 50% 더 높았다"고 합니다.

2016년에는 연례 보고서의 추세가 잘 정립되었고, 포스그렌, 제즈 험블, 진 킴이 DevOps Research and Assessment(DORA)를 설립했습니다. 그해 State of DevOps 보고서에는 DevOps 관행을 도입한 팀이 투자한 금액을 측정하는 데 도움이 되는 계산법이 포함되어 있습니다. 이 작업은 2020년에 발간된 DevOps 혁신의 ROI[5] 백서에서 확장되었습니다.

Accelerate: The Science Behind DevOps: Building and scaling high performing technology organizations[6]는 포스그렌, 험블, 킴이 작성했고 2017년 IT Revolution Press에 발표되었습니다. 이 책은 연구 프로그램의 초창기를 요약하고 개선을 유도하는 기능들에 초점을 맞췄습니다.

DORA는 2018년에 독립 보고서인 Accelerate State of DevOps 보고서: 새로운 경제 전략을 발표했습니다.[7] Puppet 팀은 같은 해부터 DORA와는 별도로 자체 보고서 시리즈[8]를 이어 나갔습니다.

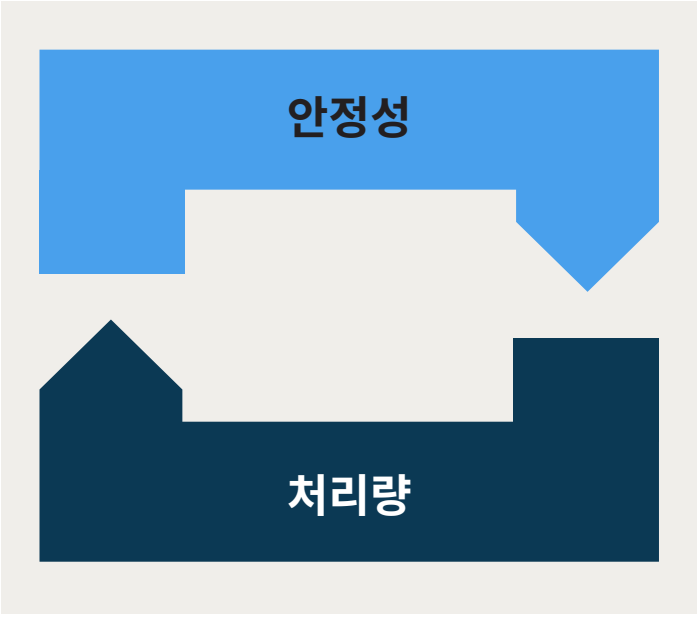
2018년 말, DORA는 플랫폼에 구애받지 않는 과학적 연구를 이어가고 있는 Google Cloud[9]에 인수되었습니다. DORA 보고서가 올해로 열 번째를 맞이했고[10] 그 결과를 여러분과 공유하게 되어 기쁘게 생각합니다. 읽어주셔서 감사합니다.

DORA의 핵심 인사이트

팀은 안정성을 위해 속도를 희생할 필요가 없습니다

기술 기반 팀은 현재 실적을 평가하고, 개선의 우선순위를 정하고, 진행 상황을 검증할 수 있도록 실적을 측정할 방법이 필요합니다. DORA는 소프트웨어 배포 프로세스의 결과를 효과적으로 측정할 수 있는 4가지 소프트웨어 배포 측정항목(4가지 핵심 측정항목)을 확인하고 검증했습니다. 이러한 소프트웨어 배포 실적의 측정은 업계 표준이 되었습니다.

연구 결과, 변경의 처리량과 안정성은 함께 움직이는 경향이 있으며, 모든 업종 카테고리에서 팀이 2가지 모두를 높은 수준으로 달성하는 것을 볼 수 있었습니다.



팀에서 4가지 핵심 측정항목을 측정하는 방법에는 다음과 같은 다양한 방법이 있습니다.

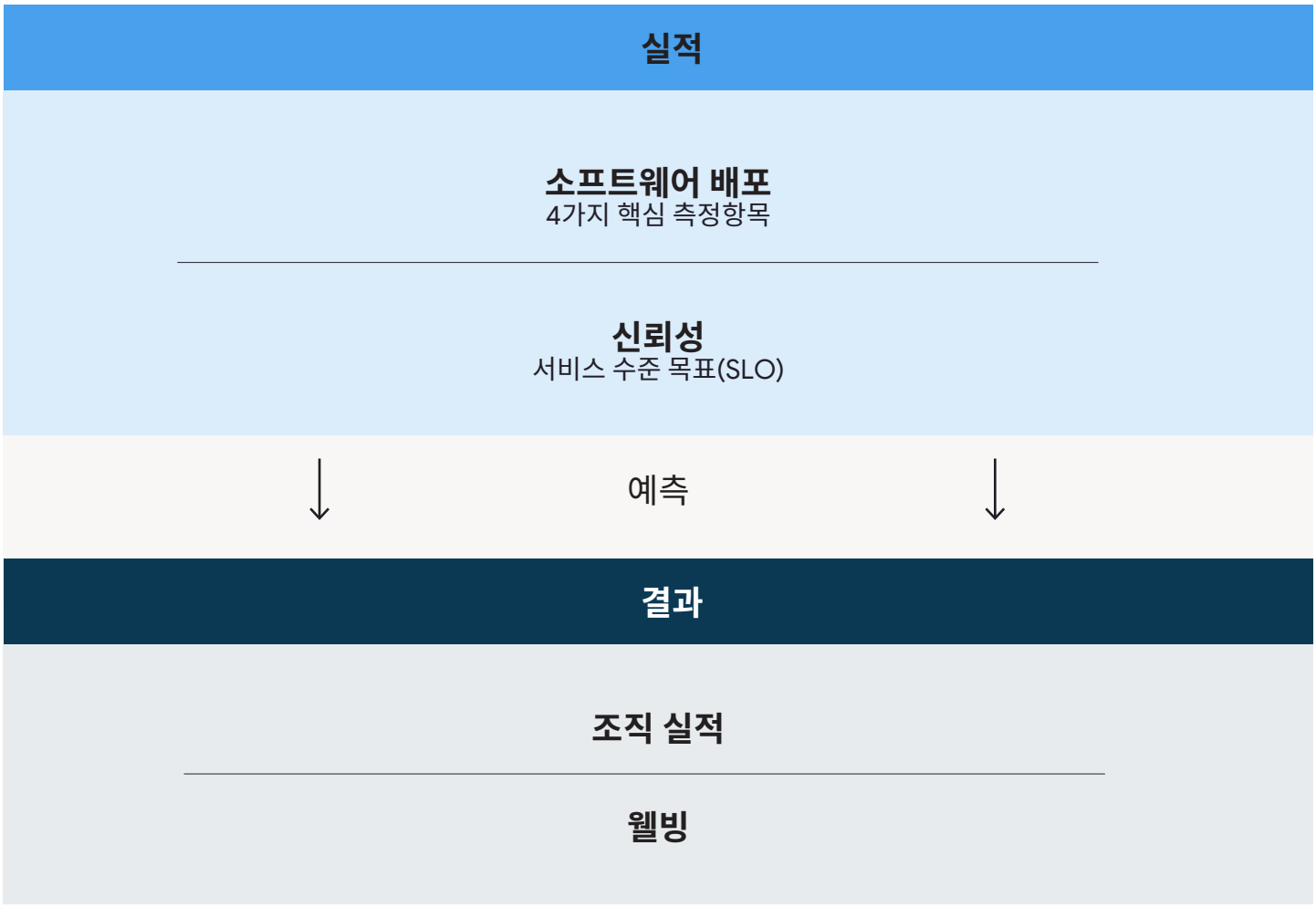
- 팀 회의 중 대화와 성찰을 통해
- DORA 빠른 점검(https://dora.dev/quickcheck)
- Software Engineering Intelligence(SEI) 카테고리의 상용 및 소스 사용 가능[11] 도구
- 팀에서 사용 중인 특정 도구에 맞게 빌드된 맞춤형 통합 기능

소프트웨어 배포 및 운영 실적이 조직의 실적을 견인

DORA는 4가지 핵심 측정항목을 사용하여 소프트웨어 배포 실적을 측정합니다. 운영 실적은 2018년에 DORA에서 처음 연구되었습니다. 소프트웨어 제품 또는 서비스에 대한 약속과 주장을 지키는 능력을 측정합니다.

소프트웨어 배포와 운영 실적이 함께 조직의 실적과 직원 웰빙을 향상할 때 최상의 결과를 얻을 수 있습니다.

기술 기반 팀에서 일하는 실무자는 애플리케이션 사용자의 신뢰성 기대치를 충족하면서 배포 프로세스에서 마찰을 줄이는 것이 중요하다는 것을 잘 알고 있습니다.



성공의 가장 중요한 요소인 문화

실적에 대한 가장 명확한 예측 변수 중 하나는 조직 문화입니다. 학습과 협업 분위기를 장려하는 높은 신뢰 문화의 힘을 지속적으로 확인했습니다. 예를 들어 2022년 연구에서 조직의 애플리케이션 개발 보안 관행의 가장 큰 예측 요인은 문화인 것으로 나타났습니다.[12]

문화는 연구의 모든 측면에 영향을 미치며, 다면적이고 항상 유동적입니다. Westrum의 조직 문화 유형론과 같은 연구에서 아이디어를 얻어 수년 동안 다양한 측정 기준을 사용해 왔습니다.[13] 웰빙 측정 기준에는 번아웃, 생산성, 직무 만족도가 포함됩니다.

계속 나아지는 성과

팀별로 더 나은 성과를 내기 위한 목표를 설정하도록 권장합니다. 개선을 추진하려면 지속적인 개선에 대한 마음가짐과 실천이 필요합니다. 이를 위해서는 현재 상황을 평가하고, 개선 작업의 우선순위를 정하고, 진행 상황을 측정하는 데 도움이 되는 피드백 메커니즘이 필요합니다.

개선을 위한 실험적 접근 방식에는 승리와 패배가 혼재하지만 팀에서는 두 시나리오 모두에서 얻은 교훈을 바탕으로 의미 있는 조치를 취할 수 있습니다.

앞으로의 10년

지난 10년 동안 서로에게서 많은 것을 배웠습니다. 연례 설문조사와 DORA Community of Practice[14]에 참여하고, 각자의 상황에 맞게 DORA를 활용해 주셔서 감사합니다.

기술 환경이 계속 진화함에 따라 DORA는 기술 중심의 팀과 조직이 성공하는 데 도움이 되는 역량과 관행을 지속적으로 연구할 것입니다. 앞으로도 기술의 인간적인 측면에 우선순위를 두고 플랫폼에 제약을 받지 않는 연구 결과를 발표하여 여러분의 여정을 안내할 수 있도록 최선을 다할 것입니다.

과거의 많은 인사이트는 새로운 기술과 관행에 대한 접근 방식을 알려주기에 충분하며, 여러분과 함께 새로운 인사이트를 찾을 수 있기를 기대합니다.

문화, 협업, 자동화, 학습, 비즈니스 목표 달성을 위한 기술 사용 등 언제나 DevOps 방식의 일부였던 기본 원칙에 전념하고 있습니다. 커뮤니티와 연구는 'DevOps'라는 명칭과 관련이 없는 사람들을 포함하여 역할이 다양한 여러 사람들의 관점을 통해 유익을 얻습니다. 'DevOps'라는 용어가 주목받지 못하는 경우도 예상해야 합니다.

올해 보고서에서는 인공지능(AI)의 사용과 영향에 대해 집중적으로 다루고 있습니다. 아시다시피, 도입이 증가하고 있으며 이 분야에서 실험할 기회가 많습니다. 앞으로도 이러한 새로운 기술 및 관행에 대해 지속적으로 연구할 예정입니다. 새로운 연구와 함께 과거 연구 결과를 활용하여 도입을 촉진하고 팀원 모두의 경험을 개선할 수 있습니다.

1. 슬라이드 - https://www.slideshare.net/jallspaw/10-deploys-per-day-dev-and-ops-cooperation-at-flickr, recording - https://www.youtube.com/watch?v=LdOe18KhtT4
2. https://legacy.devopsdays.org/events/2009-ghent/
3. https://www.puppet.com/resources/history-of-devops-reports#2013
4. 2014 State of DevOps 보고서 - https://dora.dev/research/2014/
5. The ROI of DevOps Transformation - https://dora.dev/research/2020/
6. 니콜 포스그렌, 제즈 험블, 진 킴. 2018년. Accelerate: The Science Behind DevOps: Building and Scaling High Performing Technology Organizations. IT Revolution Press
7. Accelerate State of DevOps: Strategies for a New Economy - https://dora.dev/research/2018/dora-report/
8. https://www.puppet.com/resources/history-of-devops-reports#2018
9. https://dora.dev/news/dora-joins-google-cloud
10. DORA는 몇 년 후에 설립되었지만, 포스그렌 박사가 프로그램에 합류한 해인 2014년을 첫 번째 DORA 보고서로 간주합니다. 2020년에는 보고서가 없었으므로 2024년 보고서가 열 번째 보고서입니다.
11. https://dora.dev/resources/#source-available-tools
12. 2022 Accelerate State of DevOps 보고서 - https://dora.dev/research/2022/dora-report/
13. 론 웨스트럼, 'A typology of organisation culture', BMJ Quality & Safety 13, 2호(2004년), doi:10.1136/qshc.2003.009522
14. https://dora.community

최종 의견

DORA는 지난 10년 동안 신뢰할 수 있는 연구, 인사이트, 정보 출처로 자리매김했습니다. 업계가 플랫폼 엔지니어링 및 인공지능과 같은 새로운 관행과 기술을 지속적으로 도입함에 따라, DORA는 팀 개선에 도움이 되는 업무 방식을 모색하면서 여러분과 함께할 것입니다. DORA와 여정을 함께 해주셔서 감사합니다.

연구 결과 재현하기

올해 보고서의 연구 분야와 결과는 복잡하고 때로는 불분명하거나 모순되기도 합니다. Google의 연구 결과를 재현해 보시기 바랍니다. 한 팀이나 조직에 집중하면 더 깊이 이해할 기회가 많이 생깁니다.

조직 내에서 실험 진행

DORA의 연구 결과는 다음 실험을 위한 가설로 활용할 수 있습니다. 팀 운영 방식에 대해 자세히 알아보고 DORA 연구 프로그램의 결과에서 아이디어를 얻어 개선할 수 있는 영역을 파악하세요.

조직 내에서 설문조사 실시

이 보고서와 올해 설문조사[1]에 사용된 질문을 참고하여 자체 내부 설문조사를 설계하세요. 설문조사에 대상과 관련된 보다 세분화된 질문을 포함할 수 있습니다.[2] 연구 수행 방법에 대한 자세한 내용은 방법론 장을 참조하세요. 조사 결과를 실천에 옮기는 데 집중하시기 바랍니다.

배운 내용 공유하기

실험을 통해 배운 지식을 전사적으로 공유하세요. 공유 방법은 많은 사람들을 대상으로 하는 공식적인 보고서부터 비공식적인 실무 커뮤니티, 동료들 간의 비격식적인 채팅까지 다양합니다. 다양한 접근 방식을 시도해 보고 각자의 상황과 문화에 가장 적합한 방법이 무엇인지 알아보세요. 이 역시 실험 과정입니다.

이 연구를 어떻게 활용하고 계시나요?

DORA Community(https://dora.community)에 참여하여 지속적인 개선의 여정에서 다른 여행자들과 지식과 경험, 아이디어를 공유해 보세요.

DORA COMMUNITY

1. 2024년 설문조사 https://dora.dev/research/2024/questions/
2. 소프트웨어 회사에서 내부적으로 DORA 설문조사를 수행한 경험담 - https://www.infoq.com/news/2024/08/dora-surveys-software-company/

감사의 말씀

올해는 10번째 DORA 보고서를 배포하는 기념비적인 해입니다. 이 작업을 구체화하는 데 동참하고 함께 발전해 온 연구자, 전문가, 실무자, 리더, 혁신 주체들의 헌신적인 노력에 감사드립니다.

Puppet Labs와 IT Revolution Press에서 처음 발간한 State of DevOps 보고서 이후 많은 발전이 있었습니다. 초석을 닦아주신 DORA 창립자 여러분께 진심으로 감사드립니다. 그 이후로 얼마나 많은 변화가 있었는지, 수년 동안 얼마나 많은 것을 배웠는지 되돌아보면 놀라울 따름입니다.

올해 보고서에 참여해 주신 모든 분께 깊이 감사드립니다. 업계 관행을 선도하고 영향을 미치는 것은 막중한 책임이며, 여러분의 기여는 대단히 중요합니다.

초창기부터 지금의 흥미로운 AI 시대에 이르기까지 이 여정에 함께해주신 모든 분께 감사드립니다. 여러분의 지원과 인사이트는 큰 힘이 되었습니다. 앞으로의 10년도 발견과 협업을 위해 나아가겠습니다.

DORA 보고서팀

제임스 브룩뱅크

킴 카스티요

데렉 드벨리스

벤자민 굿

네이슨 하비

미셸 어바인

아만다 루이스

에릭 맥스웰

스티브 맥기

앨리슨 박

데이브 스탠크

케빈 스토어러

다니엘라 비얄바

편집자

세스 로젠블랫

현지화 자원봉사자

앤드류 아놀라스코

마우리시오 멜렌데즈

마리-블랑슈 판투

미구엘 레예스

요시 야마구치

유진홍

DORA 가이드

리사 크리스핀

스티브 펜턴

데날리 룸마

베차엘레 (사울) 윌리엄슨

해당 분야 전문가/자문위원

존 올스포

비르기타 뵈켈러

샌더 보그단

미셸 추비르카

토마스 드 메오

제시카 드비타

롭 에드워즈

니콜 포스그렌 박사

진 킴 및 IT Revolution

로라 맥과이어 박사

제임스 파슈틴스키

라이언 J. 살바

마제드 사마드

하리니 샘패스

로빈 사비나르

션 세드록

더스틴 스미스

핀 토너

골드 스폰서

catchpoint chronosphere DATADOG

Deloitte. Excella Gearset

liatrio Middleware OPSERA

실버 스폰서

autorabit COPIA

XOps apps | by gathr

honeycomb.io mezmo

Octopus Deploy SLEUTH

저자

데렉 드벨리스

데렉은 Google의 정량적 사용자 경험 연구원이자 DORA의 리드 조사 담당자입니다. 데렉은 설문조사 연구, 로그 분석 업무에 주력하고 있으며, 제품 또는 기능을 보여주는 개념의 측정 방법을 파악하여 가변 자본 가치를 제공하는 일을 주로 합니다. 인간 대 AI 상호작용, 코로나19 발병이 금연에 미치는 영향, NLP 오류에 대한 설계, 개인 정보 보호 논의에서 UX의 역할, 팀 문화, AI와 직원 웰빙 및 생산성의 관계를 발표했습니다. 현재 과외 연구를 통해 신념과 권한의 전파를 시뮬레이션하기 위한 방법을 탐구하고 있습니다.

케빈 M. 스토어러

케빈 M. 스토어러 박사는 Google의 개발자 경험 연구원으로, DORA 팀의 질적 조사 책임자로 일하고 있습니다. 케빈은 소프트웨어 엔지니어링 분야의 전문적인 경험과 사회과학 및 인문학 분야의 대학원 학제 간 교육을 활용하여 2015년부터 다양한 문제 상황, 참여자 프로필, 연구 방법을 아우르는 인간 중심의 소프트웨어 개발자 연구를 이끌고 있습니다. 케빈의 연구는 인공지능, 정보 검색, 임베디드 시스템, 프로그래밍 언어, 유비쿼터스 컴퓨팅, 인터랙션 디자인 등의 주제로 저명한 과학 학술지에 게재되었습니다.

아만다 루이스

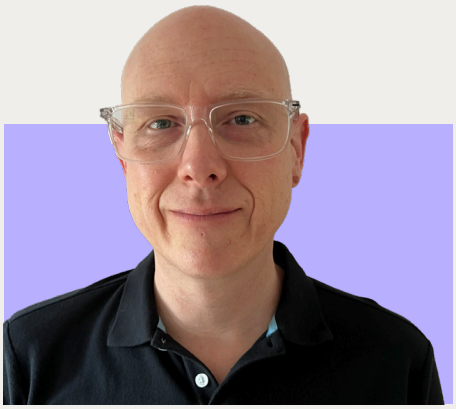
아만다 루이스는 DORA.community의 개발 리드이며 Google Cloud에서는 개발자 관계 엔지니어를 맡고 있습니다. 아만다는 개발자, 운영자, 제품 관리자, 프로젝트 매니저, 리더십 전반에서 소통하는 역할을 하며 경력을 쌓았습니다. 또한 전자상거래 플랫폼, 콘텐츠 관리 시스템, 관측 가능성 도구를 개발하고 개발자를 지원하는 팀에서 일한 경험이 있습니다. 이러한 소통 및 대화 능력은 고객 만족과 더 나은 비즈니스 성과로 이어졌습니다. 아만다는 자신의 경험과 공감 능력을 업무에 활용해 팀이 소프트웨어 배포 및 인공지능 관행을 이해하고 구현할 수 있도록 돕고 있습니다.

벤자민 굿

벤자민 굿은 Google의 클라우드 솔루션 설계자입니다. 클라우드 기술과 자동화를 통해 소프트웨어 배포 관행을 개선하는 데 열정적입니다. 또한 솔루션 설계자로서 아키텍처 안내, 기술 가이드 게시, 오픈소스 참여를 통해 Google Cloud 고객이 문제를 해결할 수 있도록 돕고 있습니다. Google에 입사하기 전에는 덴버/볼더 지역의 여러 회사에서 클라우드 운영을 담당했으며, 그 과정에서 DevOps 관행을 구현했습니다.

다니엘라 비얄바

다니엘라 비얄바는 Google의 사용자 환경 연구원입니다. 설문조사 연구를 통해 개발자의 만족도와 생산성을 높일 수 있는 요인을 파악합니다. Google에 합류하기 전에는 명상 훈련의 이점, 대학생의 경험에 영향을 미치는 심리 사회적 요인을 연구했습니다. 다니엘라는 플로리다 인터내셔널 대학교에서 실험 심리학 박사 학위를 받았습니다.



에릭 맥스웰

에릭 맥스웰은 Google에서 DevOps 혁신 관행을 이끌고 있으며 세계 최고 기업을 대상으로 가치를 더 빠르게 제공하여 개선하는 방법에 대해 조언합니다. 에릭은 엔지니어로서 최전선에서 모든 일을 자동화하고 다른 실무자들에 대한 공감대를 형성하는 데 경력의 전반부를 보냈습니다. 또한 Google 클라우드 애플리케이션 현대화 프로그램(CAMP)의 공동 작성자이며 DORA팀의 멤버입니다. Google에서 근무하기 전에는 Chef Software에서 즐거운 동료들과 멋진 소프트웨어를 개발했습니다.

킴 카스티요

킴 카스티요는 Google의 사용자 환경 프로그램 관리자입니다. 킴은 2022년부터 DORA의 연구 운영을 감독 및 보고서의 출판까지 DORA의 다양한 활동을 주도하고 있습니다. 또한 Google Cloud에서 Gemini용 UX 연구도 진행하고 있습니다. Google에 합류하기 전에는 테크 부문에서 경력을 쌓고, 기술 프로그램 관리 및 애자일 코칭 부문에서 업무를 수행했습니다. 킴은 출신 국가인 필리핀에서 법적 절차 없는 살인, 도시 빈곤 지역 개발, 지역사회 회복력 등의 주제에 초점을 맞춘 심리사회학적 연구를 바탕으로 활동하고 있습니다.

미셸 어바인은 Google의 테크니컬 라이터로, 문서와 기타 기술 커뮤니케이션에 관한 연구를 주도하고 있습니다. Google에 입사하기 전에는 교육 출판 분야에서 일했으며 물리 시뮬레이션 소프트웨어의 테크니컬 라이터로도 활동했습니다. 미셸은 워털루 대학교에서 물리학 학사 학위와 수사학 및 커뮤니케이션 디자인 석사 학위를 받았습니다.

미셸 어바인

네이슨 하비는 Google Cloud에서 DORA 팀을 이끌고 있습니다. 네이슨은 여러 훌륭한 조직, 팀, 오픈소스 커뮤니티에서 학습하고 배운 내용을 공유해 왔습니다. 그는 여러 DORA 보고서의 공동 저자이며, 2020년에 O'Reilly에서 출간한 *97 Things Every Cloud Engineer Should Know*의 기고자이자 편집자였습니다.

네이슨 하비

인구통계 및 기업통계

설문조사 대상

지난 10년 넘게 DORA 연구 프로그램은 실적이 우수한 기술 기반 조직의 역량, 관행, 측정값을 조사해 왔습니다. 규모와 업종을 막론한 수많은 조직에서 일하는 전문가 약 39,000명의 다양한 의견을 들었습니다. 유용한 정보를 공유해 주신 모든 분께 감사드립니다. 올해는 전 세계의 다양한 업계에서 일하는 약 3,000명의 전문가가 경험을 공유하여 실적이 우수한 기술 기반 조직이 성공하는 요인에 대한 이해를 높이는 데 도움을 주었습니다.

올해의 인구통계 및 기업통계 질문은 Stack Overflow에서 완료한 연구를 활용했습니다.

2023년 Stack Overflow 개발자 설문조사에는 90,000명 이상의 응답자가 참여했습니다.[1] 이 설문조사가 모든 기술 실무자를 대상으로 한 것은 아니지만, 개발자 업계의 모든 구성원을 대상으로 한 조사라고 할 수 있을 정도로 많은 인원이 참여했습니다.

이 설문조사에서 얻은 인구에 대한 감각을 통해 데이터에서 응답 편향을 찾아 연구 결과를 어느 정도까지 일반화하고자 하는지 파악했습니다. 또한 Stack Overflow 개발자 설문조사에서 제시된 인구통계 및 기업통계 질문은 훌륭하게 작성되었으며 차용할 가치가 충분합니다.

요컨대, Google의 표본과 Stack Overflow의 표본 사이에는 큰 차이가 없습니다. 즉, 표본이 모집단을 반영한다고 믿을 만한 충분한 이유가 있다는 뜻입니다.

업종

설문조사 응답자들에게 12개 카테고리에 걸쳐 조직이 주로 활동하는 산업 분야를 파악하도록 요청했습니다. 응답자들이 가장 많이 근무하는 분야는 기술(35.69%), 금융 서비스(15.66%), 소매/소비자/전자상거래(9.49%)였습니다.

업종	응답자 비율
기술	35.69%
금융 서비스	15.66%
소매/소비자/전자상거래	9.49%
기타	5.94%
산업 및 제조업	5.49%
의료 및 제약	4.60%
미디어/엔터테인먼트	4.26%
정부 기관	3.89%
교육	3.66%
에너지	3.03%
보험	2.39%
비영리 단체	1%

직원 수

설문조사 응답자에게 9개의 버킷을 사용하여 소속 조직의 직원 수를 파악하도록 요청했습니다. 응답자들이 가장 많이 근무하는 조직은 직원 수 10,000명 이상(24.10%), 직원 수 100~499명(18.50%), 직원 수 1,000~9,999명(15.60%)이었습니다.

조직 규모	비율
1명	2.0%
2~9명	3.2%
10~19명	4.3%
20~99명	14.5%
100~499명	18.5%
500~999명	11.2%
1,000~4,999명	15.6%
5,000~9,999명	6.7%
10,000명 이상	24.1%

장애

장애는 Washington Group Short Set[2]의 안내를 따르는 6가지 기준으로 식별했습니다. 장애에 대한 질문을 한 것은 올해가 5년째입니다. 장애를 보고한 응답자의 비율은 2022년 11%에서 2023년 6%, 2024년 4%로 감소했습니다.

장애	응답자 비율
장애 없음	92%
하나 이상의 장애에 해당	4%
밝히고 싶지 않음	4%

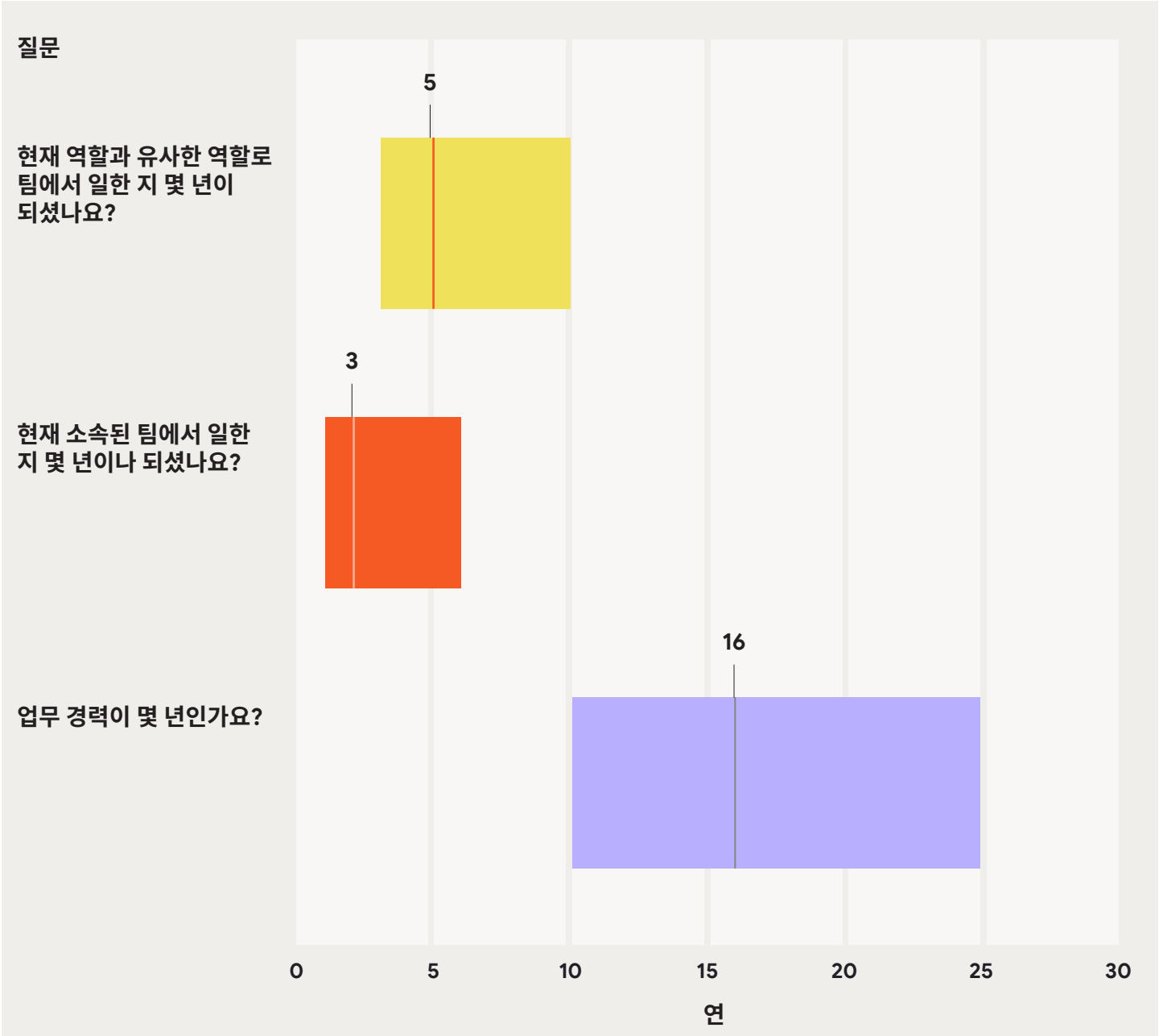
성별

설문조사 응답자에게 성별을 알려달라고 요청했습니다. 응답자의 83%는 남성, 12%는 여성, 1%는 자신의 성별을 스스로 규정하기로 선택했으며 4%는 답변을 거부했습니다.

성별	비율
남성	83%
여성	12%
스스로 규정	1%
응답하고 싶지 않음	4%

경험

설문조사 응답자에게 자신의 역할과 팀에서 근무한 경력을 보고해 달라고 요청했습니다. 응답자의 평균 근무 경력은 16년, 현재 직무 경력은 5년, 현재 소속된 팀에서의 경력은 3년이었습니다.

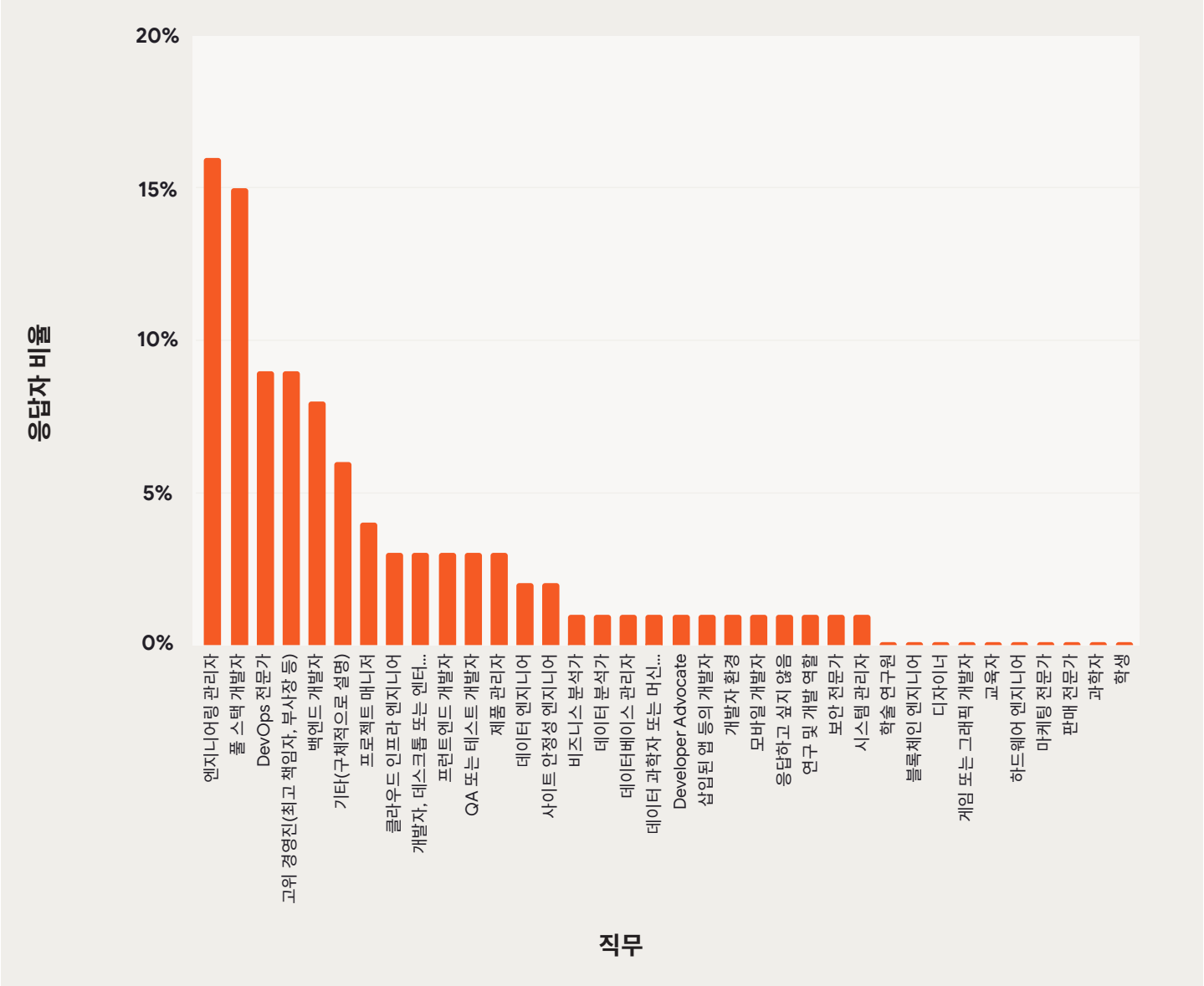


상자 너비는 25번째와 75번째 백분위수를 나타냅니다. 상자를 가로지르는 선은 중앙값을 나타냅니다.

역할

분석에서 일부 개별 직무를 함께 그룹화하여 응답자 중 적은 비율을 차지하는 직무를 분석에 의미 있게 포함할 수 있도록 했습니다. 데이터에는 다음과 같은 다른 카테고리도 다수 포함되었습니다.

- 개발자: 응답자의 29%
- 관리자: 응답자의 23%
- 고위 경영진: 응답자의 9%(2023년 대비 33% 증가)
- 분석 역할: 응답자의 약 5%



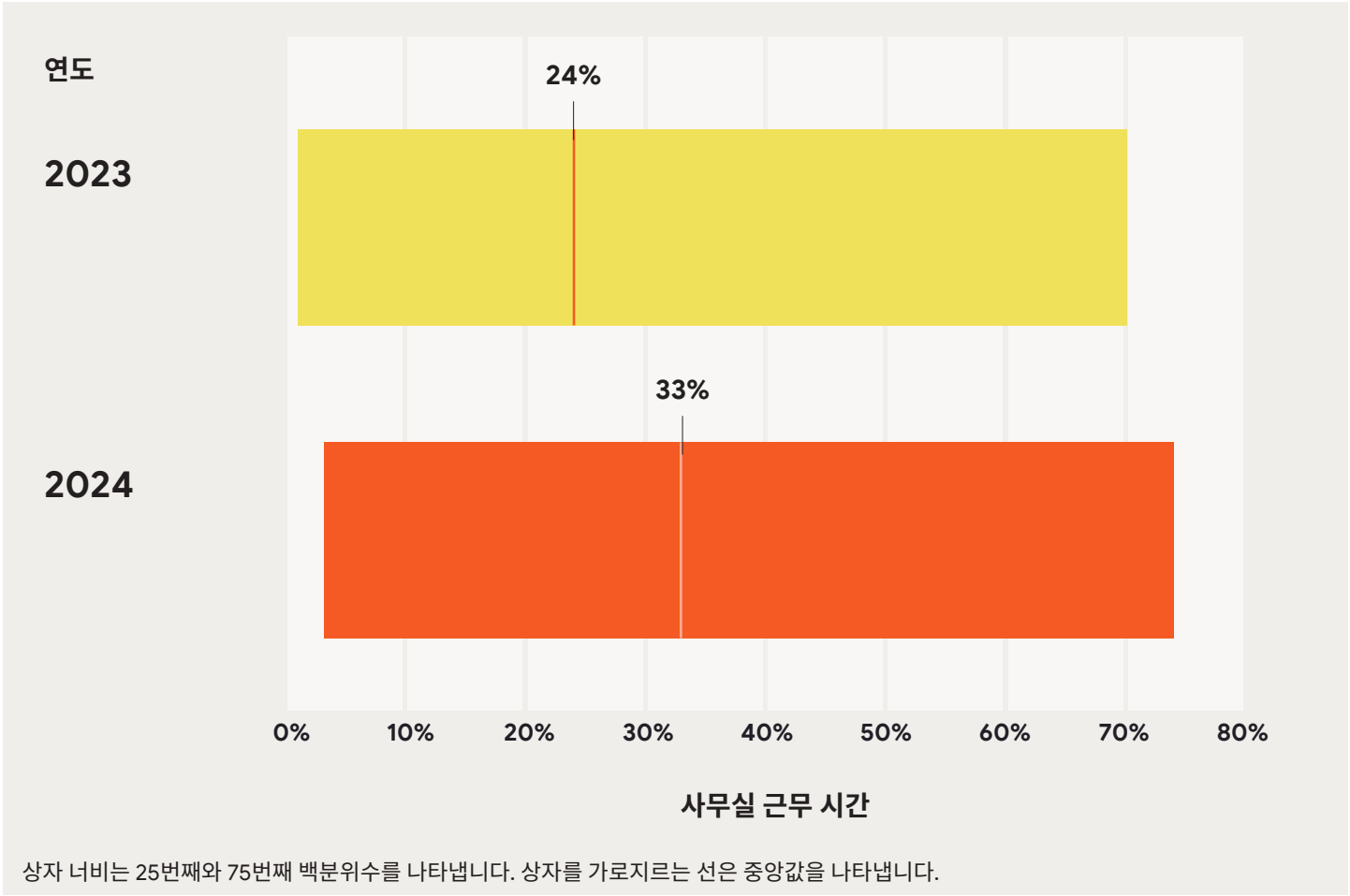
고용 상태

설문조사 응답자에게 현재 고용 상태를 알려달라고 요청했습니다. 응답자의 대다수(90%)는 조직의 정규직 직원이었습니다.

고용 유형	비율
상근 계약직	6%
정규직 직원	90%
시간제 계약직	1%
시간제 정규직	2%

근무 위치

또 한 해 동안 사무실 복귀(RTO)가 추진되었지만, 작년의 패턴은 대체로 유지되었으며, 특히 분포의 꼬리 쪽에서 두드러졌습니다. 중앙값이 37.5% 증가한 것은 하이브리드 근무 또는 적어도 일부 정규 근무가 보편화되고 있음을 시사합니다.



상자 너비는 25번째와 75번째 백분위수를 나타냅니다. 상자를 가로지르는 선은 중앙값을 나타냅니다.

국가

응답자는 104개국에서 참여했습니다. 전 세계 많은 분이 설문조사에 참여해 주시고 계십니다. 감사합니다.

국가					
미국	이탈리아	싱가포르	아이슬란드	룩셈부르크	과테말라
영국	스위스	알바니아	이란	니카라과	홍콩 특별행정구
캐나다	아르헨티나	조지아	요르단	파키스탄	몰타
독일	멕시코	그리스	케냐	페루	모리셔스
일본	포르투갈	필리핀	사우디아라비아	대한민국	모로코
인도	오스트리아	헝가리	슬로바키아	스리랑카	네팔
프랑스	루마니아	세르비아	슬로베니아	튀니지	파라과이
브라질	핀란드	아프가니스탄	태국	안도라	스와질란드
스페인	터키	알제리	우즈베키스탄	바베이도스	시리아 아랍 공화국
오스트레일리아	불가리아	이집트	앙골라	벨리즈	타이완
네덜란드	아일랜드	인도네시아	아르메니아	베냉	구 유고슬라비아 마케도니아 공화국
중국	이스라엘	러시아 연방	보스니아 헤르체고비나	볼리비아	트리니다드 토바고
스웨덴	벨기에	우크라이나	도미니카 공화국	부르키나파소	우루과이
노르웨이	칠레	베트남	에콰도르	코모로스	베네수엘라
뉴질랜드	콜롬비아	방글라데시	에스토니아	코트디부아르	
폴란드	체코	벨라루스	카자흐스탄	엘살바도르	
남아프리카 공화국	말레이시아	코스타리카	라트비아	에티오피아	
덴마크	나이지리아	크로아티아	리투아니아	감비아	

인종 및 민족

설문조사 응답자에게 자신의 인종과 민족을 알려달라고 요청했습니다. 가장 많은 응답자 그룹은 백인(32.4%) 또는 유럽인(22.7%) 이었습니다.

인종 또는 민족	비율
백인	32.4
유럽인	22.7
아시아인	9.9
북미인	4.6
인디언	4.1
밝히고 싶지 않음	4.1
히스패닉 또는 라틴계	3.5
남미인	3.2
동아시아인	2.5
아프리카인	1.8
남아시아인	1.7
다인종	1.5
스스로 규정	1.5
동남아시아인	1.4
흑인	1.3

인종 또는 민족	비율
중동인	1.3
혼혈	0.4
중앙아메리카인	0.4
잘 모르겠음	0.4
북아프리카인	0.4
카리브인	0.2
중앙아시아인	0.2
남아시아인	1.7
민족종교 집단	0.2
태평양 섬 주민	0.2
원주민(예: 아메리카 원주민 또는 호주 원주민)	0.1

1. https://survey.stackoverflow.co/2023/
2. https://www.washingtongroup-disability.com/question-sets/wg-short-set-on-functioning-wg-ss/

방법론

방법론은 데이터를 복제하고 데이터를 생성하고 분석한 방식이 가치 있는 정보를 반환할 가능성이 있는지 판단하는 데 도움이 되는 레시피와 같은 것입니다. 정확한 내용을 다루기에는 지면이 부족하지만 이러한 고려사항에 대해 생각해 볼 좋은 출발점이 되기를 바랍니다.

설문조사 개발

질문 선택

설문조사에 질문을 포함할지 여부를 고려할 때 다음과 같은 측면을 고려합니다.

이 질문은…

- **작업을 이전 작업과 연결하기 위해 설정된 질문인가요?**
- **업계에서 달성하고자 하는 결과(예: 높은 팀 실적)를 반영하고 있나요?**
- **업계에서 리소스 투자를 고려하고 있는 역량(예: AI)을 반영하고 있나요?**
- **사람들이 목표를 달성하는 데 도움이 될 것으로 생각되는 기능(예: 양질의 문서화)을 반영하고 있나요?**
- **표본의 대표성(예: 역할 또는 성별)을 평가하는 데 도움이 되나요?**
- **편향된 경로(예: 코딩 언어 또는 역할)를 차단하는 데 도움이 되나요?**
- **대다수의 응답자가 적어도 어느 정도 정확하게 대답할 수 있는 질문인가요?**

설문조사에 질문을 포함할지 여부를 결정하기 위해 문헌 자료를 검토하고, DORA Community와 교류하고, 인지 인터뷰를 실시하고, 질적 조사를 병행하고, 주제 전문가와 협력하고, 팀 워크숍을 개최합니다.

설문조사 경험

설문조사의 사용성을 개선하기 위해 세심한 주의를 기울이고 있습니다. 설문조사가 특정 기준에 부합하는지 확인하기 위해 인지 인터뷰와 사용성 테스트를 실시합니다.

- **설문조사를 완료하는 데 걸리는 시간은 평균적으로 짧아야 합니다.**
- **설문지에 대한 이해도가 높아야 합니다.**
- **난이도는 상당히 낮아야 하는데, 이는 개념의 기술적 특성을 고려할 때 큰 도전과제입니다.**

데이터 수집

현지화

매년 전 세계 많은 사람이 DORA의 설문조사에 응답했습니다. 올해에는 설문조사를 스페인어(Español), 영어(English), 일본어(日本語), 중국어 간체(简体中文), 포르투갈어(Português), 프랑스어(Français)로 현지화하여 더 넓은 범위의 독자가 접근할 수 있도록 했습니다.

설문조사 응답 수집

응답자 모집에는 다중 채널을 사용합니다. 이러한 채널은 크게 자연 방식과 패널 방식의 2가지 카테고리로 나눌 수 있습니다.

자연 방식은 원하는 소셜 수단을 모두 써서 사람들에게 설문조사에 참여해 달라고 알리는 것입니다. 블로그 게시물을 작성하고 이메일 캠페인을 사용합니다. 그리고 소셜 미디어에 게시물을 작성하고 커뮤니티의 사람들에게도 똑같이 해달라고 요청합니다. 스노볼 효과를 노리기 위해서죠.

패널 방식은 자연 채널을 보완하기 위해 사용합니다. 여기서는 더 광범위한 기술 커뮤니티에서 전통적으로 소수 집단에 속하는 사람들을 모집하려고 합니다. 또한 특정 산업과 조직에서 충분한 응답을 얻고자 합니다.

간단히 말하면 이 방식을 통해 모집 인원을 통제할 수 있습니다. 자연 방식으로는 불가능한 부분을 통제하는 것입니다. 패널 방식을 통해 충분한 응답자를 얻을 수도 있습니다. 자연 방식으로는 분석을 수행하기 위해 충분한 응답을 얻을 수 있을지 모르기 때문입니다. 올해에는 분석을 수행하기에 충분한 자연 방식의 반응이 있었고 패널이 참가자 그룹을 완성하는 데 도움을 주었습니다.

설문조사 흐름

올해는 묻고 싶은 질문이 많았지만 질문할 시간이 부족했습니다. 옵션은 다음과 같습니다.

- **매우 긴 설문조사 만들기**
- **집중할 영역의 하위 집합을 선택하기**
- **사람들을 다양한 주제에 무작위로 배정하기**

관심분야를 포기하고 싶지 않았기 때문에 참가자를 3가지 흐름 중 하나에 무작위로 배정하기로 했습니다. 3가지 흐름이 겹치는 부분이 많았지만 각 흐름은 서로 다른 영역을 심층적으로 파고들었습니다.

다음은 3가지 경로입니다.

- **AI**
- **직장**
- **플랫폼 엔지니어링**

설문조사 분석

측정 검증

설문조사에서 파악하고자 하는 개념은 매우 다양합니다. 참여할 수 있는 언어 게임에는 여러 가지가 있지만 한 가지 공통적인 관점은 이러한 개념의 측정을 변수라고 부른다는 점입니다. 이러한 변수는 DORA의 연구에 포함된 모델의 재료입니다. 이러한 측정의 유효성을 분석하기 위한 방법으로는 내부적인 방법과 외부적인 방법이 있습니다.

측정의 내부적 유효성을 이해하기 위해 개념의 존재에 대한 지표가 되는 대상을 살펴봅니다. 예를 들어 양질의 문서화는 문제를 해결하기 위해 문서를 활용하는 사람의 지표가 될 수 있습니다.

대부분의 변수는 다수의 지표로 구성됩니다. DORA에서 관심을 가지고 지켜보는 구성 개념은 다면적 특성이 있기 때문입니다.

변수의 다면적 특성을 이해하기 위해서는 DORA에서 사용하는 항목이 해당 개념과 얼마나 잘 맞는지 테스트합니다. 잘 맞는 경우 (즉, 높은 수준의 공통 변수를 공유하는 경우), 무언가가 해당 관심 개념의 기저를 이룬다고 가정합니다.

예를 들어 행복은 다면적이라고 볼 수 있습니다. 사람은 누군가가 행복감을 느낄 때 특정한 방식으로 느끼고, 특정한 방식으로 행동하고, 특정한 방식으로 생각하길 기대합니다. DORA에서는 행복이 특정한 패턴의 감정, 생각, 행동에 기반한다고 가정합니다.

따라서 행복이 찾아오면 특정 유형의 감정, 생각, 행동이 동시에 나타날 거라고 기대합니다. 그런 다음 그러한 감정, 생각, 행동에 대한 질문을 합니다. 이들이 실제로도 동시에 나타나는지 테스트하기 위해서는 확인적 요인 분석을 수행합니다.

올해에는 이 분석을 수행하기 위해 Lavaan[1] R 패키지를 활용했습니다. Lavaan은 구성 개념이 실제로 사람들이 질문에 답변하는 방식을 대표하는지 파악할 수 있도록 도와주는 다양한 적합도 통계치를 반환합니다.

개념의 지표가 맞지 않으면 개념을 측정할 신뢰할 만한 방법을 찾지 못한 것이기 때문에 개념을 수정 또는 삭제해야 할 수 있습니다.

구성 개념의 외부적 유효성은 구성 개념이 실제 세계와 얼마나 일치하는지 살펴보는 것으로 확인할 수 있습니다. 하나의 구성 개념은 다른 구성 개념과 특정한 관계를 가질 것으로 기대할 수 있습니다. 행복과 슬픔이라는 두 가지 구성 개념은 음의 상관관계에 있다고 예상할 수 있습니다.

만약 슬픔과 양의 상관관계에 있는 행복 측정값이 도출된다면 측정 결과나 이론에 의문을 품을 것입니다.

마찬가지로 두 가지 구성 개념이 양의 상관관계에 있다고 예상할 수 있으나 관계성이 강하지는 않을 것입니다. 생산성과 직무 만족도는 양의 상관관계가 있을 가능성이 있지만 동일한 개념은 아닙니다. 상관관계가 너무 높으면 동일한 것을 측정하는 것처럼 보일 수 있습니다. 이는 측정값이 두 개념 간의 차이를 식별할 수 있을 만큼 충분히 보정되지 않았거나 가설을 세운 차이가 실제로는 존재하지 않는다는 의미입니다.

모델 평가

일련의 가설을 기본 원칙으로 삼아 세상이 어떻게 작동하는지에 대한 특정 측면을 파악하기 위한 작은 장난감과 같은 가상의 모델을 빌드합니다. 수집한 데이터와 해당 모델이 얼마나 잘 부합하는지 살펴봅니다. 모델을 평가할 때는 절약의 법칙을 사용합니다. 이는 매우 단순한 모델[2]에서 시작하여 복잡성이 더 이상 정당화되지 않을 때까지 복잡성을 추가하는 방식입니다.

예를 들어 조직 실적이 소프트웨어 배포 실적과 운영 실적 간 상호작용의 산물이라고 예측합니다. 이번 연구에 사용한 단순 모델에는 상호작용이 포함되지 않습니다.

```
조직 실적 = 소프트웨어 배포 실적 + 운영 실적
```

두 번째 모델은 상호작용을 추가합니다.

```
조직 실적 = 소프트웨어 배포 실적 + 운영 실적 + 소프트웨어 배포 실적 × 운영 실적
```

'Regression and other stories'[3] 및 'Statistical Rethinking'[4]에서 권장하는 내용에 따라 추가적인 복잡성이 필요한지 확인하기 위해서는 LOOCV(Leave-One-Out 교차 검증)[5] 및 Watanabe–Akaike 정보 기준[6]을 사용합니다.

인과 추론을 위한 방향성 비순환 그래프

검증된 모델은 인과적 사고를 시작하기 위해 알아야 할 사항을 알려줍니다. 인과적 사고의 어려움에 대해서는 아래에서 설명합니다.

인과적으로 이야기하려고 *노력하는* 몇 가지 이유는 다음과 같습니다.

여러분의 질문이 근본적으로 인과관계에 관한 질문이라고 생각합니다. 여러분은 어떤 일을 하는 것이 무언가를 창출할 수 있는지 알고 싶어합니다. 비인과적인 상관관계가 있다고만 생각한다면 무언가를 하기 위해 투자하지 않을 것입니다.

분석 결과는 세상에 대한 인과적 이해에 달려 있습니다. 회귀에서 얻는 실제 수치는 회귀에 무엇을 포함하느냐에 따라 달라집니다. 회귀에 포함할 내용은 데이터가 어떻게 생성되었다고 생각하는지, 즉 인과관계 주장에 따라 달라져야 합니다. 따라서 분명히 해야 합니다.

인과적 사고는 호기심을 일으키고 많은 시간이 투자되는 분야입니다. 세상의 다양한 측면이 어떻게 연결되어 있는지, 그 이유는 무엇인지 궁금해하는 경우가 많습니다. 이들의 인과관계를 생각하기 위해 삶의 모든 측면에 대해 실험할 필요는 없습니다.

인과적 사고는 행동의 핵심이며, 이 보고서가 여러분이 행동을 결정할 때 도움이 되기를 바랍니다.

검증된 모델을 사용하여, 효과를 이해하기 위해 고려해야 할 사항을 알려드릴 수 있습니다. 즉, 차이점이 단 하나뿐인 두 개의 동일한 세계를 만드는 A/B 실험의 형태로 데이터를 얻을 수 있습니다. 이 논리에 따르면 두 세계 사이에 나타나는 모든 차이는 바로 그 초기 차이점에 기인합니다.

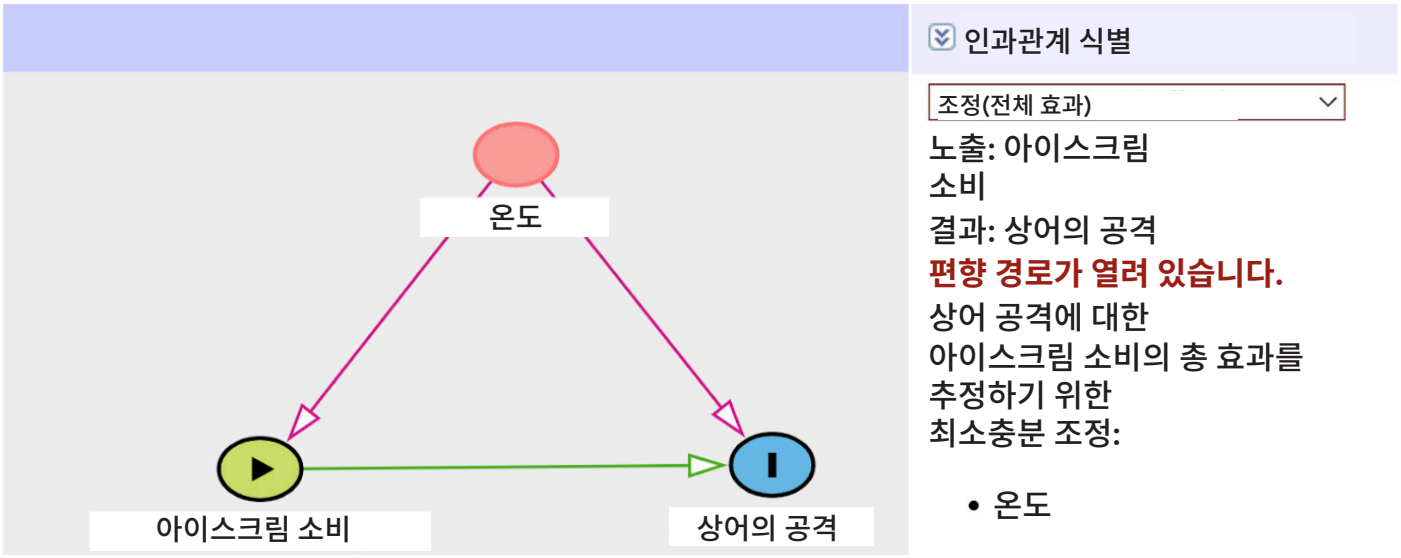
관찰 데이터와 설문조사 데이터에서는 그 차이가 명확하지 않습니다. 참가자마다 다른 점이 많아서 혼란 변수가 발생합니다. 인과 추론 방법은 실험을 모방하려는 시도에서 이러한 차이를 설명하려고 합니다. 즉, 한 가지(예: AI 도입)를 제외한 모든 것을 일정하게 유지하는 것입니다.

상어 공격을 '유발'하는 아이스크림이라는 전형적인 예를 들어 보겠습니다. 이 정보에는 문제가 있습니다. 사람들은 더운 날씨에 아이스크림을 먹는 경향이 있고, 더운 날씨에 해변에 가는 경향이 있습니다. 사람들이 아이스크림을 먹고 해변에 가는 경향이 있는 상황은 사람들이 아이스크림을 먹지 않고 해변에 가지 않는 상황과 다릅니다. 이 데이터는 실험의 논리를 따르지 않습니다. 혼란 변수는 온도입니다.

방향성 비순환 그래프(DAG)는 세상에 존재하는 모든 것 중에서 한 가지를 제외한 모든 것을 일정하게 유지함으로써 실험을 모방하려는 시도를 통해, 세상이 다른 방식을 식별하고 상황을 개선하기 위한 접근 방식을 제공합니다. 아이스크림과 상어 공격의 예에서 DAG가 어떻게 방향을 제시하는지 살펴봅시다. 아이스크림 소비가 상어 공격에 미치는 영향을 정량화하고자 합니다.

모델을 그리고, 이해하고자 하는 효과에 대해 도구에 말하면 도구가 그 효과에 대한 추정치에 편향을 줄 수 있는 요소를 알려줍니다. 이 경우, 도구는 온도를 조정하지 않으면 아이스크림 소비가 상어 공격에 미치는 영향을 추정할 수 없다고 말합니다. 이는 아이스크림 소비량 이외의 모든 것을 동일하게 유지한 다음 아이스크림 소비량에 따라 상어 공격이 계속 변동하는지 확인하는 통계적 접근 방식입니다.

모델에 대한 개요는 모델 장에 나와 있습니다.



이미지 출처: https://www.dagitty.net/dags.html

방향성 비순환 그래프는 특정 효과에 대해 분석할 때 고려해야 할 사항을 알려줍니다.

예를 들어 AI 도입이 생산성에 미치는 영향을 분석할 때 무엇을 고려해야 할까요?

베이지안 통계

이 분석은 베이지안 통계를 사용하여 수행됩니다. 베이지안 통계는 여러 장점이 있습니다.

- **유의미한 것과 그렇지 않은 것을 구분하는 사고방식에서 탈피합니다(10명에게 빈도주의 p-값을 설명해 달라고 하면 10가지 다른 대답을 얻을 수 있습니다).**
- **가설이 제시된 데이터의 확률이 아니라, 데이터가 제공된 가설의 확률을 알고 싶습니다.**
- **사전 지식을 모델에 통합하거나, 적어도 얼마나 모르는지 명확하게 하고자 합니다.**[7]
- **모델링 프로세스의 기본 가정에 맞서야 합니다.**
- **사후 분포를 조사하여 데이터의 규모, 불확실성, 전반적으로 모델이 데이터를 얼마나 잘 이해했는지를 파악할 수 있습니다. 궁극적으로는 데이터를 통해 무엇을 알고 있고, 무엇을 모르고 있는지를 잘 알려줍니다.**
- **매우 통일된 방식으로 많은 통계 문제를 해결하는 유연한 프레임워크입니다.**

'시뮬레이션'이란 무엇을 의미합니까?

데이터는 DORA에서 직접 만드는 것이 아닙니다. DORA는 베이지안 통계를 사용하여 '다른 파라미터 값이 나타날 예상 빈도'를 포착하는 사후 확률을 계산합니다.[8] '시뮬레이션'은 데이터가 주어진 파라미터(평균값, 베타 가중치, 시그마, 절편 등)에 대해 가장 신뢰할 수 있는 값을 찾기 위해 이 사후 확률에서 1,000회 넘게 데이터를 추출하는 과정입니다.

'사후 확률을 0.1, 0.7, 0.5, 1과 같은 파라미터 값으로 가득 찬 버킷이라고 가정해 보겠습니다. 버킷 안에서 각각의 값은 사후 확률에 비례하여 존재합니다. 따라서 최댓값 가까이에 있는 값이 최솟값 가까이에 있는 값보다 훨씬 더 일반적입니다.'[9]

이 모든 과정은 시뮬레이션을 사용해 데이터의 가능한 해석을 탐구하고 불확실성 수준을 파악하는 것과 같습니다. 각각의 시뮬레이션은 데이터 외에는 아무것도 모르는 AI와 정보에 입각한 추측으로 빈칸(파라미터)을 채우려는 몇 가지 규칙으로 생각해 볼 수 있습니다. 이 과정을 4,000번 수행하면 4,000개의 AI가 주어진 파라미터에 대해 추측한 값을 얻게 되는 것입니다.

이러한 추측으로 많은 것을 알 수 있습니다. 평균적인 추측값이 무엇인지, 이러한 추측의 89%[10]가 어떤 값에 속하는지, 얼마나 많은 추측값이 특정 수준 이상인지, 이러한 추측값에 얼마나 많은 변수가 있는지 등을 알게 될 수 있습니다. 여러 모델에 걸친 결합 추측(시뮬레이션)과 같은 흥미로운 작업도 수행할 수 있습니다.

수많은 선 또는 잠재적 값의 분포가 포함된 그래프는 데이터를 고려할 때 가장 타당한 것이 무엇이며 얼마나 많은 불확실성이 존재하는지 표현하는 것입니다.

연구 결과를 커뮤니티에 결합

연구 결과는 기술 중심의 팀과 조직에 의미 있는 관점을 제공하지만, 이는 대화와 공유된 학습을 통해 가장 잘 이해할 수 있습니다. DORA Community에 참여할 때 다양한 통찰력을 얻고, 가정에 도전하며, 이러한 발견을 해석하고 적용하는 새로운 방법을 발견할 수 있습니다.

DORA Community(https://dora.community)에 참여하여 경험을 공유하고, 다른 사람들로부터 배우고, 이러한 권장사항을 실행하는 다양한 접근 방식을 발견하시기 바랍니다. 함께 힘을 합쳐 이러한 인사이트를 활용하는 최선의 방법을 모색하고 조직 내에서 의미 있는 변화를 이끌어낼 수 있습니다.

인터뷰

올해는 연례 설문조사에 심층적이고 반구조화된 인터뷰를 추가하여 정량적 결과를 삼각측량하고, 맥락을 파악하고, 이를 보다 명확하게 했습니다. 인터뷰 가이드는 설문조사에 포함된 주제와 유사하며, Google Meet을 통해 원격으로 진행되는 각 세션의 소요 시간은 약 75분으로 설계되었습니다.

총 11명의 참가자를 인터뷰했는데, 이 참가자들의 프로필은 설문조사에 포함된 기준에 부합했습니다. 모든 인터뷰는 동영상 녹화와 오디오 녹음으로 진행되었습니다. 세션은 57분에서 85분 동안 진행되었으며, 모든 참가자로부터 수집된 데이터의 총시간은 14시간 15분이었습니다. 참가자들의 데이터는 P(N) 형태의 식별자를 사용하여 가명처리되었으며, 여기서 N은 인터뷰 순서에 해당합니다.

모든 인터뷰 내용은 자동화 소프트웨어를 사용하여 기록되었습니다. 설문조사 주제를 사전 코드로 사용하여 스크립트 작성을 수동으로 코딩했습니다. 이 보고서의 최종 출판물에 나오는 인용문은 다시 검토한 후 포함하기 전에 수동으로 스크립트를 작성했습니다. 이 보고서의 저자가 참가자 인용문에 추가한 단어는 괄호([])로 표시하고, 제거한 단어는 생략 부호(..)로 표시했으며, 명확히 하기 위해 필요한 경우에만 편집했습니다.

결과의 추론적 도약

목표는 현실적인 세계의 모습을 표현하는 것입니다. 이 표현을 통해 모두가 일하는 방식을 개선할 수 있습니다. 보이는 바와 같이 복잡성을 단순화하고 있으며, 바로 이 모델의 핵심입니다. 호르헤 루이스 보르게스는 '과학의 정확성에 관하여'라는 아주 짧은 이야기를 썼는데, 그 이야기에서 그는 제국의 지도를 1:1 비율로 만드는 제국에 대해 이야기합니다.[11] 어처구니없는 것은 이로 인해 지도가 완전히 쓸모없게 된다는 것입니다(적어도 제 해석은 그렇습니다). 단순화 작업은 도움이 되어야 합니다.

그렇지만 몇 가지 추론적 도약에 대해서는 분명히 하고 싶습니다.

인과관계

존 스튜어트 밀에 따르면, X가 Y를 야기한다고 말하려면 다음 3가지 조건을 충족해야 합니다.[12]

- **상관관계:** X가 Y와 공존해야 하나요?
- **시간적 선행성:** X는 Y보다 우선해야 하나요?
- 위의 DAG 섹션에서 설명한 대로 편향 경로가 고려되었습니까?

상관관계를 이해할 수 있다고 확신합니다. 이는 주로 표준 통계 절차입니다. 이 설문조사는 특정 시점의 상황을 반영하고 있기 때문에 시간적 선행성은 이론적이며, 데이터의 일부가 아닙니다.

편향 경로의 경우 위에서 구조 방정식 모델과 방향성 비순환 그래프에 대해 언급한 것처럼 편향 경로를 설명하기 위한 작업을 수행하지만, 이는 매우 이론적인 작업이며 시간적 선행성과는 달리 데이터에서 탐색할 수 있다는 의미가 있습니다.

이것은 종단 연구나 적절한 실험을 하지 않았다는 것을 의미합니다. 그럼에도 불구하고 인과적 사고가 세상을 이해하는 방법이라고 생각하고, 인과적 추론에 새롭게 등장하는 기술을 최대한 활용하여 여러분에게 좋은 추정치를 제공하기 위해 최선을 다하고 있습니다. 상관관계는 인과관계를 암시하지는 않지만, 인과관계에 대한 여러분의 생각을 암시합니다.

미시적 수준의 현상 -> 거시적 수준의 현상

대부분의 경우 개별 수준에서 기능을 살펴보고 그것이 상위 수준과 어떻게 연결되는지 확인합니다. 예를 들어 개인별 AI 도입을 애플리케이션이나 서비스 및 팀 실적과 연관 지었습니다. 이는 언뜻 보기에 그리 직관적이지 않습니다. 거시적 수준의 현상이 개별 수준의 현상을 유발하는 이야기는 일반적으로 더 쉽게 설명할 수 있습니다. 인플레이션(거시적)이 계란 구매 여부(미시적)에 영향을 미치는 것이 내가 계란을 사지 않아서 인플레이션이 발생한다는 것보다 더 그럴듯한 이야기처럼 들립니다.

조직의 실적(거시적)이 개인의 웰빙(미시적)에 영향을 미치는 경우에도 마찬가지입니다. 휴리스틱으로 보면 개인이 조직에 미치는 영향력보다 조직이 개인에게 미치는 영향력이 더 클 가능성이 높습니다.

그렇다면 개인의 행동이 팀이나 조직의 성과에 영향을 미친다고 굳이 말해야 하는 이유는 무엇일까요? 완전히 비논리적이지는 않다고 생각하는 추론적 도약을 합니다. 즉, 규모에 따라 다음 내용이 사실일 가능성이 높다고 가정합니다.

즉, 개인이 어떤 일(X)을 할 확률은 그가 X를 하는 조직이나 팀에 속해 있을 때 더 높다고 믿습니다. 따라서 어떤 일을 하는 개인은 X를 하는 경향이 있는 팀과 조직을 대표합니다. 물론 여기서 논란이 꽤 있을 수 있지만 패턴이 나타나고 이 가정이 몇 가지 중요한 능력을 제공할 수 있어야 합니다.

DORA 외부의 예를 들어보겠습니다. 평균 키가 서로 다른 두 나라를 상상해 보겠습니다. 한 나라의 평균 키는 약 167cm입니다. 다른 나라의 평균 키는 약 188cm입니다. 표준 편차는 동일합니다. 각 나라에서 무작위로 한 사람을 뽑는다면 키가 큰 사람이 어느 나라 사람일 가능성이 더 높다고 생각하시나요? 이 작업을 수천 번 반복하면 키가 큰 사람들이 키가 큰 나라를 대표하게 됩니다. 개인의 키는 대략적으로 나라의 키와 비슷할 것입니다.

```
p(X를 하는 개인 | X를 하는 조직) > p(X를 하는 개인 | X를 하지 않는 조직).
```

꼭 필요한 것은 아니지만, 사실인지 확인하기 위해 간단한 시뮬레이션을 실행해 보았습니다.

```r
#R code

#set seed for reproducibility
set.seed(10)

#6'2 and 5'6
height_means = c(6 + 1/6, 5.5)

#constant standard deviation at 1/4 of
foot
std_dev =0.25

#random draws
draws = 1000

#random draws from country A
country_a <-rnorm(draws, mean = height_
means[1], sd = std_dev)

#random draws from country B
country_b <-rnorm(draws, mean = height_
means[2], sd = std_dev)

#how of the draws represent the correct
difference
represented_difference = sum(country_a >
country_b) / 1000

#show results as percentage
represented_difference * 100
```

놀랄 만한 결과가 아닙니다. 1,000번의 무작위 추출 중 97.2%가 올바른 방향이었습니다. 물론 무작위 추출이 아니거나 국가 간 차이가 작고 표본 수가 적으면 잘못 판단하기 쉬울 수 있습니다. 하지만 요점은 거시적 수준에서의 차이가 미시적 수준에서 나타나는 경향이 있다는 것입니다.

1. 로셀 Y(2012년). 'lavaan: An R Package for Structural Equation Modeling.' Journal of Statistical Software, 48(2), 1~36. https://doi.org/10.18637/jss.v048.i02
2. 이 작업에는 잠재적 혼란 변수를 조사하는 작업도 포함됩니다.
3. 겔만, 앤드류, 제니퍼 힐, 아키 베타리. 2021년. Regression and Other Stories. 공증: Cambridge University Press.
4. 리차드 맥엘리스. 2016년. Statistical rethinking: A Bayesian Course with Examples in R and Stan. 공증: CRC Press/Taylor & Francis Group.
5. 겔만, 앤드류, 제니퍼 힐, 아키 베타리. 2021년. Regression and Other Stories. 공증: Cambridge University Press
6. 리차드 맥엘리스. 2016년. Statistical rethinking: A Bayesian Course with Examples in R and Stan. 공증: CRC Press/Taylor & Francis Group.
7. 이전 항목은 약한 경향이 있으며(회의적, 중립적, 부족한 정보) 결과가 이전 항목에 의해 조건화되지 않았는지 확인합니다.
8. 리차드 맥엘리스. Statistical rethinking: A Bayesian course with examples in R and Stan. Chapman and Hall/CRC, 2018, 페이지 50
9. 리차드 맥엘리스. Statistical rethinking: A Bayesian course with examples in R and Stan. Chapman and Hall/CRC, 2018, 페이지 52
10. Statistical Rethinking의 56페이지에서 89%를 선택한 맥엘리스의 추론을 따름. "왜 이런 값을 사용했나요? 이유는 없습니다. 또한 기존의 95% 구간은 많은 독자가 무의식적인 가설 검정을 하도록 유도하므로 이 값은 기존의 95% 구간을 피합니다." 제공하는 구간은 단순히 '모델과 데이터에 적합한 파라미터 값의 범위'를 보여주기 위한 것입니다.
11. 보르헤스, J. L. (1999년). Collected fictions. Penguin.
12. 앤젤라 리 덕워스, 엘리 츠카야마, 헨리 메이. 'Establishing causality using longitudinal hierarchical linear modeling: An illustration predicting achievement from self-control.' Social psychological and personality science 1, 4호(2010년): 311~317.

모델

기존에는 다양한 구조 방정식 모델링 기법(부분 최소자승법, 공분산 기반, 베이지안)을 사용하여 검증한 하나의 거대 모델을 구축했습니다. 2023년 보고서에서는 특정 프로세스를 이해하는 데 도움이 되는 여러 소규모 모델에 집중하는 방식으로 전환했습니다.

예를 들어 품질 문서의 물리적 특성을 이해하기 위해 정교한 모델을 만들었습니다. 특정 효과를 이해하도록 맞춤화된 소규모 모델[1]을 만들면 다음과 같은 중요한 이점이 있습니다.

- **모델 적합성이 떨어지는 영역을 쉽게 파악할 수 있습니다.**
- **모델에 추가하는 모든 것에는 힘이 작용하고 중력을 갖습니다. 모델이 커질수록 변수가 서로에게 힘을 가하는 다양한 방식을 모두 이해하기는 정말 어렵습니다.**
- **허위 관계를 조성하는 것에 대한 컨디셔닝을 방지합니다.[2]**

모델은 어떻게 사용하나요?

질문이 많지만 대부분의 중요한 질문은 다음과 같은 형식일 것입니다.

X를 하면 Y는 어떻게 됩니까?

X는 일반적으로 양질의 문서를 작성하거나 AI를 도입하거나 문화에 투자하는 것과 같은 관행입니다.

Y는 일반적으로 개인 수준(예: 생산성)에서부터 조직 수준(예: 시장점유율)까지 달성하거나 피해야 할 목표입니다.

이러한 형태의 질문을 해결할 목적으로 모델[3]을 구축, 평가, 사용합니다. X를 수행한 후 중요한 결과에 어떤 일이 발생하는지에 대한 정확한 추정치를 제공하기 위해 노력합니다.[4] 효과를 보고할 때는 2가지 중요한 특징을 설명합니다.

1. 효과의 **방향성**에 대해 얼마나 확신할 수 있는지, 즉 이 관행이 유익할지 해로울지 얼마나 확실합니까?

2. 효과의 **규모**를 얼마나 확신할 수 있나요? 특정 관행이 미치는 영향과 이러한 추정치를 둘러싼 불확실성의 정도에 대한 상대적인 의미를 제공할 것입니다.

올해 관심을 끄는 기능 중 몇 가지를 소개합니다.

- **AI 도입**
- **플랫폼 사용**
- **플랫폼 연한**
- **혁신적인 리더십**
- **우선순위 안정성**
- **사용자 중심 전략**

올해의 성과와 결과 그룹의 일부는 다음과 같습니다.

- **개인의 실적 및 웰빙(예: 번아웃)**
- **팀 실적**
- **제품 실적**
- **개발 워크플로(예: 코드베이스 복잡성 및 문서 품질)**
- **소프트웨어 배포 실적**
- **조직 실적**

이러한 결과에 집중하는 이유는 그 자체가 목적이라고 믿기 때문입니다. 물론 이러한 결과 중 일부는 다른 결과보다 더 큰 영향을 미칩니다. 조직 실적과 팀 실적이 소프트웨어 배포 실적과 아무런 관련이 없다는 사실을 알게 되었다면 소프트웨어 배포 실적이 낮더라도 괜찮을 것입니다.

하지만 조직의 실적이 개인의 웰빙에 달려 있지 않더라도 직원의 웰빙을 우선시하는 것이 바람직합니다.

반복 모델

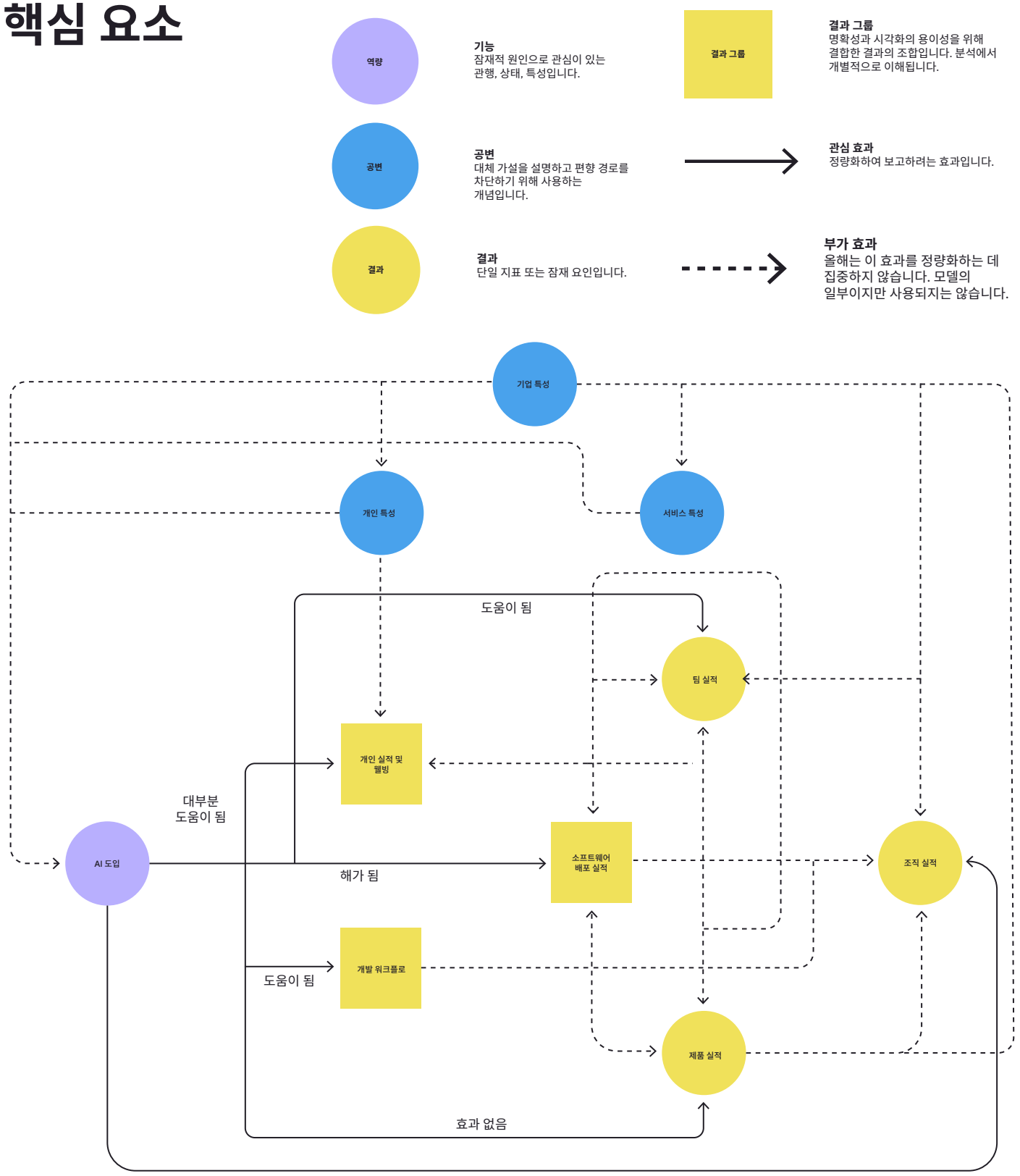
지난 3년 동안 특히 중재와 조정에 관한 세부적인 가설들을 많이 개발하고 탐구했습니다.

올해에는 이러한 유형의 가설에 집중하는 시간을 줄이고 기능이 결과에 미치는 영향을 추정하는 데 더 많은 시간을 할애했습니다. 즉, 각 기능에 대한 모델은 거의 동일합니다.

따라서 AI 도입의 효과에 대한 모델은 사용자 중심 전략의 효과에 대한 모델과 설계상 매우 유사합니다. 모델을 복사하고 기능 이름을 변경할 수도 있지만, 그렇게 해도 크게 유용하지 않을 수 있습니다.

대신 AI 모델만 보여드리지만, 이것이 각 모델의 밑바탕이 되는 도식 또는 형태라는 것을 알고 있습니다. 분석을 실행하는 데 관심이 있다면, DAGitty와 같은 도구에서 이 모델을 구성하면 분석에 사용한 회귀를 거의 비슷하게 재현할 수 있을 것입니다. 즉, 가독성을 위해 표시되는 내용을 약간 단순화했습니다. 또한 각 기능의 모델은 매우 유사하지만 그 효과는 서로 다릅니다. 예를 들어 아래에서 AI 도입은 일반적으로 소프트웨어 배포 실적에 해를 끼치지만 내부 문서 및 사용자 중심 전략 같은 경우에는 그 반대라는 사실을 확인할 수 있으며, 자세한 내용은 각 장을 참조하시기 바랍니다.

핵심 요소



1. 495~496페이지에 있는 겔만 외 여러 명이 작성한 'Regression and other stories'의 B.6 많은 모델에 맞추기 및 B.9 대규모 회귀분석의 부산물이 아닌 타겟팅된 방식의 인과 추론 수행하기 섹션에서 유용한 몇 가지 중요한 팁을 제공합니다.
2. 이에 대한 자세한 논의는 Statistical Rethinking의 6장에서 확인할 수 있습니다. 지금은 그중에서도 충돌변수 편향에 대해 이야기하고 있습니다.
3. 방법론 장에서 이러한 모델이 방향성 비순환 그래프와 어떻게 연결되는지에 대해 설명하는 대화를 참조하세요.
4. 인과관계에 대해서는 방법 장에서 간략하게 설명합니다.

추천 도움말

DORA Community에 참여하여 소프트웨어 배포 및 운영 실적 개선을 주제로 논의하고 배우며 협업해 보세요. https://dora.community

DORA 빠른 점검 수행 https://dora.dev/quickcheck

학습 분위기, 빠른 흐름, 빠른 피드백을 가능하게 하는 기능을 살펴보세요. https://dora.dev/capabilities

생성형 인공지능에 대한 개발자의 신뢰 강화. https://dora.dev/research/2024/trust-in-ai/

도서 읽어보기: *Accelerate: The Science Behind DevOps: Building and scaling high performing technology organizations*. IT Revolution. https://itrevolution.com/product/accelerate

도서 읽어보기: *Team Topologies: Organizing Business and Technology Teams for Fast Flow*. IT Revolution Press. https://teamtopologies.com/

이전 DORA 보고서를 포함한 DORA 연구 프로그램의 간행물. https://dora.dev/publications

연구 및 보고서에 관한 자주 묻는 질문. http://dora.dev/faq

정오표 - 이 보고서에 대한 변경사항, 수정사항, 설명을 읽고 제출하세요. https://dora.dev/publications/errata

2024년 DORA 보고서의 최신 버전인지 확인하세요. https://dora.dev/vc/?v=2024.3