場当たり的改善からの脱却:

IDEALモデル×Four Keysによる科学的プロセス改善

～ Four Keysを組織に定着させる体系的フレームワーク ～

ソフトウェアプロセス改善カンファレンス2025 (SPI Japan 2025)
[Day2: 2025/10/23 木曜日・セッション3A]

ファインディ株式会社 CTO室 高橋 裕之

1989年より組込みエンジニアとして、電話網の交換機開発携わり国産OS CTRONをスクラッチで開発。また、BSD UNIXベースのTCP/IPプロトコル開発に従事。その後、メーカーでRTOSや組込みLinuxを基盤としたテレビ、デジタルカメラ、ビデオカメラなどの開発に16年携わる。
2005年からソフトウェア開発で起きるさまざまな問題に向き合うことを決意しSPI（ソフトウェアプロセス改善）の専門家へ転身。
問題を抱える開発チームに向き合いながら、計測エビデンスをもとにしたケイパビリティモデルに基づくプロセス改善活動を得意とする。
Findy Tech Blog 編集長。

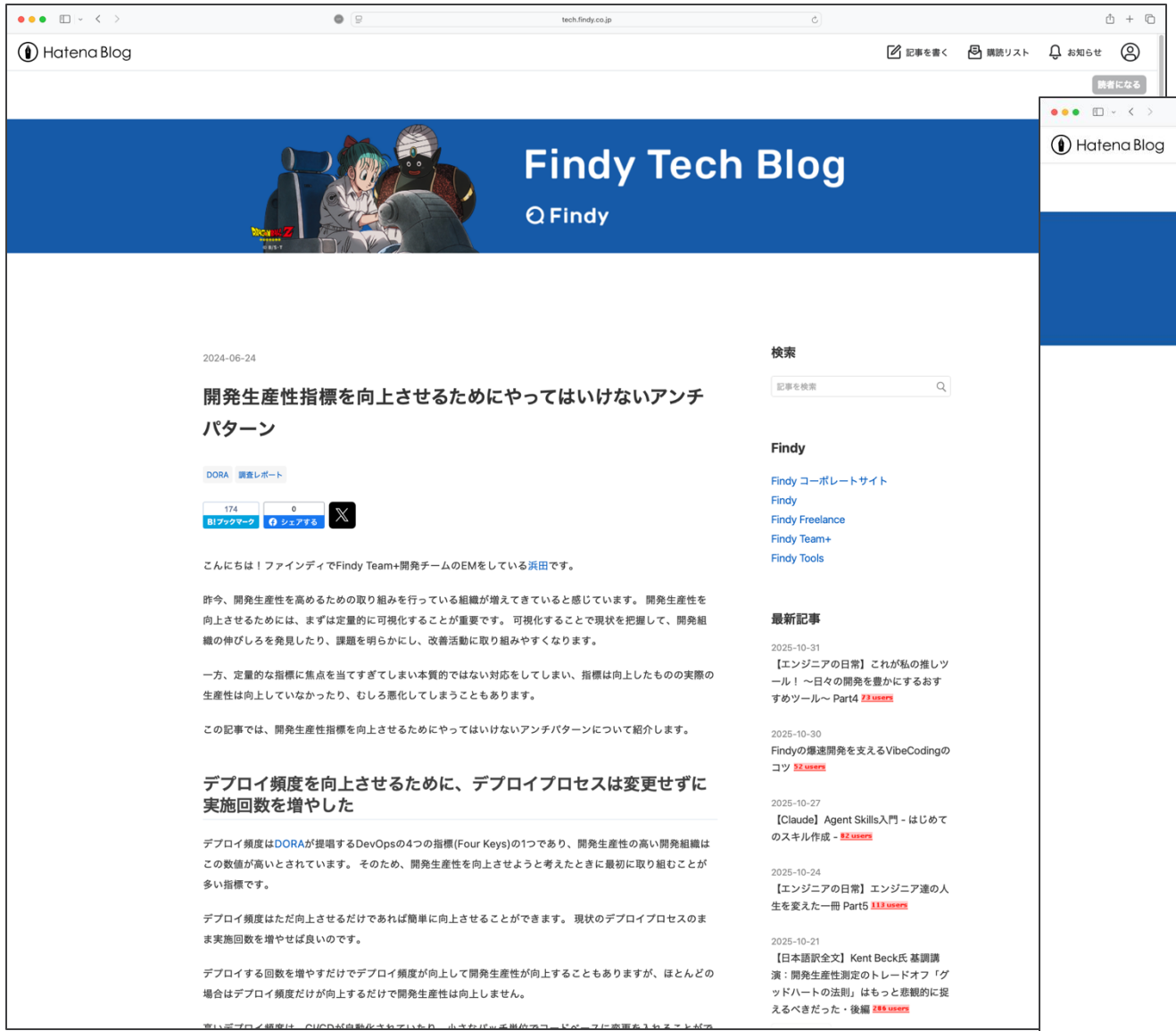
ファインディ株式会社 CTO室
Staff Engineer (Engineering Excellence),
SPI Coach, Agile Coach

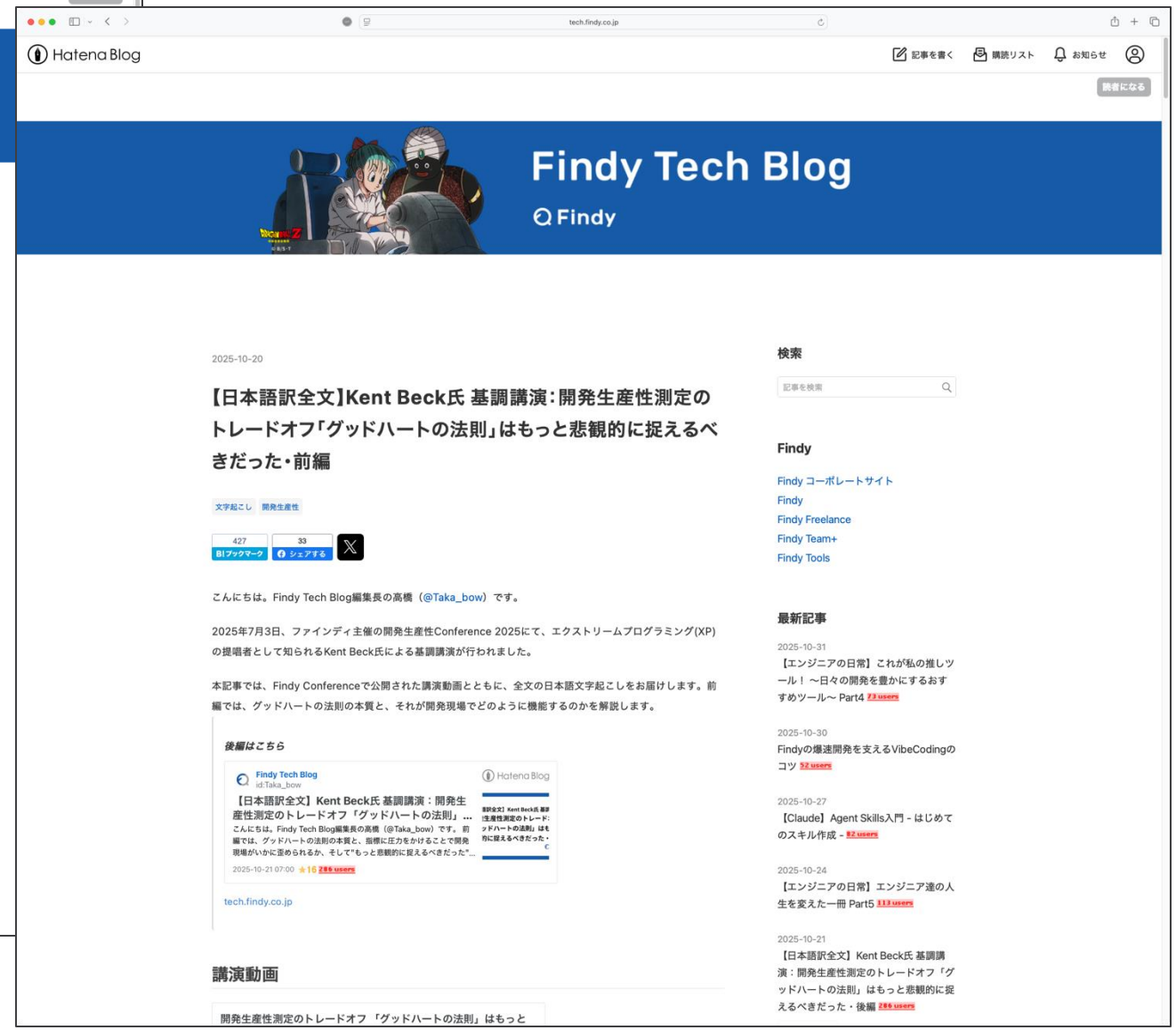
高橋 裕之 / Hiroyuki Takahashi

@Taka_bow
takabow
hiroyukitakah

- 1989年 株式会社ジェーシーイ
- 1993年 フリーランス
- 1995年 株式会社メイテック
- 1996年 日立通信システム株式会社（現・株式会社日立情報通信エンジニアリング）
- 2002年 ソニーデジタルネットワークアプリケーションズ株式会社
  コンシューマ向けカメラ開発、組織横断型プロセス改善（SEPG）に従事
- 2013年 ウイングアーク１ｓｔ株式会社
  プロセス改善コーチ兼アジャイルコーチとして活動。2018年よりソフトウェアプロセス＆品質改善部部長、製品品質管理責任者、オープンソース管理責任者を務める。
- 2021年 株式会社ビズリーチ
  プロセス改善コーチ兼アジャイルコーチとして活動。QAとプロセス改善部門のマネジャーとして、DevOpsアプローチによる開発透明性向上とDORA(Four Keys)メトリクスを用いた開発生産性による改善活動を実施。SODA（Software Outcome Delivery Architecture）構想を立案し、プロダクトの事業影響を定量化・可視化する組織変革をリード。
- 2024年 ファインディ株式会社
  ソフトウェアプロセス改善コーチ兼アジャイルコーチとして活動。
  エンジニア組織への新技術・方法論導入支援のイネーブルメントを担当。

Findy Tech Blog 編集長 - https://tech.findy.co.jp



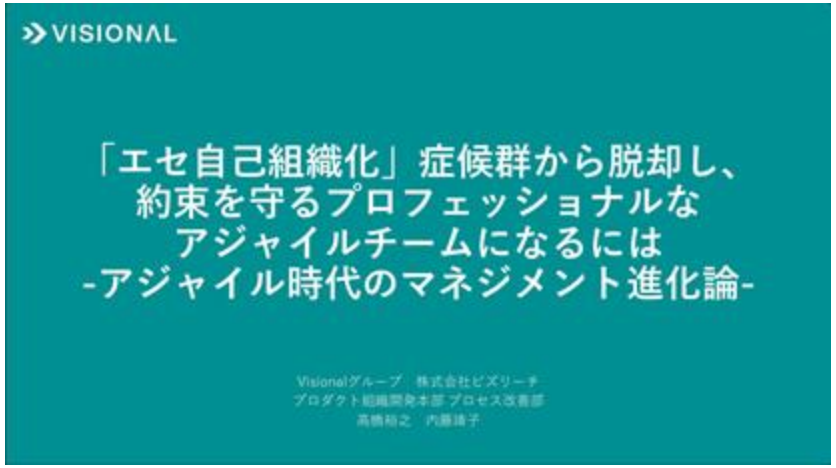


※画像は『ドラゴンボールZ』とのコラボが行われている2025年10月時点のものである © B/S・T

Portfolio

https://speakerdeck.com/takabow/ji-hua-hashu-kufalsedehanakuli-terumofalse

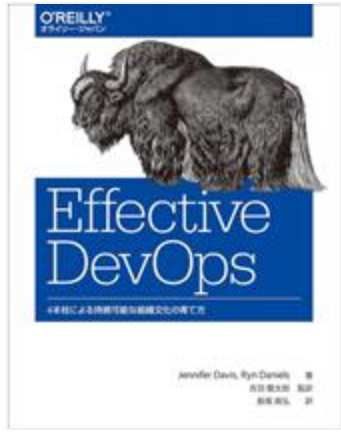
https://speakerdeck.com/takabow/wen-hua-de-fu-zhai-tofalsezhan-i-lao-pu-sohutoueakai-fa-hui-she-deaziyairubian-ge-woshi-gua-keta8nian-jian



https://speakerdeck.com/visional_engineering_and_design/number-rsgt2023

https://speakerdeck.com/visional_engineering_and_design/devopsdays-tokyo-2024

https://speakerdeck.com/takabow/zhi-zao-ye-tosohutoueahaben-dang-nigong-cun-dekiteitanoka-pin-zhi-tosupidowowen-izhi-su-dc76f0ad-2d4c-4f2a-bf4d-cecbceb7eb5e



技術翻訳レビュー

Findy

会社紹介

会社概要

会社名	**ファインディ株式会社 / Findy Inc.**
代表取締役	**山田 裕一朗**
設立	2014 年 2 月　※ 本格的な事業開始は2016年7月
社員数	**376 名**
資本金	**27億5,386 万円**　※ 資本準備金含む
住所	**東京都品川区大崎1-2-2 アートヴィレッジ大崎セントラルタワー 5階**
事業許可番号	13-ユ-308478
サービス	・IT/Webエンジニアの転職サービス「Findy」 ・ハイスキルなフリーランスエンジニア紹介サービス「Findy Freelance」 ・経営と開発現場をつなぐAI戦略支援SaaS「Findy Team+」 ・開発ツールのレビューサイト「Findy Tools」 ・テックカンファレンスのプラットフォーム「Findy Conference」 ・顧客価値を追求する、AI時代の製品開発マネジメント「Findy Insights」等
投資家	**グローバル・ブレイン、ユナイテッド、SMBCベンチャーキャピタル、KDDI、 JA三井リース、みずほキャピタル、博報堂ＤＹベンチャーズ、Carbide Ventures、等**

経営理念

**つくる人がもっとかがやけば、
世界はきっと豊かになる。**

ビジョン

**挑戦するエンジニアの
プラットフォームをつくる。**

Findy

サービス紹介

ファインディが展開するエンジニアプラットフォーム

Findy

Findy Freelance

Findy Team+

Findy Tools

Findy Conference

Findy AI+

Findy AI Career

Findy Insights

挑戦するエンジニアに最新のナレッジを

- 日本に居ながら海外カンファレンス体験を！
  - 開発生産性Conference / アーキテクチャConference / Data Engineering Summit / 内製開発Summit etc...
  - AI Engineering Summit Tokyo

Dr. Nicole Forsgren （開発生産性Conference 2024）


Kent Beck （開発生産性Conference 2025）

Gene Kim （開発生産性Conference 2025）

挑戦するエンジニアに最新のナレッジを

- 日本IT業界に起きている「いまの課題」へ切り込む

Findy

本コンテンツの目的

本日お伝えしたいこと Learning Outcome	• 場当たり的改善から脱却し、改善を確実に成功させる方法を共有 • Four Keys: 何を測るか（科学的測定指標） • IDEALモデル: どう進めるか（体系的フレームワーク） • 統合の実践: 両者を組み合わせた具体的な実践方法と成果
Agenda	• 自己紹介・会社紹介 • 技術的発展の階層構造 • 業界の現状（日本） • 「何を測るべきか」の共通認識の欠如 • ソースコード管理ツールとAI活用の関係 • DORAの成果と影響 • IDEALモデルの再発見 • 実践方法 - 5段階の詳細 • まとめ

ソフトウェア関連の論文・文献

ICTの進化

西暦

主な出来事

1970 1976 1981 1987 1989/02/01 1994/10/31 1996/8 2001/5/18 2003/9/1 2005/10 2009/11/27 2011/7/16 2014/8/18 2017/6/22 2019/9/17 2021/3/15 2023/11/21 2025/6/25

1977 1979 1984 1988/3/1 1995/10/1 2000/12/1 2003/9/1 2005/10/7 2008/12/30 2010/10/12 2013/1/10 2016/10/6 2018/4/1 2019/9/30 2021/12/1 2023/11/28 2024/12/25

1978/5/1 1986 1989 1995 1999 2002 2004/11/16 2010/7/27 2012/3/15 2013/8/1 2018/3/27 2021/8/1 2022/12/17 2024/7/11

1975 1978 1979 1982 1990/10/10 1992/3/13 1996/4/1 2002/11/8 2004/7/16 2006/10/19 2010～ 2011/10/18 2014/4/22 2017/10/14 2018/11/22 2021/12/10 2024/9/13

パソコン通信 | 携帯電話・PHSの普及、インターネット黎明期 | 携帯電話の多機能化、ブロードバンド時代 | スマートフォン普及時代（2010年、モバイル端末利用率がパソコン利用率を超える）

メインフレーム時代：
IBM System 360とVT100（1970～1980）

PC/AT互換機が誕生（1981～）

PC-9800シリーズ（1982～2003）

Apple Macintosh
（1984～）

ワークステーションの時代：
Sun SPARCstation
（1989～1994）

数字送信の開始によるポケベルブーム（日本）
（1992～1996）

テレホーダイ
（1995～2023）

初代iMac
（1998～）

F501i HYPER iモード開始
（1999/2/22）

家庭向けADSL・FTTHの普及、ブロードバンド時代に突入
（2001～）

初代iPhone発表
（2007/1/9）

初代iPad発表
（2010/4/3）

Samsung Galaxy S II
（2011/5/2）

Netflixが日本に上陸(2015/9/2)
※🇺🇸では2007/1にサービスイン

ARMアーキテクチャのSoC
Apple M1 生産（2020）

MacBook Pro M4 Max
（2024）

50年前 40年前 30年前 20年前 10年前

1972 1974 1976 1978 1980 1982 1984 1986 1988 1990 1992 1994 1996 1998 2000 2002 2004 2006 2008 2010 2012 2014 2016 2018 2020 2022 2024 2026

1974年：🇺🇸「TCP/IP」の最初の仕様がRFC 675として公開

1982年～：🇺🇸米国防総省(DoD)は全ての軍用コンピュータ網のためにTCP/IP標準を作成。その後、コンピューター産業に開放され、4.2 BSD UNIXを皮切りにほとんどの商用OSにTCP/IPが実装された。

★ピアソンショック（2013年8月1日）

Point: 米国防総省(DoD)が与えた影響

1980年～：🇺🇸米国防総省(DoD)がウォーターフォールを採用

1980年後半～1990年前半：🇺🇸米国防総省(DoD)の発注したソフトウェアに問題が多発。米会計局でも多くの遅延／途中での挫折が発生

1997～2010年代：🇺🇸米国防総省(DoD)がCMMレベル3以上を防衛関連のソフトウェア調達条件として要求し始める。

2010年～：🇺🇸米国議会は2010会計年度国防権限法(NDAA 2010)において「迅速なITシステム獲得のための新たな取得プロセスの実施」(第804条)をDoDに命じ、事実上アジャイル型開発の原則を法制化

1970～1980年代：🌎多くの企業や政府機関が「ウォーターフォールモデル」を公式な開発プロセスとして採用。
同時に、Royceの思惑と違う、誤解された「ウォーターフォール」が広まる

1990年代：🇺🇸ウォーターフォールのリスクを軽減する開発手法が次々と誕生（インクリメンタル、スパイラル、RUP など）

2000年代～：🇺🇸米国を中心にでアジャイル手法が急速に広まり、ウォーターフォールは特にWeb系・スタートアップ企業ではほぼ使われなくなる。

2000年代～：🇯🇵日本は大企業を中心に、ウォータフォールモデルを採用し続ける。

2020/3～2023/5：🌎COVID-19

1970年代後半～1980年代末：🇯🇵日本車と家電がアメリカ市場を席巻。これが、徐々に日米貿易摩擦の火種に……

1988～：🇺🇸日本を含む諸外国へ「通商法スーパー301条」発動

2000/3～：🇺🇸ITバブル崩壊

2010年以降～：🇯🇵日本のスタートアップはクラウドファーストで事業を立ち上げ、初期からアジャイル手法を採用

1979～：🇺🇸日本の製造業の高品質、ものづくりの強さを研究。

1991～：🇯🇵バブル崩壊「失われた○○年」

2001/2/11：🌎アジャイルソフトウェア開発宣言

2000年代後半～：🌎クラウドファースト・クラウドネイティブ時代に突入

2021～：🌎AI（GenAI）が前提の時代へ

2008/9～：🇺🇸リーマンショック

2018：🇺🇸マイクロソフト、GitHub を 75 億ドルで買収

1980年代後半～1990年代前半：🌎UNIX戦争

1985～1990：🇯🇵国家プロジェクト「Σ計画」が頓挫

2009：🇺🇸Flickrのエンジニアである John Allspaw と Paul Hammond が初めてDevOpsに繋がる伝説的な講演を行う

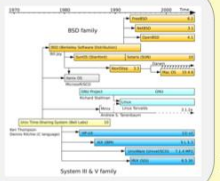
© The Deming Institute


“トリニトロン”
日本製品が欧米で人気


"ウォークマン"
1号機 TPS-L2
（1979）


Toyota Corolla
Liftback SR5 001
（1980～）


世界初のポータブル
CDプレーヤー
D-50（1984）



Java v1.0 正式リリース
Javaはオブジェクト指向と仮想マシン技術を普及させ、後の言語設計にも影響を与えた
（1996）


Bugzilla
リリース
（2000）

JIRA
Jira
リリース
（2002）


Linux 0.01 リリース
UNIXが商業化・断片化していく中、Linuxはオープンソースの力で統一的な開発基盤となり、世界中の技術革新を支えた
（1991/9/17）

Windows95 リリース
コンシューマ向けOSにTCP/IPが標準搭載されワークステーション並みに
（1995）


Subversionリリース
（2000）


CVS 誕生
（1990）

1990～2000：🌎第一次ブラウザ戦争。Internet Explorerの躍進、Netscape Navigatorの消滅

flickr
リリース
（2004/2）

aws
AWS
サービス開始
（2006）

Google Cloud
サービス開始
（2008）

Azure
サービス開始
（2010）

Selenium
（2004）

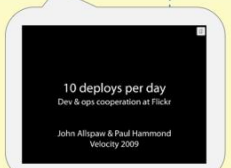
REDMINE
flexible project management
（2006）

trac
（2004）

git
（2005）

GitHub
リリース
（2008）

AUTOMOTIVE SPICE®
欧州の自動車メーカが中心となって公開（2005）



GitLab
（2011）


Jenkins 誕生
（2011）

2009～2020：🌎第二次ブラウザ戦争
Google Chromeの躍進、 Internet Explorerの衰退

GitHub Actions
（2018）

DATADOG
（2018）

Findy Team+
Team+
（2021/10）

Claude 1
（2023/3/14）

Cline
（2024/7）

GitHub Copilot
（2021/6/29）

Gemini 1
（2023/12/6）

ChatGPT
一般公開
（2022/11/30）

Devin
（2024/3/12）

Agile Alliance + Project Management Institute

2025/1/3：🇺🇸Agile AllianceがPMBOKなどを策定するPMIに加盟

ソフトウェア開発現代史年表 Ver2.09

西暦

50年前 / 40年前 / 30年前

1972 1974 1976 1978 1980 1982 1984 1986 1988 1990 1992 1994 1996 1998 2000

主な出来事

1974年：「TCP/IP」の最初の仕様がRFC 675として公開

1982年～：米国防総省(DoD)は全ての軍用コンピュータ網のためにTCP/IP標準を作成。その後、コンピューター産業に開放され、4

Point: 米国防総省(DoD)が与えた影響

1980年～：米国防総省(DoD)がウォーターフォールを採用

1980年後半～1990年前半：米国防総省(DoD)の発注したソフトウェアに問題が多発。米会計局でも多くの遅延／途中での挫折が発生

1997～2010年代：米国
条件として要求し始める。

1970～1980年代：多くの企業や政府機関が「ウォーターフォールモデル」を公式な開発プロセスとして採用。
同時に、Royceの思惑と違う、誤解された「ウォーターフォール」が広まる

1990年代：ウォーターフォールのリスクを軽減する開発手法が次々と誕生（インクリメンタル、スパイラル、RUP など）

200
200

1970年代後半～1980年代末：日本車と家電がアメリカ市場を席巻。これが、徐々に日米貿易摩擦の火種に……

1988～：日本を含む諸外国へ「通商法スーパー301条」発動

2

1979～：日本の製造業の高品質、ものづくりの強さを研究。

1991～：バブル崩壊「失われた○○年」

1980年代後半～1990年代前半：UNIX戦争

1985～1990：国家プロジェクト「Σ計画」が頓挫

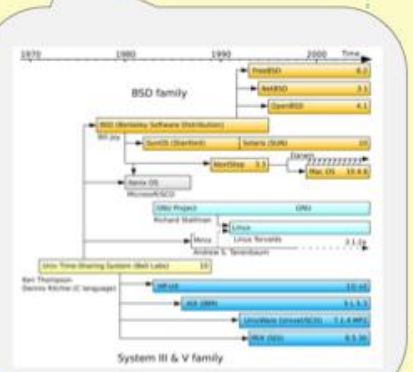
© The Deming Institute


“トリニトロン”
日本製品が欧米で人気


"ウォークマン”
1号機 TPS-L2
（1979）


Toyota Corolla
Liftback SR5 001
（1980～）


世界初のポータブル
CDプレーヤー
D-50（1984）




Java v1.0 正式リリース
Javaはオブジェクト指向と
仮想マシン技術を普及させ、
後の言語設計にも影響を与えた
（1996）


Bugzi
リリー
（200


Linux 0.01 リリース
UNIXが商業化・断片化していく中、
Linuxはオープンソースの力で
統一的な開発基盤となり、
世界中の技術革新を支えた
（1991/9/17）

Windows95 リリース
コンシューマ向けOSに
TCP/IPが標準搭載され
ワークステーション並みに
（1995）


CVS 誕生
（1990）

Subversion
（200

1990～2000：第一次ブラウザ戦争。Internet Explorerの躍進、
Netscape Navigatorの消滅

ソフトウェア開発現代史年表 Ver2.09

20年前 10年前

1998 2000 2002 2004 2006 2008 2010 2012 2014 2016 2018 2020 2022 2024 2026

ューター産業に開放され、4.2 BSD UNIXを皮切りにほとんどの商用OSにTCP/IPが実装された。

★ピアソンショック（2013年8月1日）

1997～2010年代：米国防総省(DoD)がCMMレベル3以上を防衛関連のソフトウェア調達条件として要求し始める。

2010年～：米国議会は2010会計年度国防権限法(NDAA 2010)において「迅速なITシステム獲得のための新たな取得プロセスの実施」(第804条)をDoDに命じ、事実上アジャイル型開発の原則を法制化 *

る開発手法が次々

2000年代～：米国を中心にでアジャイル手法が急速に広まり、ウォーターフォールは特にWeb系・スタートアップ企業ではほぼ使われなくなる。

2000年代～：日本は大企業を中心に、ウォータフォールモデルを採用し続ける。

2020/3～2023/5：COVID-19

2000/3～：ITバブル崩壊

2010年以降～：日本のスタートアップはクラウドファーストで事業を立ち上げ、初期からアジャイル手法を採用

2001/2/11：アジャイルソフトウェア開発宣言

2000年代後半～：クラウドファースト・クラウドネイティブ時代に突入

2021～：AI（GenAI）が前提の時代へ

2008/9～：リーマンショック

2018：マイクロソフト、GitHub を 75 億ドルで買収

2009：FlickrのエンジニアであるJohn AllspawとPaul Hammondが初めてDevOpsに繋がる伝説的な講演を行う

flickr リリース (2004/2)

AWS サービス開始 (2006)

Google Cloud サービス開始 (2008)

Azure サービス開始 (2010)

Findy Team+ Team+ (2021/10)

Claude 1 (2023/3/14)

Cline (2024/7)

式リリース
ジェクト指向と
技術を普及させ、
計にも影響を与えた

Bugzilla リリース (2000)

Jira リリース (2002)

Selenium (2004)

REDMINE flexible project management (2006)

trac (2004)

GitHub Copilot (2021/6/29)

Gemini 1 (2023/12/6)

GitLab (2011)

GitHub Actions (2018)

ChatGPT 一般公開 (2022/11/30)

Devin (2024/3/12)

GitHub リリース (2008)

git (2005)

Subversionリリース (2000)

Jenkins 誕生 (2011)

DATADOG (2018)

AUTOMOTIVE SPICE®

欧州の自動車メーカが中心となって公開（2005）

Agile Alliance + Project Management Institute.

lorerの躍進、

2009～2020：第二次ブラウザ戦争
Google Chromeの躍進、 Internet Explorerの衰退

2025/1/3：Agile AllianceがPMBOKなどを策定するPMIに加盟

ソフトウェア関連の論文・文献

ICTの進化

西暦

1970

1976

1981

1987

1989/02/01

1994/10/31

1996/8

2000

1977

1979

1984

1988/3/1

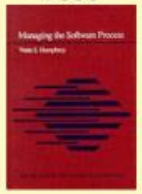
1995/10/1

2000/12/1

1978/5/1

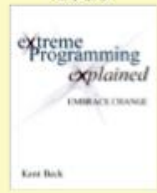
1986

1989

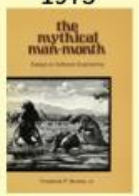


1995

1999



1975



1978

1979

1982

1990/10/10

1992/3/13

1996/4/1

パソコン通信

携帯電話・PHSの普及、インターネッ

メインフレーム時代：
IBM System 360とVT100（1970～1980）

PC/AT互換機が誕生（1981～）

PC-9800シリーズ（1982～2003）

Apple Macintosh
（1984～）

ワークステーションの時代：
Sun SPARCstation
（1989～1994）

数字送信の開始による
ポケベルブーム（日本）
（1992～1996）

テレホーダイ
（1995～2023）

初代iMac
（1998～）

F501i HYPER
iモード開始
（1999/2/22）

50年前　40年前　30年前

1972 1974 1976 1978 1980 1982 1984 1986 1988 1990 1992 1994 1996 1998 2000


2001/5/18
2003/9/1
2005/10
2009/11/27
2011/7/16
2014/8/18
2017/6/22
2019/9/17
2021/3/15
2023/11/21
2025/6/25
2000/12/1
2003/9/1
2005/10/7
2008/12/30
2010/10/12
2013/1/10
2016/10/6
2018/4/1
2019/9/30
2021/12/1
2023/11/28
2024/12/25
1999
2002
2004/11/16
2010/7/27
2012/3/15
2013/8/1
2018/3/27
2021/8/1
2022/12/17
2024/7/11
2002/11/8
2004/7/16
2006/10/19
2010〜
2011/10/18
2014/4/22
2017/10/14
2018/11/22
2021/12/10
2024/9/13
の普及、インターネット黎明期
携帯電話の多機能化、ブロードバンド時代
スマートフォン普及時代（2010年、モバイル端末利用率がパソコン利用率を超える）
初代iMac
(1998〜)
F501i HYPER
iモード開始
(1999/2/22)
家庭向けADSL・FTTHの普及、
ブロードバンド時代に突入
(2001〜)
初代iPhone発表
(2007/1/9)
初代iPad発表
(2010/4/3)
Samsung
Galaxy S II
(2011/5/2)
NETFLIX
Netflixが日本に上陸(2015/9/2)
※🇺🇸では2007/1にサービスイン
ARMアーキテクチャのSoC
Apple M1 生産（2020）
MacBook Pro M4 Max
(2024)
20年前
10年前
1998
2000
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022
2024
2026

ソフトウェア関連の論文・文献

ICTの進化

西暦

主な出来事

1970 1976 1981 1987 1989/02/01 1994/10/31 1996/8 2001/5/18 2003/9/1 2005/10 2009/11/27 2011/7/16 2014/8/18 2017/6/22 2019/9/17 2021/3/15 2023/11/21 2025/6/25

1977 1979 1984 1988/3/1 1995/10/1 2000/12/1 2003/9/1 2005/10/7 2008/12/30 2010/10/12 2013/1/10 2016/10/6 2018/4/1 2019/9/30 2021/12/1 2023/11/28 2024/12/25

1978/5/1 1986 1989 1995 1999 2002 2004/11/16 2010/7/27 2012/3/15 2013/8/1 2018/3/27 2021/8/1 2022/12/17 2024/7/11

1975 1978 1979 1982 1990/10/10 1992/3/13 1996/4/1 2002/11/8 2004/7/16 2006/10/19 2010～ 2011/10/18 2014/4/22 2017/10/14 2018/11/22 2021/12/10 2024/9/13

パソコン通信 | 携帯電話・PHSの普及、インターネット黎明期 | 携帯電話の多機能化、ブロードバンド時代 | スマートフォン普及時代（2010年、モバイル端末利用率がパソコン利用率を超える）

メインフレーム時代：
IBM System 360とVT100（1970～1980）

PC/AT互換機が誕生（1981～）

PC-9800シリーズ（1982～2003）

Apple Macintosh
（1984～）

ワークステーションの時代：
Sun SPARCstation
（1989～1994）

数字送信の開始によるポケベルブーム（日本）
（1992～1996）

テレホーダイ
（1995～2023）

初代iMac
（1998～）

F501i HYPER iモード開始
（1999/2/22）

家庭向けADSL・FTTHの普及、ブロードバンド時代に突入
（2001～）

初代iPhone発表
（2007/1/9）

初代iPad発表
（2010/4/3）

Samsung Galaxy S II
（2011/5/2）

Netflixが日本に上陸(2015/9/2)
※🇺🇸では2007/1にサービスイン

ARMアーキテクチャのSoC
Apple M1 生産（2020）

MacBook Pro M4 Max
（2024）

50年前 40年前 30年前 20年前 10年前

1972 1974 1976 1978 1980 1982 1984 1986 1988 1990 1992 1994 1996 1998 2000 2002 2004 2006 2008 2010 2012 2014 2016 2018 2020 2022 2024 2026

1974年：🇺🇸「TCP/IP」の最初の仕様がRFC 675として公開

1982年～：🇺🇸米国防総省(DoD)は全ての軍用コンピュータ網のためにTCP/IP標準を作成。その後、コンピューター産業に開放され、4.2 BSD UNIXを皮切りにほとんどの商用OSにTCP/IPが実装された。

★ピアソンショック（2013年8月1日）

Point: 米国防総省(DoD)が与えた影響

1980年～：🇺🇸米国防総省(DoD)がウォーターフォールを採用

1980年後半～1990年前半：🇺🇸米国防総省(DoD)の発注したソフトウェアに問題が多発。米会計局でも多くの遅延／途中での挫折が発生

1997～2010年代：🇺🇸米国防総省(DoD)がCMMレベル3以上を防衛関連のソフトウェア調達条件として要求し始める。

2010年～：🇺🇸米国議会は2010会計年度国防権限法(NDAA 2010)において「迅速なITシステム獲得のための新たな取得プロセスの実施」(第804条)をDoDに命じ、事実上アジャイル型開発の原則を法制化

1970～1980年代：🌎多くの企業や政府機関が「ウォーターフォールモデル」を公式な開発プロセスとして採用。
同時に、Royceの思惑と違う、誤解された「ウォーターフォール」が広まる

1990年代：🇺🇸ウォーターフォールのリスクを軽減する開発手法が次々と誕生（インクリメンタル、スパイラル、RUP など）

2000年代～：🇺🇸米国を中心にでアジャイル手法が急速に広まり、ウォーターフォールは特にWeb系・スタートアップ企業ではほぼ使われなくなる。

2000年代～：🇯🇵日本は大企業を中心に、ウォータフォールモデルを採用し続ける。

2020/3～2023/5：🌎COVID-19

1970年代後半～1980年代末：🇯🇵日本車と家電がアメリカ市場を席巻。これが、徐々に日米貿易摩擦の火種に……

1988～：🇺🇸日本を含む諸外国へ「通商法スーパー301条」発動

2000/3～：🇺🇸ITバブル崩壊

2010年以降～：🇯🇵日本のスタートアップはクラウドファーストで事業を立ち上げ、初期からアジャイル手法を採用

1979～：🇺🇸日本の製造業の高品質、ものづくりの強さを研究。

1991～：🇯🇵バブル崩壊「失われた○○年」

2001/2/11：🌎アジャイルソフトウェア開発宣言

2000年代後半～：🌎クラウドファースト・クラウドネイティブ時代に突入

2021～：🌎AI（GenAI）が前提の時代へ

2008/9～：🇺🇸リーマンショック

2018：🇺🇸マイクロソフト、GitHub を 75 億ドルで買収

1980年代後半～1990年代前半：🌎UNIX戦争

1985～1990：🇯🇵国家プロジェクト「Σ計画」が頓挫

2009：🇺🇸Flickrのエンジニアである John Allspaw と Paul Hammond が初めてDevOpsに繋がる伝説的な講演を行う

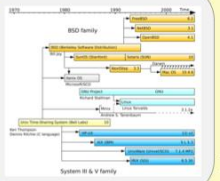
© The Deming Institute


“トリニトロン”
日本製品が欧米で人気


"ウォークマン"
1号機 TPS-L2
（1979）


Toyota Corolla
Liftback SR5 001
（1980～）


世界初のポータブル
CDプレーヤー
D-50（1984）



Java v1.0 正式リリース
Javaはオブジェクト指向と仮想マシン技術を普及させ、後の言語設計にも影響を与えた
（1996）


Bugzilla
リリース
（2000）

JIRA
Jira
リリース
（2002）


Linux 0.01 リリース
UNIXが商業化・断片化していく中、Linuxはオープンソースの力で統一的な開発基盤となり、世界中の技術革新を支えた
（1991/9/17）

Windows95 リリース
コンシューマ向けOSにTCP/IPが標準搭載されワークステーション並みに
（1995）


Subversionリリース
（2000）


CVS 誕生
（1990）

1990～2000：🌎第一次ブラウザ戦争。Internet Explorerの躍進、Netscape Navigatorの消滅

flickr
リリース
（2004/2）

aws
AWS
サービス開始
（2006）

Google Cloud
サービス開始
（2008）

Azure
サービス開始
（2010）

Selenium
（2004）

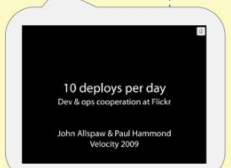
REDMINE
flexible project management
（2006）

trac
（2004）

git
（2005）

GitHub
リリース
（2008）

AUTOMOTIVE SPICE®
欧州の自動車メーカが中心となって公開（2005）



GitLab
（2011）


Jenkins 誕生
（2011）

2009～2020：🌎第二次ブラウザ戦争
Google Chromeの躍進、 Internet Explorerの衰退

GitHub Actions
（2018）

DATADOG
（2018）

Findy Team+
Team+
（2021/10）

Claude 1
（2023/3/14）

Cline
（2024/7）

GitHub Copilot
（2021/6/29）

Gemini 1
（2023/12/6）

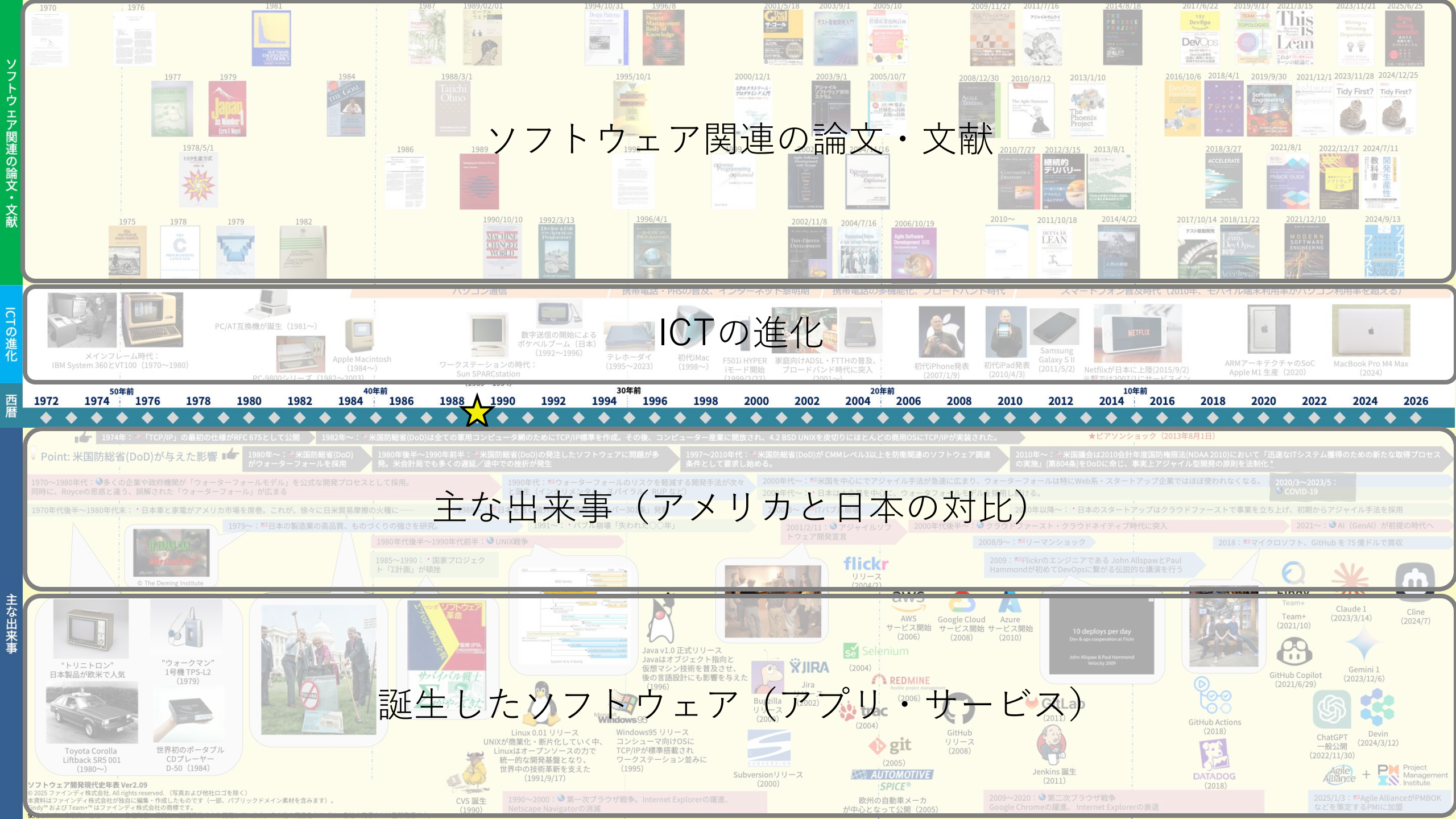
ChatGPT
一般公開
（2022/11/30）

Devin
（2024/3/12）

Agile Alliance + Project Management Institute

2025/1/3：🇺🇸Agile AllianceがPMBOKなどを策定するPMIに加盟

ソフトウェア開発現代史年表 Ver2.09


ソフトウェア関連の論文・文献
ICTの進化
主な出来事（アメリカと日本の対比）
誕生したソフトウェア（アプリ・サービス）
西暦
1972
1974
1976
1978
1980
1982
1984
1986
1988
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022
2024
2026
50年前
40年前
30年前
20年前
10年前

（参考）ソフトウェア開発現代史シリーズ https://speakerdeck.com/takabow

2024/8/8

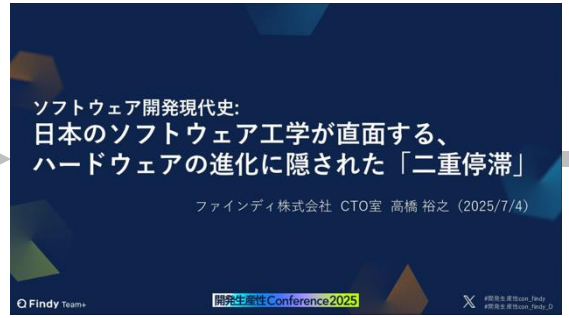
お客様向け資料(非公開)
「モダンソフトウェア
エンジニアリングとは」

■モダンなソフトウェアエンジニアリングを導入することの具体的なメリットを把握する

■戦後日本の製造業とソフトウェアエンジニアリングの歴史を振り返り、世界と日本の進化になぜ差ついたのかを理解する

■なぜ日本のソフトウェア開発は「ウォーターフォール」を受け入れ続けているのか？の謎に迫る。

■DevOpsが未定着な日本の謎とDORA10年の歴史を振り返り、ソフトウェア開発マネージメントに持ち込まれた「科学的」とは何か？を解説



■日米のソフトウェア産業を比較し、日本特有の停滞要因を歴史的・組織的視点から分析

Findy

Findy

DORA Metrics（Four Keys）とは

ソフトウェア関連の論文・文献

ICTの進化

西暦

主な出来事

1970 1976 1981 1987 1989/02/01 1994/10/31 1996/8 2001/5/18 2003/9/1 2005/10 2009/11/27 2011/7/16 2014/8/18 2017/6/22 2019/9/17 2021/3/15 2023/11/21 2025/6/25

1977 1979 1984 1988/3/1 1995/10/1 2000/12/1 2003/9/1 2005/10/7 2008/12/30 2010/10/12 2013/1/10 2016/10/6 2018/4/1 2019/9/30 2021/12/1 2023/11/28 2024/12/25

1978/5/1 1986 1989 1995 1999 2002 2004/11/16 2010/7/27 2012/3/15 2013/8/1 2018/3/27 2021/8/1 2022/12/17 2024/7/11

1975 1978 1979 1982 1990/10/10 1992/3/13 1996/4/1 2002/11/8 2004/7/16 2006/10/19 2010～ 2011/10/18 2014/4/22 2017/10/14 2018/11/22 2021/12/10 2024/9/13

パソコン通信 | 携帯電話・PHSの普及、インターネット黎明期 | 携帯電話の多機能化、ブロードバンド時代 | スマートフォン普及時代（2010年、モバイル端末利用率がパソコン利用率を超える）

メインフレーム時代：
IBM System 360とVT100（1970～1980）

PC/AT互換機が誕生（1981～）

PC-9800シリーズ（1982～2003）

Apple Macintosh
（1984～）

ワークステーションの時代：
Sun SPARCstation
（1989～1994）

数字送信の開始によるポケベルブーム（日本）
（1992～1996）

テレホーダイ
（1995～2023）

初代iMac
（1998～）

F501i HYPER iモード開始
（1999/2/22）

家庭向けADSL・FTTHの普及、ブロードバンド時代に突入
（2001～）

初代iPhone発表
（2007/1/9）

初代iPad発表
（2010/4/3）

Samsung Galaxy S II
（2011/5/2）

Netflixが日本に上陸(2015/9/2)
※🇺🇸では2007/1にサービスイン

ARMアーキテクチャのSoC
Apple M1 生産（2020）

MacBook Pro M4 Max
（2024）

50年前 40年前 30年前 20年前 10年前

1972 1974 1976 1978 1980 1982 1984 1986 1988 1990 1992 1994 1996 1998 2000 2002 2004 2006 2008 2010 2012 2014 2016 2018 2020 2022 2024 2026

1974年：🇺🇸「TCP/IP」の最初の仕様がRFC 675として公開

1982年～：🇺🇸米国防総省(DoD)は全ての軍用コンピュータ網のためにTCP/IP標準を作成。その後、コンピューター産業に開放され、4.2 BSD UNIXを皮切りにほとんどの商用OSにTCP/IPが実装された。

★ピアソンショック（2013年8月1日）

Point: 米国防総省(DoD)が与えた影響

1980年～：🇺🇸米国防総省(DoD)がウォーターフォールを採用

1980年後半～1990年前半：🇺🇸米国防総省(DoD)の発注したソフトウェアに問題が多発。米会計局でも多くの遅延／途中での挫折が発生

1997～2010年代：🇺🇸米国防総省(DoD)がCMMレベル3以上を防衛関連のソフトウェア調達条件として要求し始める。

2010年～：🇺🇸米国議会は2010会計年度国防権限法(NDAA 2010)において「迅速なITシステム獲得のための新たな取得プロセスの実施」(第804条)をDoDに命じ、事実上アジャイル型開発の原則を法制化

1970～1980年代：🌎多くの企業や政府機関が「ウォーターフォールモデル」を公式な開発プロセスとして採用。
同時に、Royceの思惑と違う、誤解された「ウォーターフォール」が広まる

1990年代：🇺🇸ウォーターフォールのリスクを軽減する開発手法が次々と誕生（インクリメンタル、スパイラル、RUP など）

2000年代～：🇺🇸米国を中心にでアジャイル手法が急速に広まり、ウォーターフォールは特にWeb系・スタートアップ企業ではほぼ使われなくなる。

2000年代～：🇯🇵日本は大企業を中心に、ウォータフォールモデルを採用し続ける。

2020/3～2023/5：🌎COVID-19

1970年代後半～1980年代末：🇯🇵日本車と家電がアメリカ市場を席巻。これが、徐々に日米貿易摩擦の火種に……

1988～：🇺🇸日本を含む諸外国へ「通商法スーパー301条」発動

2000/3～：🇺🇸ITバブル崩壊

2010年以降～：🇯🇵日本のスタートアップはクラウドファーストで事業を立ち上げ、初期からアジャイル手法を採用

1979～：🇺🇸日本の製造業の高品質、ものづくりの強さを研究。

1991～：🇯🇵バブル崩壊「失われた○○年」

2001/2/11：🌎アジャイルソフトウェア開発宣言

2000年代後半～：🌎クラウドファースト・クラウドネイティブ時代に突入

2021～：🌎AI（GenAI）が前提の時代へ

2008/9～：🇺🇸リーマンショック

2018：🇺🇸マイクロソフト、GitHub を 75 億ドルで買収

1980年代後半～1990年代前半：🌎UNIX戦争

1985～1990：🇯🇵国家プロジェクト「Σ計画」が頓挫

2009：🇺🇸Flickrのエンジニアである John Allspaw と Paul Hammond が初めてDevOpsに繋がる伝説的な講演を行う

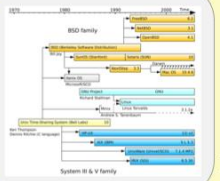
© The Deming Institute


“トリニトロン”
日本製品が欧米で人気


"ウォークマン"
1号機 TPS-L2
（1979）


Toyota Corolla
Liftback SR5 001
（1980～）


世界初のポータブル
CDプレーヤー
D-50（1984）



Java v1.0 正式リリース
Javaはオブジェクト指向と仮想マシン技術を普及させ、後の言語設計にも影響を与えた
（1996）


Bugzilla
リリース
（2000）

JIRA
Jira
リリース
（2002）


Linux 0.01 リリース
UNIXが商業化・断片化していく中、Linuxはオープンソースの力で統一的な開発基盤となり、世界中の技術革新を支えた
（1991/9/17）

Windows95 リリース
コンシューマ向けOSにTCP/IPが標準搭載されワークステーション並みに
（1995）


Subversionリリース
（2000）


CVS 誕生
（1990）

1990～2000：🌎第一次ブラウザ戦争。Internet Explorerの躍進、Netscape Navigatorの消滅

flickr
リリース
（2004/2）

aws
AWS
サービス開始
（2006）

Google Cloud
サービス開始
（2008）

Azure
サービス開始
（2010）

Selenium
（2004）

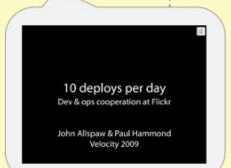
REDMINE
flexible project management
（2006）

trac
（2004）

git
（2005）

GitHub
リリース
（2008）

AUTOMOTIVE SPICE®
欧州の自動車メーカが中心となって公開（2005）



GitLab
（2011）


Jenkins 誕生
（2011）

2009～2020：🌎第二次ブラウザ戦争
Google Chromeの躍進、 Internet Explorerの衰退

GitHub Actions
（2018）

DATADOG
（2018）

Findy Team+
Team+
（2021/10）

Claude 1
（2023/3/14）

Cline
（2024/7）

GitHub Copilot
（2021/6/29）

Gemini 1
（2023/12/6）

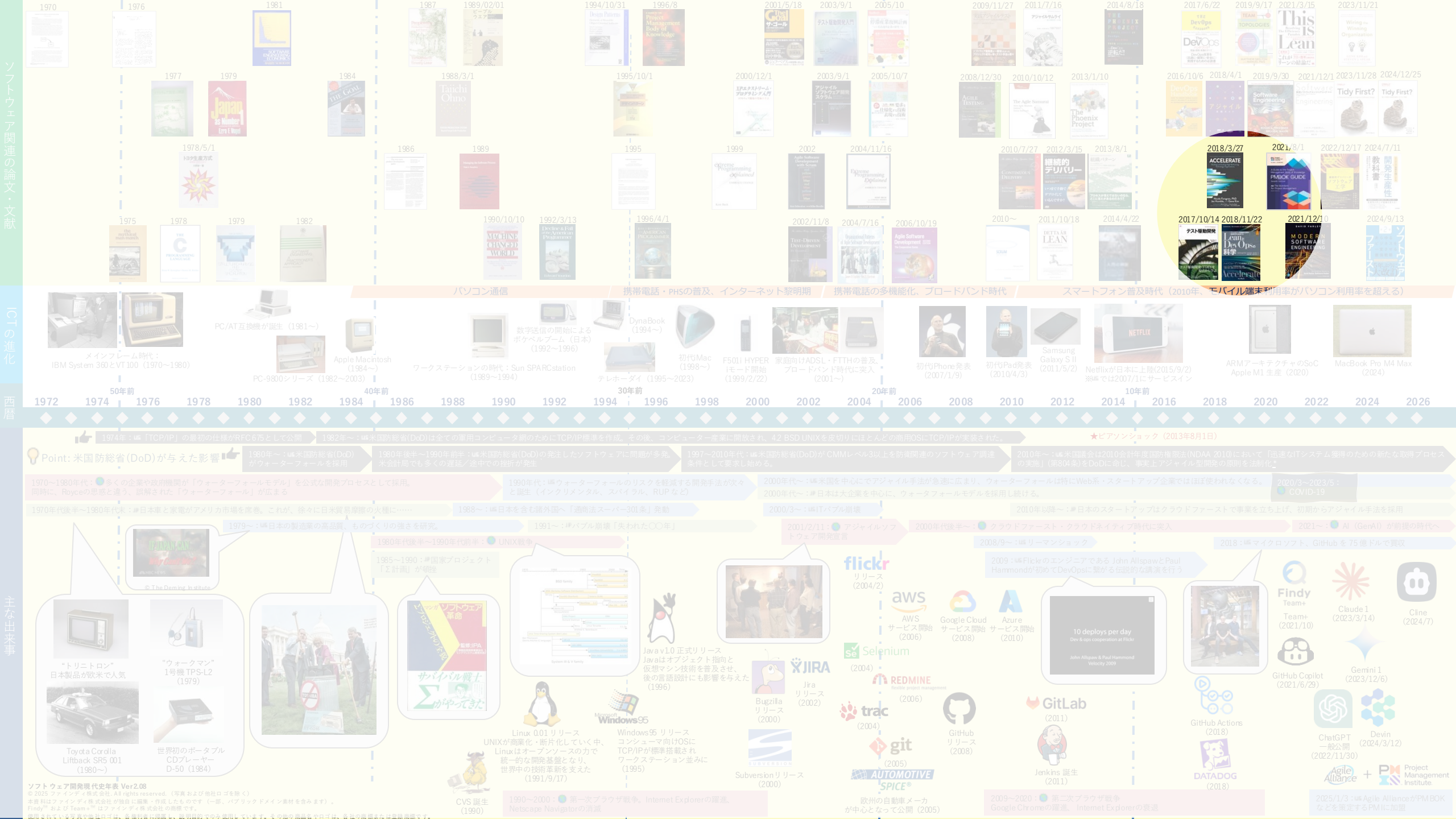
ChatGPT
一般公開
（2022/11/30）

Devin
（2024/3/12）

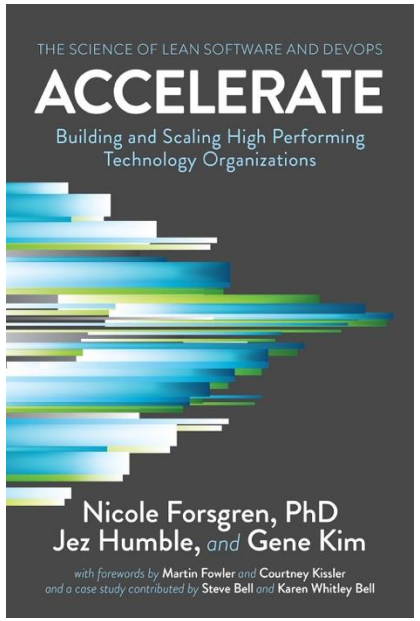
Agile Alliance + Project Management Institute

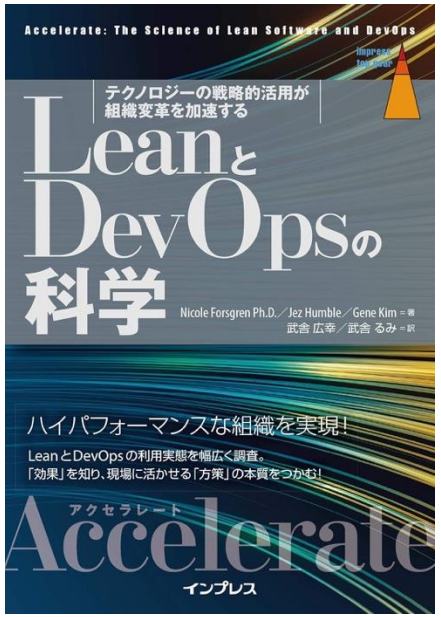
2025/1/3：🇺🇸Agile AllianceがPMBOKなどを策定するPMIに加盟

ソフトウェア開発現代史年表 Ver2.09



LeanとDevOpsの科学: テクノロジーの戦略的活用が組織変革を加速する

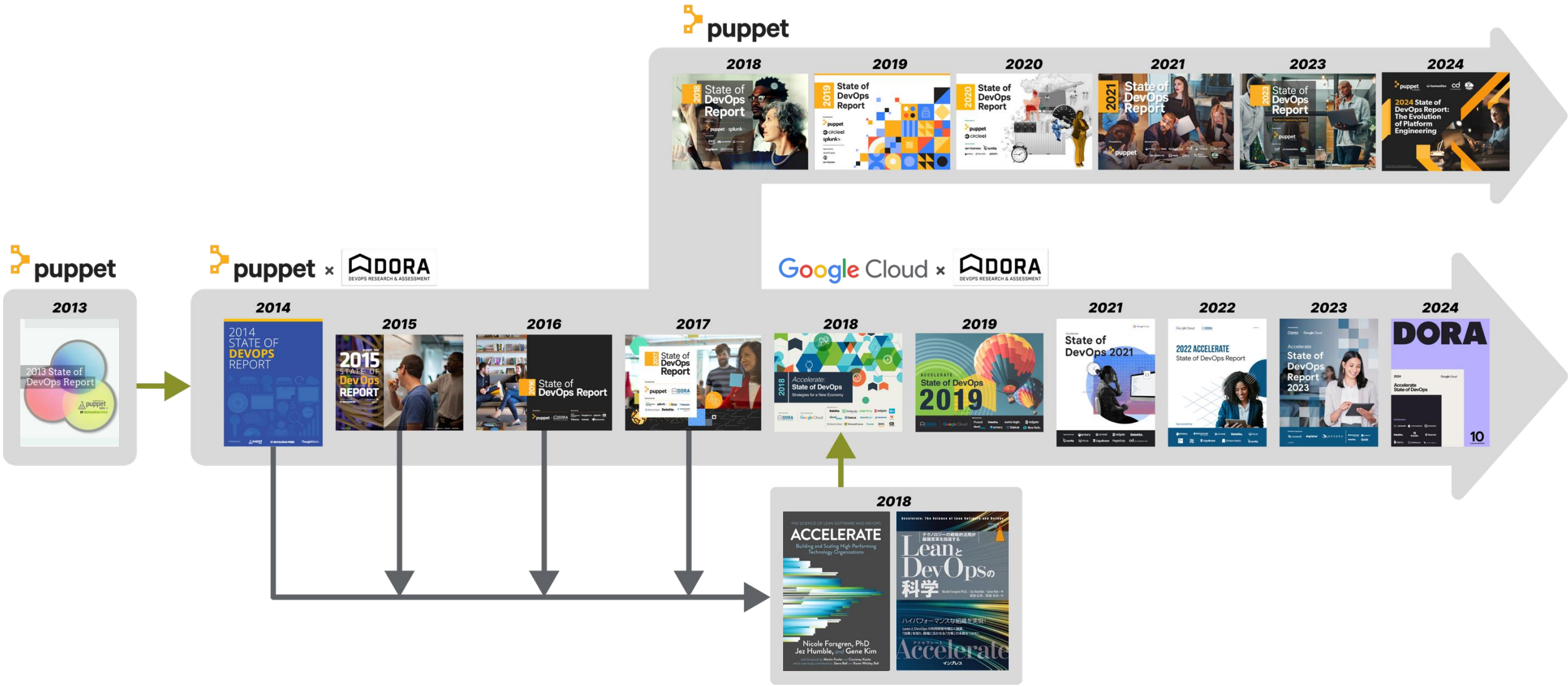




- 原書タイトルは「Accelerate: The Science Behind Devops: Building and Scaling High Performing Technology Organizations」
- 2018年3月出版、日本語版は2018年11月に出版。
- 迅速かつ高品質なデリバリを実施している組織とそうでない組織の違いに関する研究結果をDORAがまとめている。

- この書籍がきっかけで有名になったり、その重要性が広く認識されるようになったりした法則、概念
  1. Four Keys（4つの主要指標）
  2. 速度（スループット）と安定性はトレードオフではない
  3. 継続的デリバリー (Continuous Delivery / CD) の効果の証明
  4. 疎結合アーキテクチャの重要性
  5. リーン（Lean）の原則の適用
  6. Westrum（ウェストラム）の組織文化類型
  7. ケイパビリティモデル (Capability Model)
  8. バーンアウト（燃え尽き症候群）と組織文化・ツールの関係

Findy



Findy

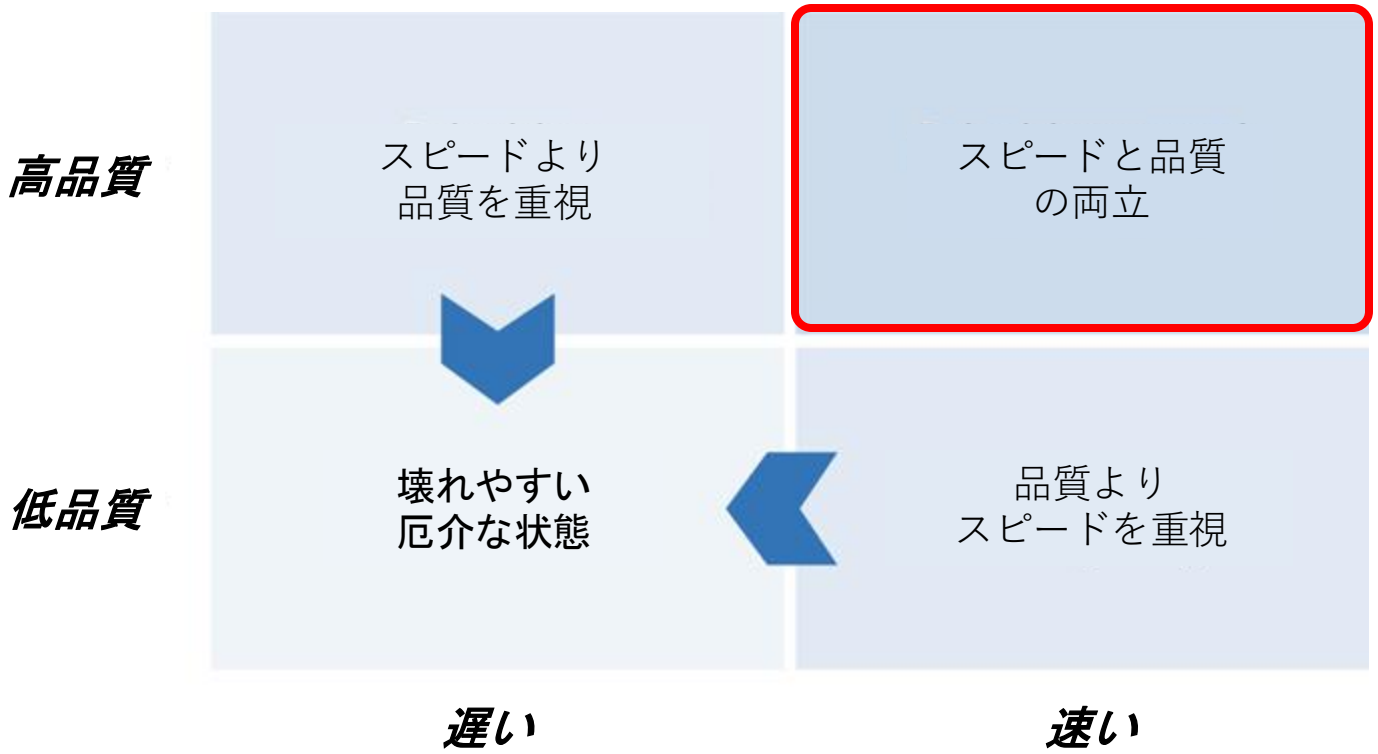
1. Four Keys（4つの主要指標）

- ソフトウェアデリバリーのパフォーマンスを客観的に測定するために定義された、以下の4つの指標群です。これらは DORA（DevOps Research and Assessment）の研究の中核であり、「DORA メトリクス」とも呼ばれます。
- DevOps の成果や組織のソフトウェアデリバリー能力を測定・比較するためのデファクトスタンダードとなりました。

指標	内容
Deployment frequency (デプロイ頻度)	変更を本番環境に push する頻度。
Lead time for changes (変更のリードタイム)	コードの変更を commit してからデプロイするまでの時間。
Change failure rate (変更失敗率)	デプロイにより障害が発生し、すぐに対処する必要が生じる頻度。
Failed deployment recovery time (失敗デプロイの復旧時間)	デプロイの失敗時に復旧にかかる時間。

2. 速度（スループット）と安定性はトレードオフではない

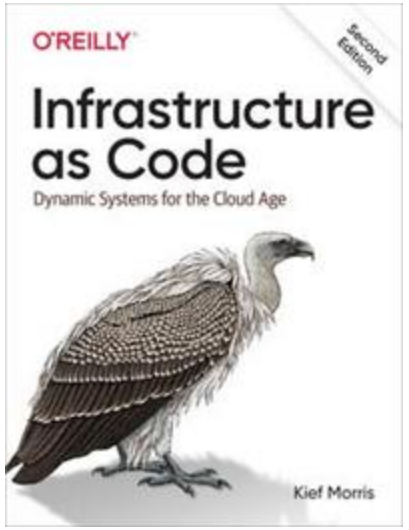
- Four Keys を用いた大規模な調査・分析から導き出された、本書の最も重要な発見の一つです。従来信じられていた「速度を上げると安定性が犠牲になる」という考えは誤りであり、高パフォーマンスな組織では、速度と安定性は両立し、むしろ互いに強化し合う関係にあることがデータで示されました。
- DevOps の価値を強力に裏付け、多くの組織が開発・運用プラクティスを見直す大きな動機となりました。

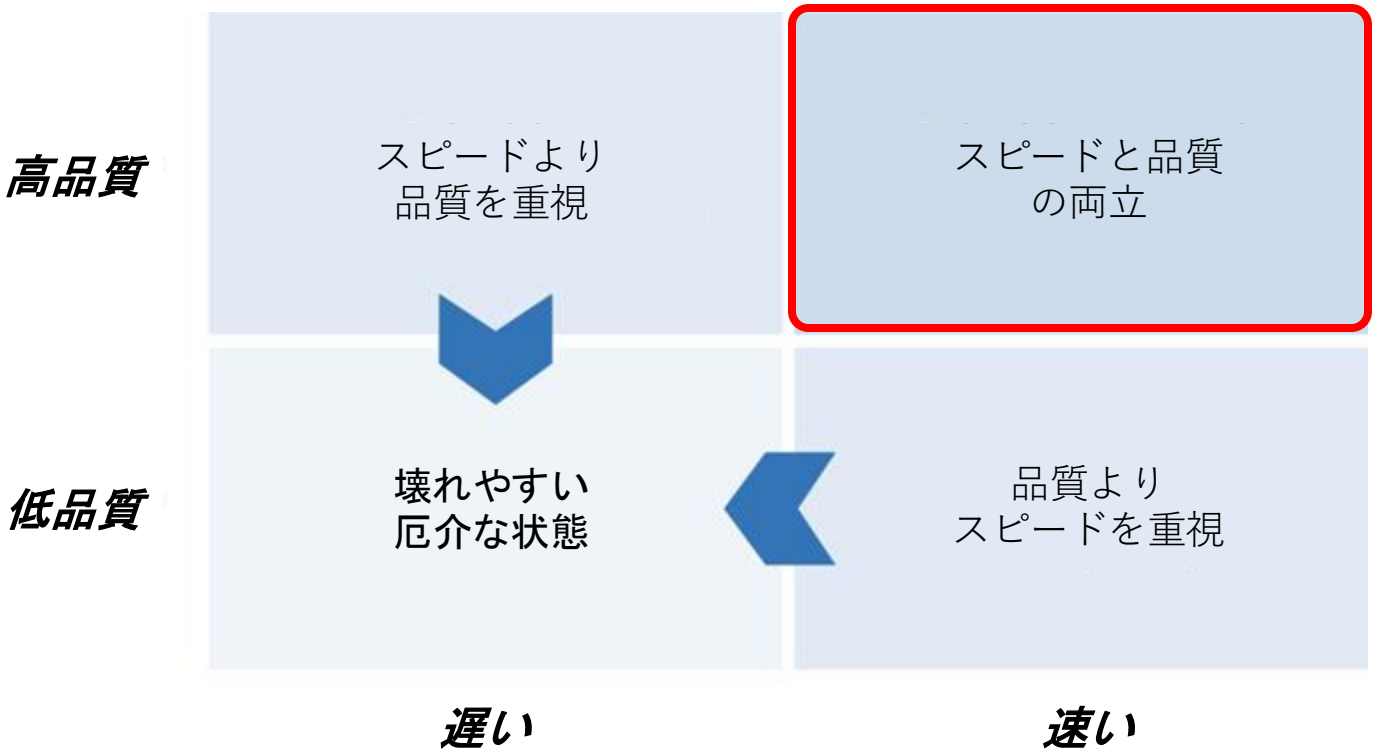


2. 速度（スループット）と安定性はトレードオフではない

左上の象限: スピードより品質を重視

- 「私たちは重要な仕事をしているのだから、きちんとやらなければならない」という考え方です。しかし、納期のプレッシャーにより「場当たり的な対応」を強いられます。重厚なプロセスが改善の障壁となり、技術的負債は「既知の問題」リストとともに増大します。これらのチームは左下の象限に落ち込みます。改善が困難なため、結果として低品質なシステムになってしまいます。失敗への対応として更にプロセスを追加します。このプロセスが改善をより困難にし、脆弱性とリスクを増大させます。これがさらなる失敗とプロセスの追加を招きます。特にリスクに敏感な業界では、このように働く組織の多くの人々が、これが普通だと思い込んでいます。





右下の象限: 品質よりスピードを重視

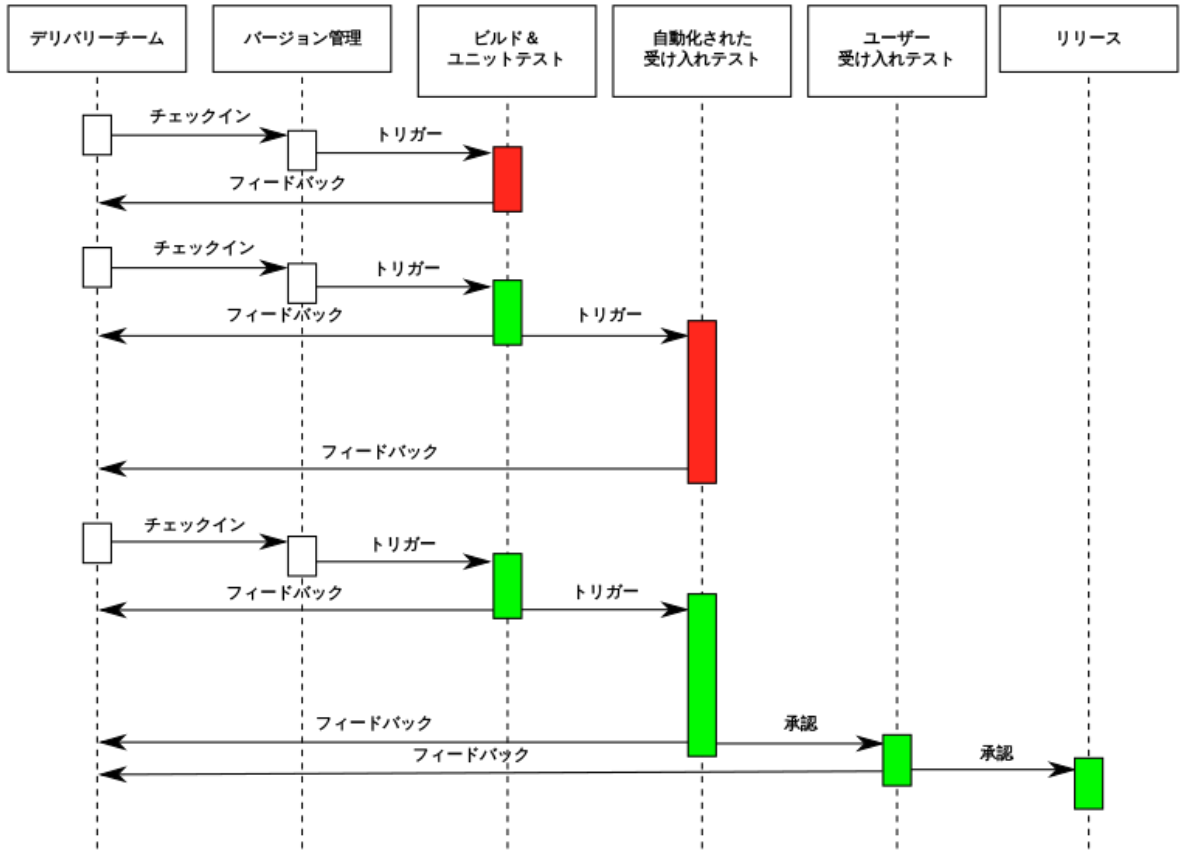
- いわゆる「早く作って壊す」という考え方です。スピードを重視して品質を犠牲にするチームは、乱雑で脆弱なシステムを構築します。その粗悪なシステムによって作業が遅くなり、左下の象限に滑り落ちていきます。このやり方を続けてきたスタートアップの多くが「勢い」を失ったと嘆きます。以前なら素早く対応できた簡単な変更が、システムが複雑に絡み合っているために、今では何日も何週間もかかるようになっています。

Findy

(出典) Infrastructure as Code, 2nd Edition , Figure 1-2. Speed and quality map to quadrants

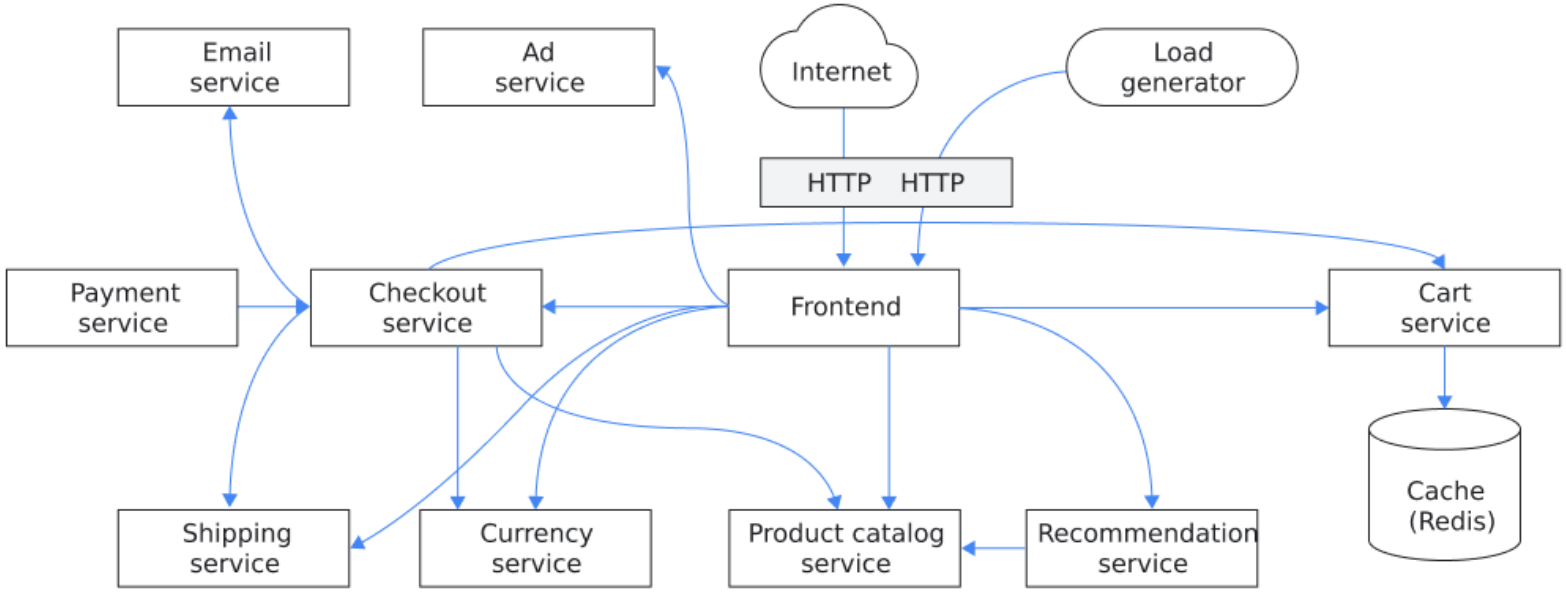
3. 継続的デリバリー (Continuous Delivery / CD) の効果の証明

- Four Keys で測定されるパフォーマンス（速度と安定性の両方）を高める上で、バージョン管理、自動テスト、自動デプロイといった CD の具体的なプラクティスが直接的に貢献することを科学的に証明しました。
- CD が単なる理想論ではなく、ビジネス成果に繋がる実践であることの説得力を高め、その普及を後押ししました。



4. 疎結合アーキテクチャの重要性

- チームが独立してテストやデプロイを行える「疎結合アーキテクチャ」（例：マイクロサービス）が、CD を容易にし、デリバリーパフォーマンス向上に大きく寄与することを示しました。
- 技術的なアーキテクチャ選択が、組織の俊敏性や DevOps 実践の効果に直接的な影響を与えるという考え方をデータで補強しました。



5. リーン（Lean）の原則の適用

- WIP 制限、小さなバッチでの作業、フィードバックループの短縮といったリーン原則が、ソフトウェアデリバリーのフローを改善し、リードタイム短縮や品質向上に有効であることを示しました。
- DevOps とリーンの考え方の親和性を裏付け、プロセス改善の具体的な指針を提供しました。

リーン原則のカテゴリー	核となる目的（何を目指すか）	最も重要なケイパビリティ（具体的な実践例）
1. リーン思考に基づく管理と監視	**作業の流れ（フロー）を最適化**し、過負荷を避け、制約を可視化する。	**進行中の作業（WIP）の制限** と **負担の軽い変更承認プロセス** (ピアレビューなど)。
2. 製品とプロセス	顧客価値の検証とフィードバックの加速、実験文化の促進。	**作業の細分化**（1週間未満での完了を目標）と **チームによる実験** の奨励・実現。
3. 継続的デリバリ（CD）	高頻度、高品質、低リスクなソフトウェアリリースを実現する技術的基盤。	**デプロイの自動化**（手作業の介入を不要にする）と **トランクベースの開発**（ブランチ寿命を短く保つ）。
4. アーキテクチャ	チームが他者に依存せず、独立して迅速にデプロイ・テストできるシステムの設計。	**疎結合のアーキテクチャ** の実現（テストとデプロイの容易性の確保）。
5. 組織文化	**信頼、協働、学習** を促進し、健全でハイパフォーマンスな環境を築く。	**Westrum推奨の創造的な組織文化の育成**（失敗を非難せず調査し、学びの機会とする）。

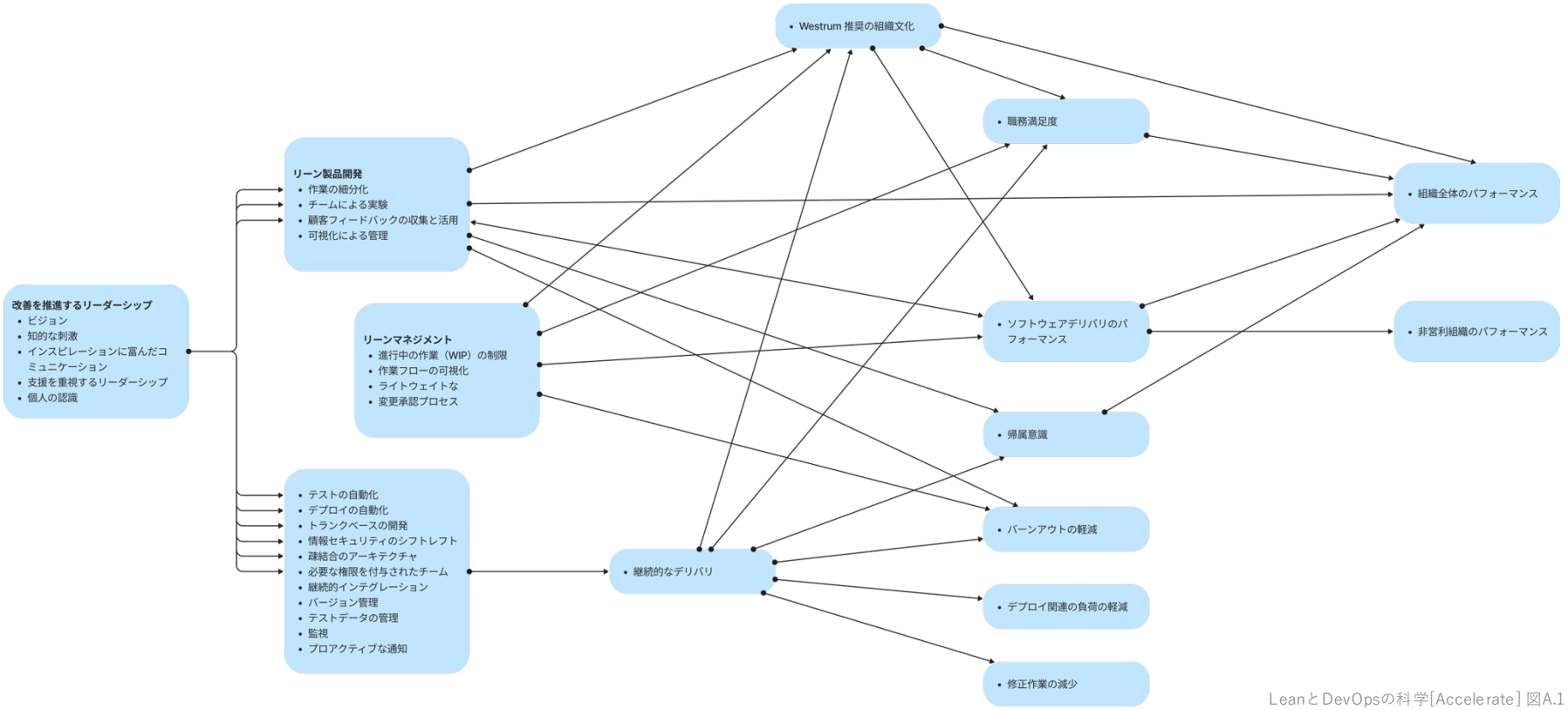
6. Westrum（ウェストラム）の組織文化類型

- 組織文化を情報の流れ方で分類し（病的、官僚的、生成的）、情報共有、協力、学習を促す「生成的（Generative）文化」が、Four Keys で測られるような高いパフォーマンスと強く関連していることを示しました。
- DevOps 成功における**健全な組織文化の構築**が不可欠であることをデータで示し、その重要性を広く認識させました。

特性	病理的（権力重視）	官僚的（ルール重視）	生成的（成果重視）
組織の価値観・指向性	権力と統制が重視される	規則と手続きが重視される	ミッションや成果が重視される
協力の姿勢	協力はほとんど見られない	必要最低限の協力が行われる	積極的に協力し合う文化がある
問題を報告した人への対応	報告した人が責められる（処罰される）	報告しても関心を持たれず無視される	報告が歓迎され、問題対応力が育てられる
責任の取り方	責任を押しつけあう（逃げる）	形式上の責任だけを取る	チームで責任を共有し、積極的に対応する
部門間の連携（橋渡し）	他部署との連携は妨げられる	最低限は許容される	積極的に部門を越えて連携する
失敗への対応	スケープゴートを探して責任追及する	誰が悪かったかを調査する	失敗から学び、次に活かす姿勢がある
新しいアイデアへの対応	新しいことは否定され、抑え込まれる	新しいことは規則に反するとして懸念される	新しいことに挑戦し、積極的に取り入れる

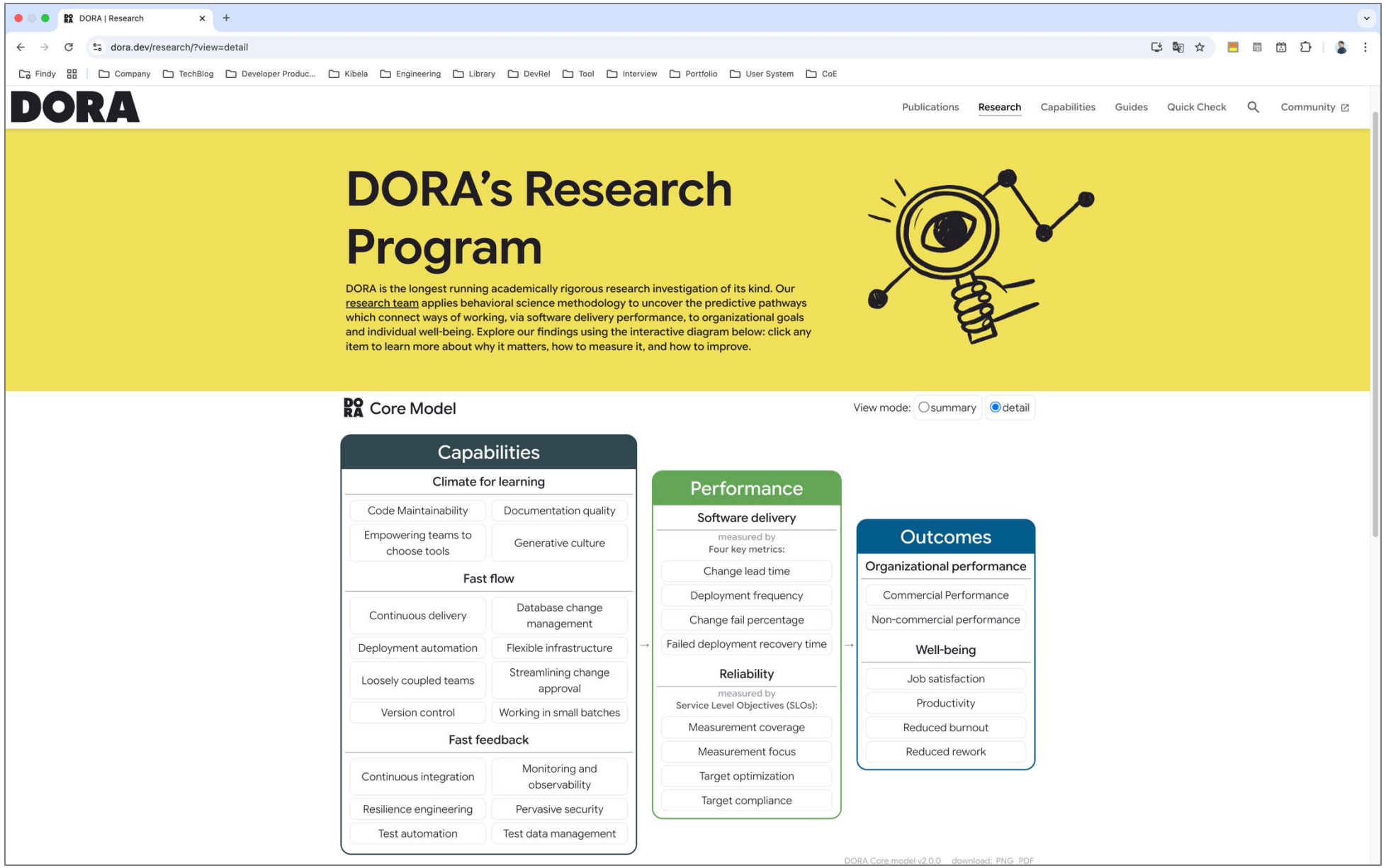
7. ケイパビリティモデル (Capability Model)

- CDプラクティス、アーキテクチャ、リーン原則、組織文化などを含むパフォーマンス向上に統計的に有意な影響を与えることが特定された具体的な技術的・プロセス的・文化的な「能力（Capability）」の集合体です。
- Four Keys で現状を測定した後、具体的にどの能力を開発・強化すればパフォーマンスが向上するのかを示す実践的なガイドを提供し、組織が改善活動を進める上でのロードマップとなりました。



LeanとDevOpsの科学[Accelerate] 図A.1 本研究の全体の構成 を元に筆者 が作成

7. ケイパビリティモデル (Capability Model)



7. ケイパビリティモデル (Capability Model)

DORA Community Guides

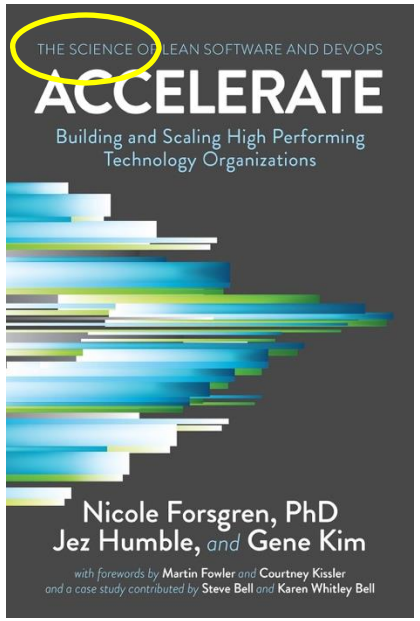
https://dora.community/join

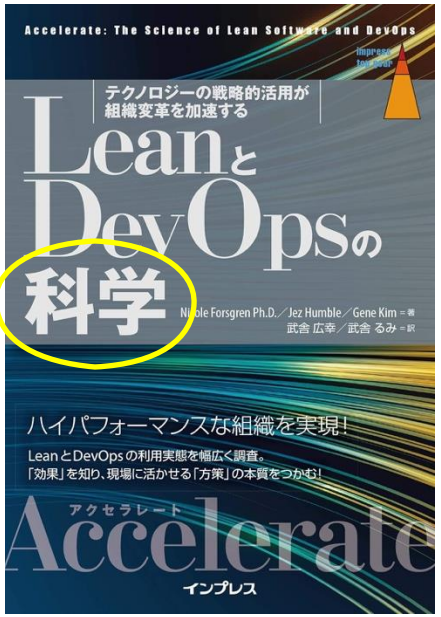
Steve Fenton	Denali Lumma	Betsalel (Saul) Williamson	Lisa Crispin
			
スティーブ・フェントンは、Octopus Deployで働く開発者であり、Microsoft MVPを5回受賞しているソフトウェアデリバリーの専門家です。彼はTypeScript、Octopus Deploy、Webオペレーションに関する書籍を執筆しており、ソフトウェアエンジニア、SDET（ソフトウェア開発エンジニア）、開発マネージャー、製品およびデータディレクターなど、多様な役割をスタートアップから大企業にわたって経験しています。	デナリ・ルンマは、20年以上の経験を持つテクノロジーエグゼクティブで、Netflix、23andMe、Okta、Salesforce、Uberなどで初期エンジニアとしてユニコーン企業の成長に貢献してきました。B2C、B2B、ヘルスケア、ライフサイエンスなどの分野での製品開発やインフラ、セキュリティ、品質に関するチームをサポート。現在はDoublingの創設者兼CEOとして、Y CombinatorやAndreessen Horowitzなどのポートフォリオ企業に技術アドバイザリーを提供しています。	ベツァレル（サウル）・ウィリアムソンは、電気技師であり、コミュニティ開発やDevOpsの広範な経験を持っています。彼はWilliamson Computing Services, LLCのオーナーで、企業のDevOpsおよび新製品開発におけるベストプラクティスの実装を支援しています。サウルは障がい者およびLGBT+の擁護者であり、多様性、公平性、包括性を推進しています。仕事以外では、趣味のソフトウェアプロジェクトやクラシックピアノ、旅行を楽しみ、カリフォルニア州パームスプリングスに住んでいます。	リサ・クリスピンは、ジャネット・グレゴリーと共に『Agile Testing Condensed』、『More Agile Testing』、『Agile Testing』の3冊を共著し、アジャイルテストのビデオコースも提供しています。彼女は2012年にAgile Testing Daysで最も影響力のあるアジャイルテストの専門家に選ばれ、Janetとともに「Agile Testing Fellowship」を共同設立し、アジャイルチーム向けの「ホリスティックテスト」のトレーニングを提供しています。詳しくは、lisacrispin.com をご覧ください。
https://www.linkedin.com/in/stevefenton/	https://www.linkedin.com/in/denali-lumma/	https://www.linkedin.com/in/betsalel/	https://www.linkedin.com/in/lisa-crispin-88420a/ https://x.com/lisacrispin

8. バーンアウト（燃え尽き症候群）と組織文化・ツールの関係

- デプロイの苦痛、責任の押し付け合いといったネガティブな文化や、不十分なツールがバーンアウトのリスクを高める一方、生成的文化や適切なツールはそれを低減しうることを示唆しました。
- 従業員のウェルビーイング（幸福感）が、組織の持続的なパフォーマンスにとっても重要であるという視点を提供しました。

LeanとDevOpsの科学: テクノロジーの戦略的活用が組織変革を加速する





- 原書タイトルは「Accelerate: The Science Behind Devops: Building and Scaling High Performing Technology Organizations」
- 2018年3月出版、日本語版は2018年11月に出版。
- 迅速かつ高品質なデリバリを実施している組織とそうでない組織の違いに関する研究結果をDORAがまとめている。

- この書籍がきっかけで有名になったり、その重要性が広く認識されるようになったりした法則、概念
  1. Four Keys（4つの主要指標）
  2. 速度（スループット）と安定性はトレードオフではない
  3. 継続的デリバリー (Continuous Delivery / CD) の効果の証明
  4. 疎結合アーキテクチャの重要性
  5. リーン（Lean）の原則の適用
  6. Westrum（ウェストラム）の組織文化類型
  7. ケイパビリティモデル (Capability Model)
  8. バーンアウト（燃え尽き症候群）と組織文化・ツールの関係

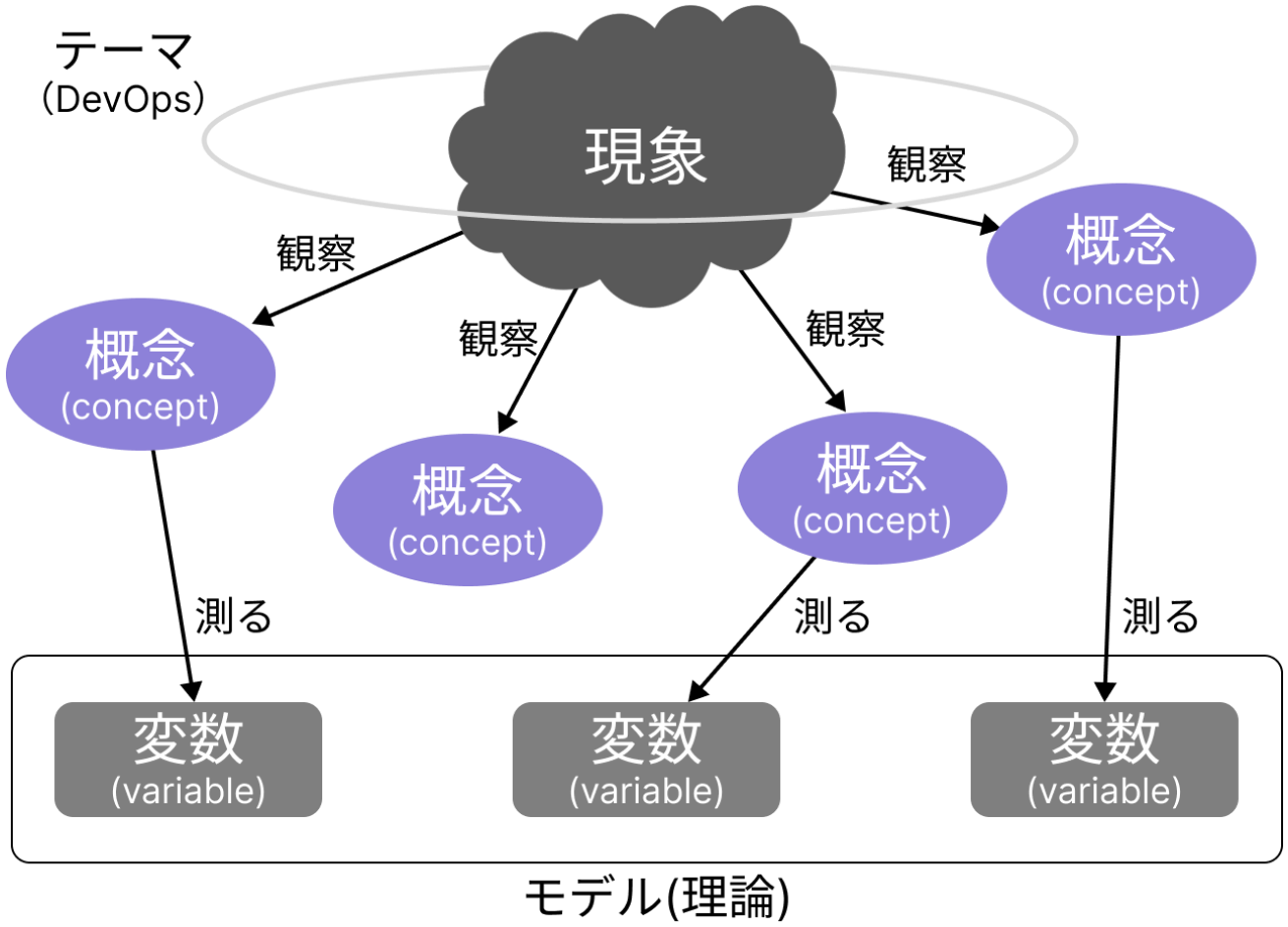
Findy

科学とは何か？

科学とは「なぜだろう？」という疑問に対して、実験と観察を重ねながら、誰でも同じ結果を得られる答えを見つけ出す取り組み、と言えます。

「科学的リサーチ」方法とは

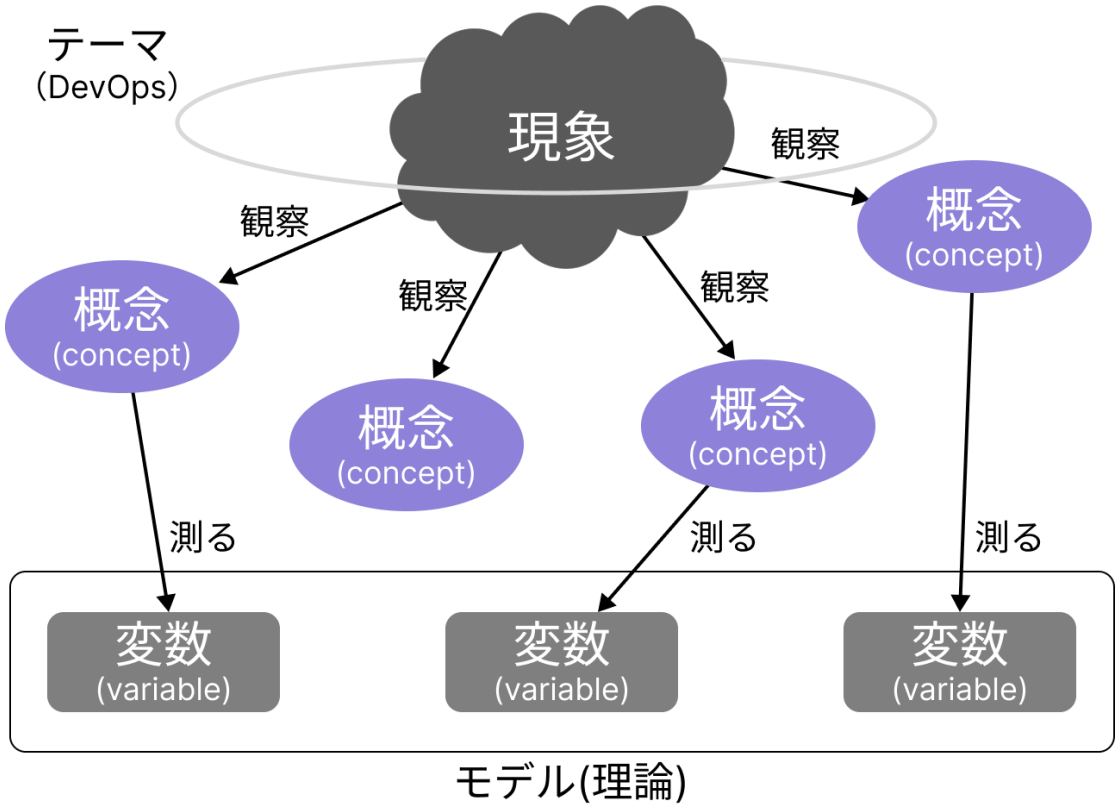
- 科学的リサーチとは、現象を体系的に調査し、新しい発見や理論を導き出す手法です。観察から始まり、概念化、**測定可能な変数**への置き換え、モデル化という段階を経て、客観的なデータに基づいた結論を導きます。



向後, 千春. (2016). 18歳からの「大人の学び」基礎講座: 学ぶ, 書く, リサーチする, 生きる. 北大路書房. 図3-3に筆者が加筆

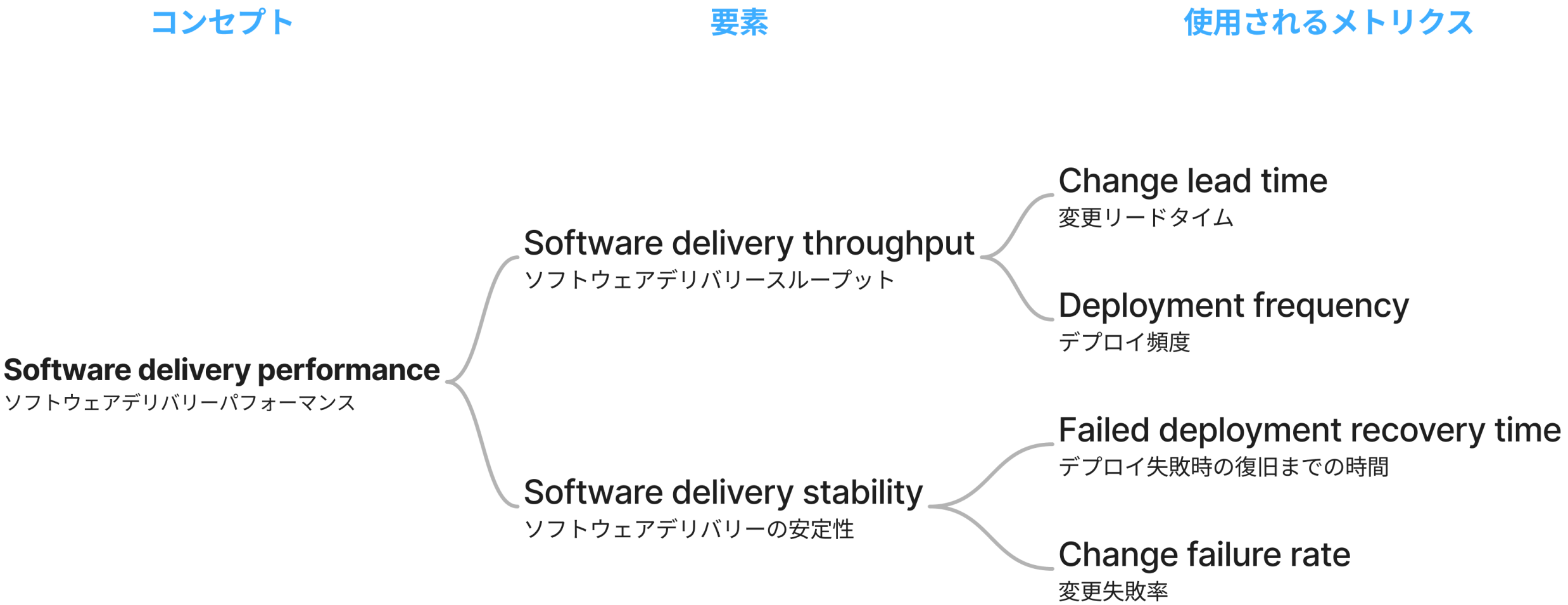
科学的リサーチプロセスは次の4段階で進みます

Step	説明
1 現象の観察	「DevOpsの実践は本当にビジネスの成果に繋がるのか？」といった問いからDORAの研究が始まりました。
2 概念の特定	DORAは、継続的デリバリー、継続的インテグレーション、自動化テストの実施などを概念として特定しました。
3 変数への変換	概念を測定可能な形に変換します。例えば、デプロイ頻度、変更リードタイム、障害復旧時間、変更失敗率などの具体的指標として定義します。
4 モデルの構築	収集したデータを統計的に分析し、相関または因果関係を明らかにします。DORAの研究では、自動化テストの実施が変更リードタイムを短縮することや、チームの独立性がサービスの信頼性向上につながることを実証しました。



向後, 千春. (2016). 18歳からの「大人の学び」基礎講座: 学ぶ, 書く, リサーチする, 生きる. 北大路書房. 図3-3に筆者が加筆

DORA Metrics (Four Keys)



DORA Metrics (Four Keys)

使用されるメトリクス

Change lead time
変更リードタイム

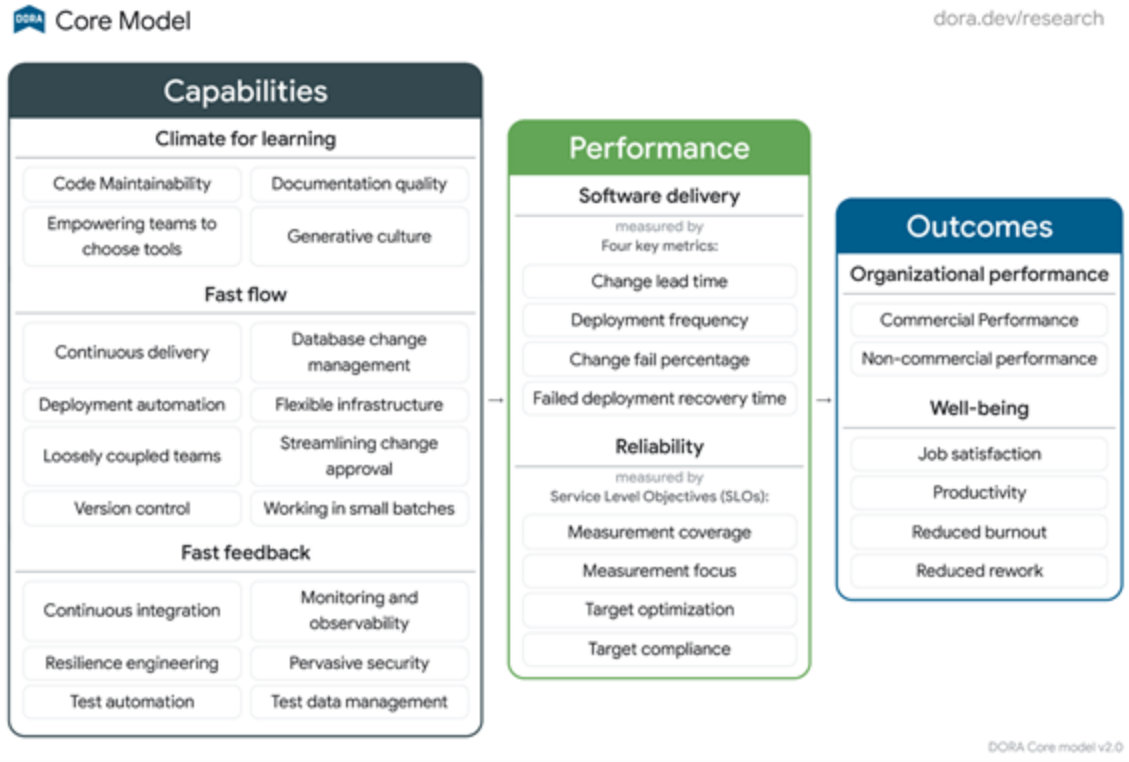
Deployment frequency
デプロイ頻度

Failed deployment recovery time
デプロイ失敗時の復旧までの時間

Change failure rate
変更失敗率

統計的相関

DORA Model



Findy

技術的発展の階層構造

DevOpsムーブメントが来ないままGenAIムーブメントが来てしまった日本

この2年で、状況は劇的に
変わっています
（日本ではこれから？）

ソフトウェア関連の論文・文献

ICTの進化

西暦

主な出来事

1970 1976 1981 1987 1989/02/01 1994/10/31 1996/8 2001/5/18 2003/9/1 2005/10 2009/11/27 2011/7/16 2014/8/18 2017/6/22 2019/9/17 2021/3/15 2023/11/21 2025/6/25

1977 1979 1984 1988/3/1 1995/10/1 2000/12/1 2003/9/1 2005/10/7 2008/12/30 2010/10/12 2013/1/10 2016/10/6 2018/4/1 2019/9/30 2021/12/1 2023/11/28 2024/12/25

1978/5/1 1986 1989 1995 1999 2002 2004/11/16 2010/7/27 2012/3/15 2013/8/1 2018/3/27 2021/8/1 2022/12/17 2024/7/11

1975 1978 1979 1982 1990/10/10 1992/3/13 1996/4/1 2002/11/8 2004/7/16 2006/10/19 2010～ 2011/10/18 2014/4/22 2017/10/14 2018/11/22 2021/12/10 2024/9/13

パソコン通信 | 携帯電話・PHSの普及、インターネット黎明期 | 携帯電話の多機能化、ブロードバンド時代 | スマートフォン普及時代（2010年、モバイル端末利用率がパソコン利用率を超える）

メインフレーム時代：
IBM System 360とVT100（1970～1980）

PC/AT互換機が誕生（1981～）

PC-9800シリーズ（1982～2003）

Apple Macintosh
（1984～）

ワークステーションの時代：
Sun SPARCstation
（1989～1994）

数字送信の開始によるポケベルブーム（日本）
（1992～1996）

テレホーダイ
（1995～2023）

初代iMac
（1998～）

F501i HYPER iモード開始
（1999/2/22）

家庭向けADSL・FTTHの普及、ブロードバンド時代に突入
（2001～）

初代iPhone発表
（2007/1/9）

初代iPad発表
（2010/4/3）

Samsung Galaxy S II
（2011/5/2）

Netflixが日本に上陸(2015/9/2)
※🇺🇸では2007/1にサービスイン

ARMアーキテクチャのSoC
Apple M1 生産（2020）

MacBook Pro M4 Max
（2024）

50年前 40年前 30年前 20年前 10年前

1972 1974 1976 1978 1980 1982 1984 1986 1988 1990 1992 1994 1996 1998 2000 2002 2004 2006 2008 2010 2012 2014 2016 2018 2020 2022 2024 2026

1974年：🇺🇸「TCP/IP」の最初の仕様がRFC 675として公開

1982年～：🇺🇸米国防総省(DoD)は全ての軍用コンピュータ網のためにTCP/IP標準を作成。その後、コンピューター産業に開放され、4.2 BSD UNIXを皮切りにほとんどの商用OSにTCP/IPが実装された。

★ピアソンショック（2013年8月1日）

Point: 米国防総省(DoD)が与えた影響

1980年～：🇺🇸米国防総省(DoD)がウォーターフォールを採用

1980年後半～1990年前半：🇺🇸米国防総省(DoD)の発注したソフトウェアに問題が多発。米会計局でも多くの遅延／途中での挫折が発生

1997～2010年代：🇺🇸米国防総省(DoD)がCMMレベル3以上を防衛関連のソフトウェア調達条件として要求し始める。

2010年～：🇺🇸米国議会は2010会計年度国防権限法(NDAA 2010)において「迅速なITシステム獲得のための新たな取得プロセスの実施」(第804条)をDoDに命じ、事実上アジャイル型開発の原則を法制化

1970～1980年代：🌎多くの企業や政府機関が「ウォーターフォールモデル」を公式な開発プロセスとして採用。
同時に、Royceの思惑と違う、誤解された「ウォーターフォール」が広まる

1990年代：🇺🇸ウォーターフォールのリスクを軽減する開発手法が次々と誕生（インクリメンタル、スパイラル、RUP など）

2000年代～：🇺🇸米国を中心にでアジャイル手法が急速に広まり、ウォーターフォールは特にWeb系・スタートアップ企業ではほぼ使われなくなる。

2000年代～：🇯🇵日本は大企業を中心に、ウォータフォールモデルを採用し続ける。

2020/3～2023/5：🌎COVID-19

1970年代後半～1980年代末：🇯🇵日本車と家電がアメリカ市場を席巻。これが、徐々に日米貿易摩擦の火種に……

1988～：🇺🇸日本を含む諸外国へ「通商法スーパー301条」発動

2000/3～：🇺🇸ITバブル崩壊

2010年以降～：🇯🇵日本のスタートアップはクラウドファーストで事業を立ち上げ、初期からアジャイル手法を採用

1979～：🇺🇸日本の製造業の高品質、ものづくりの強さを研究。

1991～：🇯🇵バブル崩壊「失われた○○年」

2001/2/11：🌎アジャイルソフトウェア開発宣言

2000年代後半～：🌎クラウドファースト・クラウドネイティブ時代に突入

2021～：🌎AI（GenAI）が前提の時代へ

2008/9～：🇺🇸リーマンショック

2018：🇺🇸マイクロソフト、GitHub を 75 億ドルで買収

1980年代後半～1990年代前半：🌎UNIX戦争

1985～1990：🇯🇵国家プロジェクト「Σ計画」が頓挫

2009：🇺🇸Flickrのエンジニアである John Allspaw と Paul Hammond が初めてDevOpsに繋がる伝説的な講演を行う

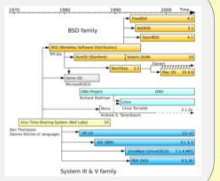
© The Deming Institute


“トリニトロン”
日本製品が欧米で人気


"ウォークマン"
1号機 TPS-L2
（1979）


Toyota Corolla
Liftback SR5 001
（1980～）


世界初のポータブル
CDプレーヤー
D-50（1984）



Java v1.0 正式リリース
Javaはオブジェクト指向と仮想マシン技術を普及させ、後の言語設計にも影響を与えた
（1996）


Bugzilla
リリース
（2000）

JIRA
Jira
リリース
（2002）


Linux 0.01 リリース
UNIXが商業化・断片化していく中、Linuxはオープンソースの力で統一的な開発基盤となり、世界中の技術革新を支えた
（1991/9/17）

Windows95 リリース
コンシューマ向けOSにTCP/IPが標準搭載されワークステーション並みに
（1995）


Subversionリリース
（2000）


CVS 誕生
（1990）

1990～2000：🌎第一次ブラウザ戦争。Internet Explorerの躍進、Netscape Navigatorの消滅

flickr
リリース
（2004/2）

aws
AWS
サービス開始
（2006）

Google Cloud
サービス開始
（2008）

Azure
サービス開始
（2010）

Selenium
（2004）

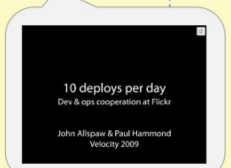
REDMINE
flexible project management
（2006）

trac
（2004）

git
（2005）

GitHub
リリース
（2008）

AUTOMOTIVE SPICE®
欧州の自動車メーカが中心となって公開（2005）



GitLab
（2011）


Jenkins 誕生
（2011）

2009～2020：🌎第二次ブラウザ戦争
Google Chromeの躍進、 Internet Explorerの衰退

GitHub Actions
（2018）

DATADOG
（2018）

Findy Team+
Team+
（2021/10）

Claude 1
（2023/3/14）

Cline
（2024/7）

GitHub Copilot
（2021/6/29）

Gemini 1
（2023/12/6）

ChatGPT
一般公開
（2022/11/30）

Devin
（2024/3/12）

Agile Alliance + Project Management Institute

2025/1/3：🇺🇸Agile AllianceがPMBOKなどを策定するPMIに加盟

ソフトウェア開発現代史年表 Ver2.09

2008 2010 2012 2014 2016 2018 2020 2022 2024 2026

Pが実装された。

★ピアソンショック（2013年8月1日）

トウェア調達

2010年～：米国議会は2010会計年度国防権限法(NDAA 2010)において「迅速なITシステム獲得のための新たな取得プロセスの実施」(第804条)をDoDに命じ、事実上アジャイル型開発の原則を法制化 *

速に広まり、ウォーターフォールは特にWeb系・スタートアップ企業ではほぼ使われなくなる。

ォールモデルを採用し続ける。

2020/3～2023/5：COVID-19

2010年以降～：日本のスタートアップはクラウドファーストで事業を立ち上げ、初期からアジャイル手法を採用

代後半～：クラウドファースト・クラウドネイティブ時代に突入

2021～：AI（GenAI）が前提の時代へ

2008/9～：リーマンショック

2018：マイクロソフト、GitHub を 75 億ドルで買収

2009：FlickrのエンジニアであるJohn AllspawとPaul Hammondが初めてDevOpsに繋がる伝説的な講演を行う

oogle Cloud
サービス開始
(2008)

Azure
サービス開始
(2010)

Findy Team+
Team+
(2021/10)

Claude 1
(2023/3/14)

Cline
(2024/7)

GitHub Copilot
(2021/6/29)

Gemini 1
(2023/12/6)

GitLab
(2011)

GitHub Actions
(2018)

GitHub
リリース
(2008)

ChatGPT
一般公開
(2022/11/30)

Devin
(2024/3/12)

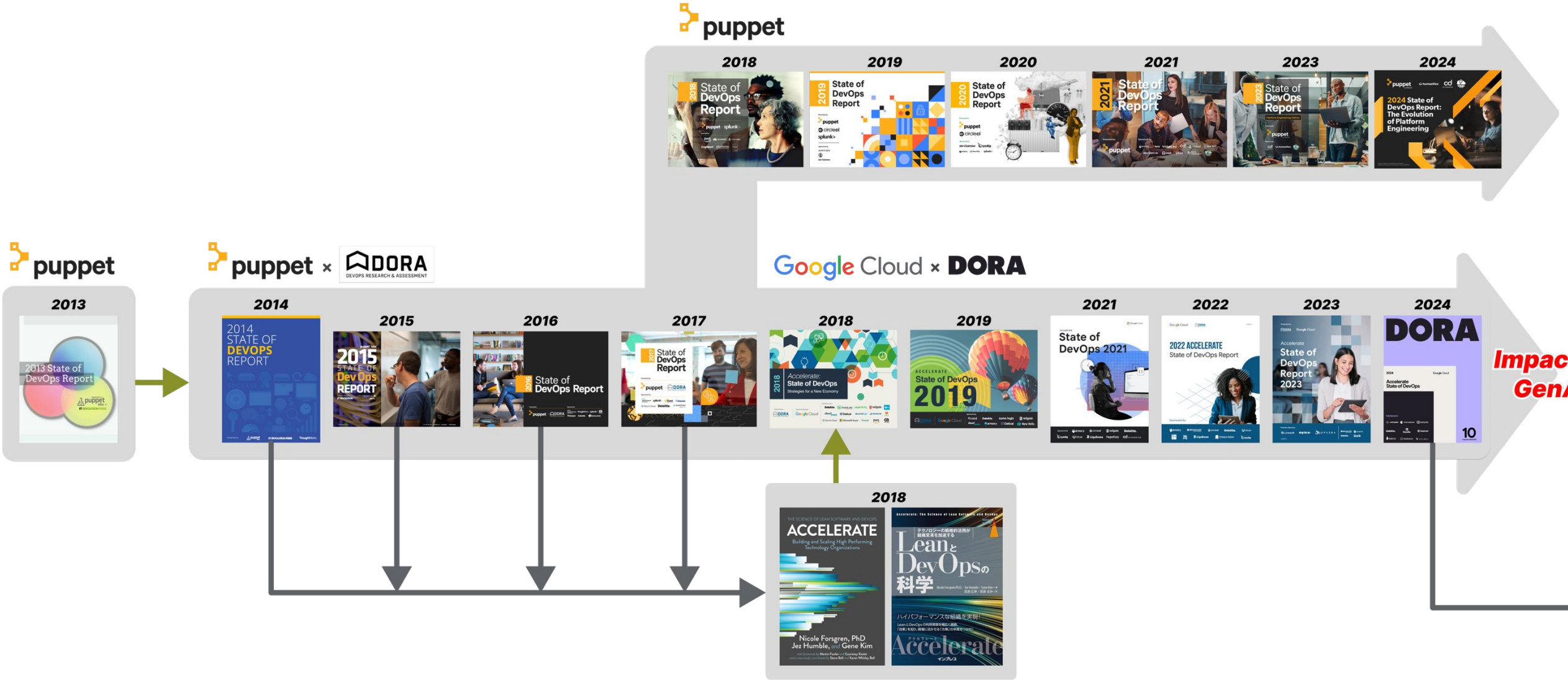
Jenkins 誕生
(2011)

DATADOG
(2018)

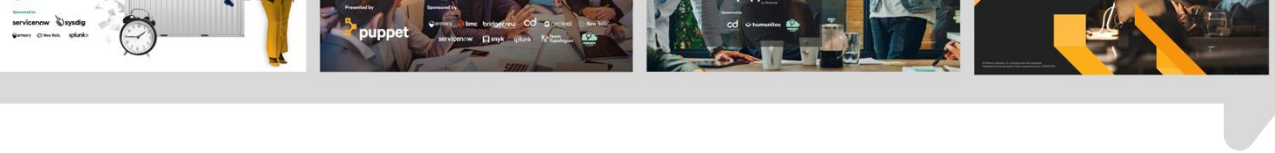
Agile Alliance + Project Management Institute

2009～2020：第二次ブラウザ戦争
Google Chromeの躍進、 Internet Explorerの衰退

2025/1/3：Agile AllianceがPMBOKなどを策定するPMIに加盟



Findy



DORA

2019

2021

2022

2023

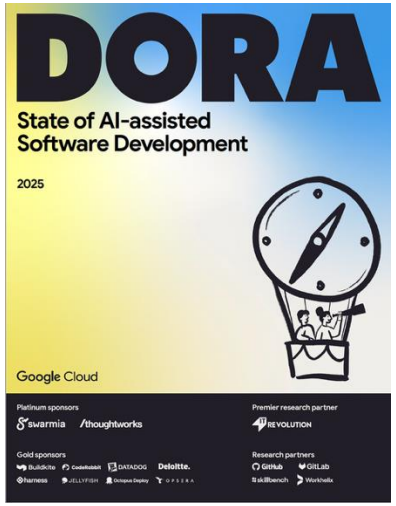
2024

Impact of GenAI

2025

2025

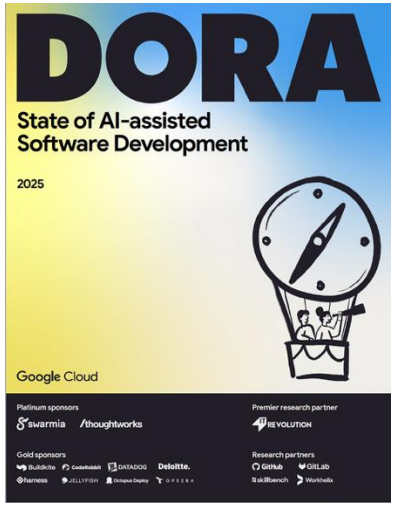
2025 DORA State of AI-assisted Software Development Report

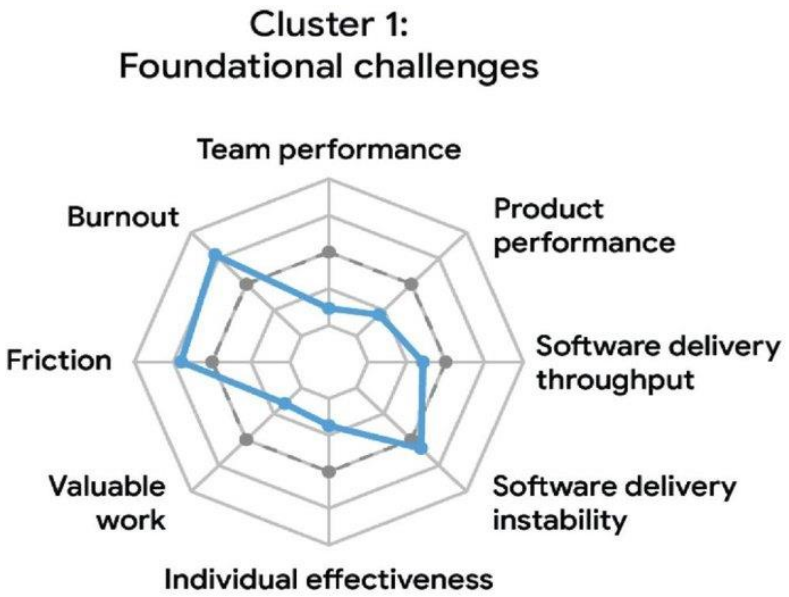


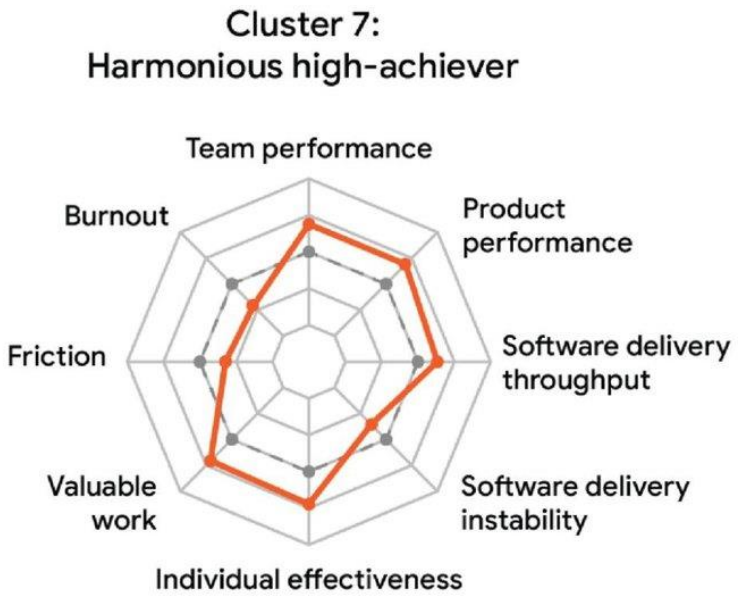
- **AI導入はほぼ普遍的になった**
  - 調査回答者の**大多数（95%）が現在AIに依存**しており、**80%以上が生産性が向上したと信じている。**しかし、注目すべきことに、**30%は現在AIが生成したコードに**ほとんどまたは全く**信頼を持っておらず**、重要な検証スキルの必要性を示している。
- **AI導入は現在、ソフトウェアデリバリースループットを改善している**
  - これは昨年からの重要な変化であう。しかし、依然としてデリバリーの不安定性を増加。これは、チームがスピードに適応している一方で、その基盤となるシステムはまだAI加速開発を安全に管理するように進化していないことを示唆している。
- **バリューストリームマネジメント（VSM）**
  - アイデアから顧客までの作業の流れを可視化、分析、改善する実践は、AIの力の乗数として機能し、局所的な生産性向上がチームと製品パフォーマンスの測定可能な改善に確実に変換されるようにる。
- **組織の90%がプラットフォームエンジニアリングを採用**
  - 高品質な内部プラットフォームがAI成功のための不可欠な基盤となってる。

2025 DORA State of AI-assisted Software Development Report

- **クラスター分析によって、7つのチームのアーキタイプが判明**
  - **クラスター1「基本的な課題」グループは、**サバイバル モードに陥っており、プロセスと環境に大きなギャップがあるため、パフォーマンスが低く、システムの安定性が高く、燃え尽き症候群や摩擦のレベルが高い状態。
  - **クラスター7「調和のとれた高パフォーマンス」は、**チームのウェルビーイング、プロダクトの成果、ソフトウェア デリバリーなど、複数の分野で優れた成果を上げている。





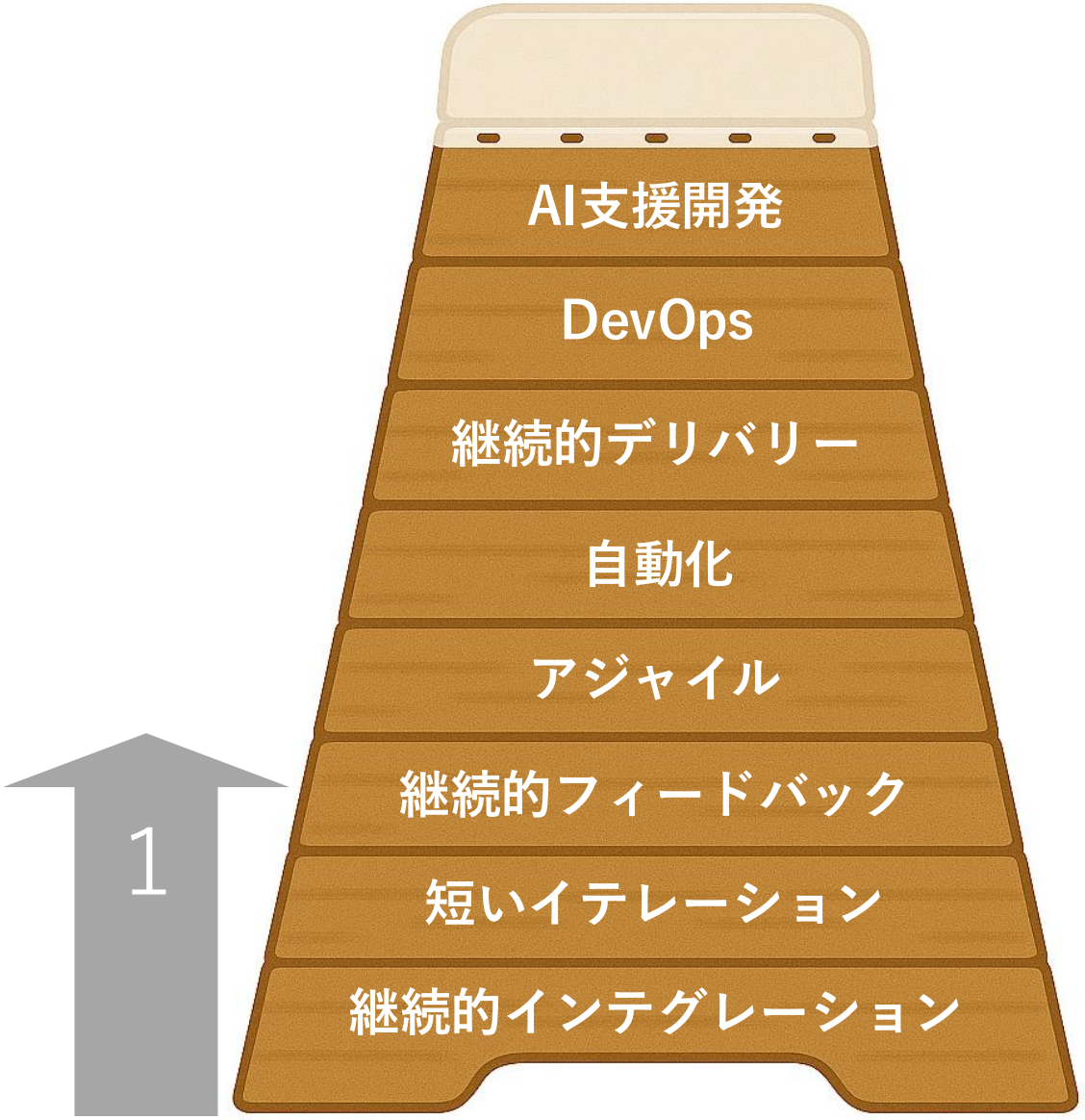


Findy

技術的発展の階層構造

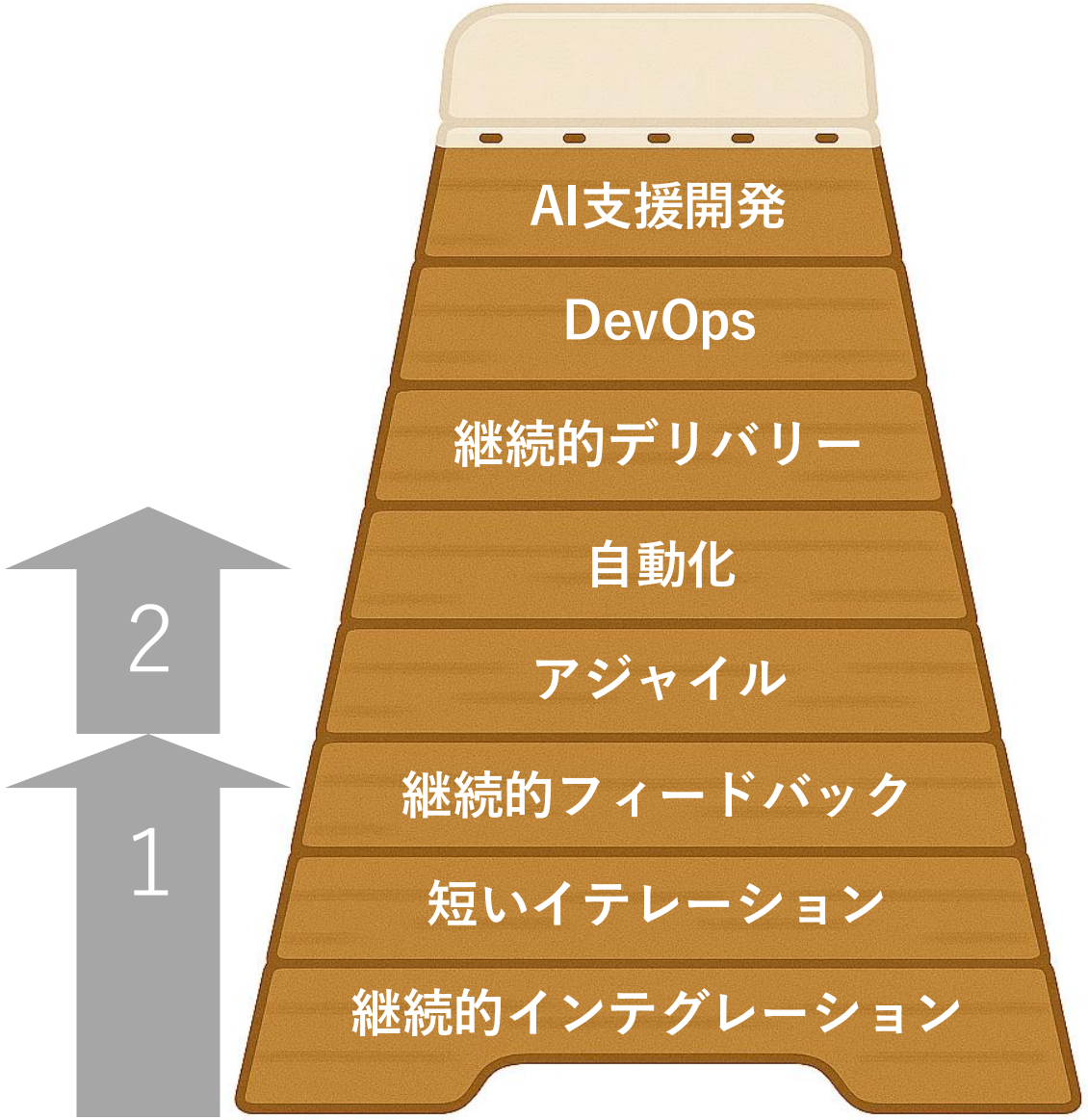
技術的発展の階層構造

- **foundation 1: WFのリスク軽減手法**（1990年代〜）
  - インクリメンタル、スパイラル、RUP
  - 継続的インテグレーション(CI)
  - 短いイテレーション、継続的フィードバック



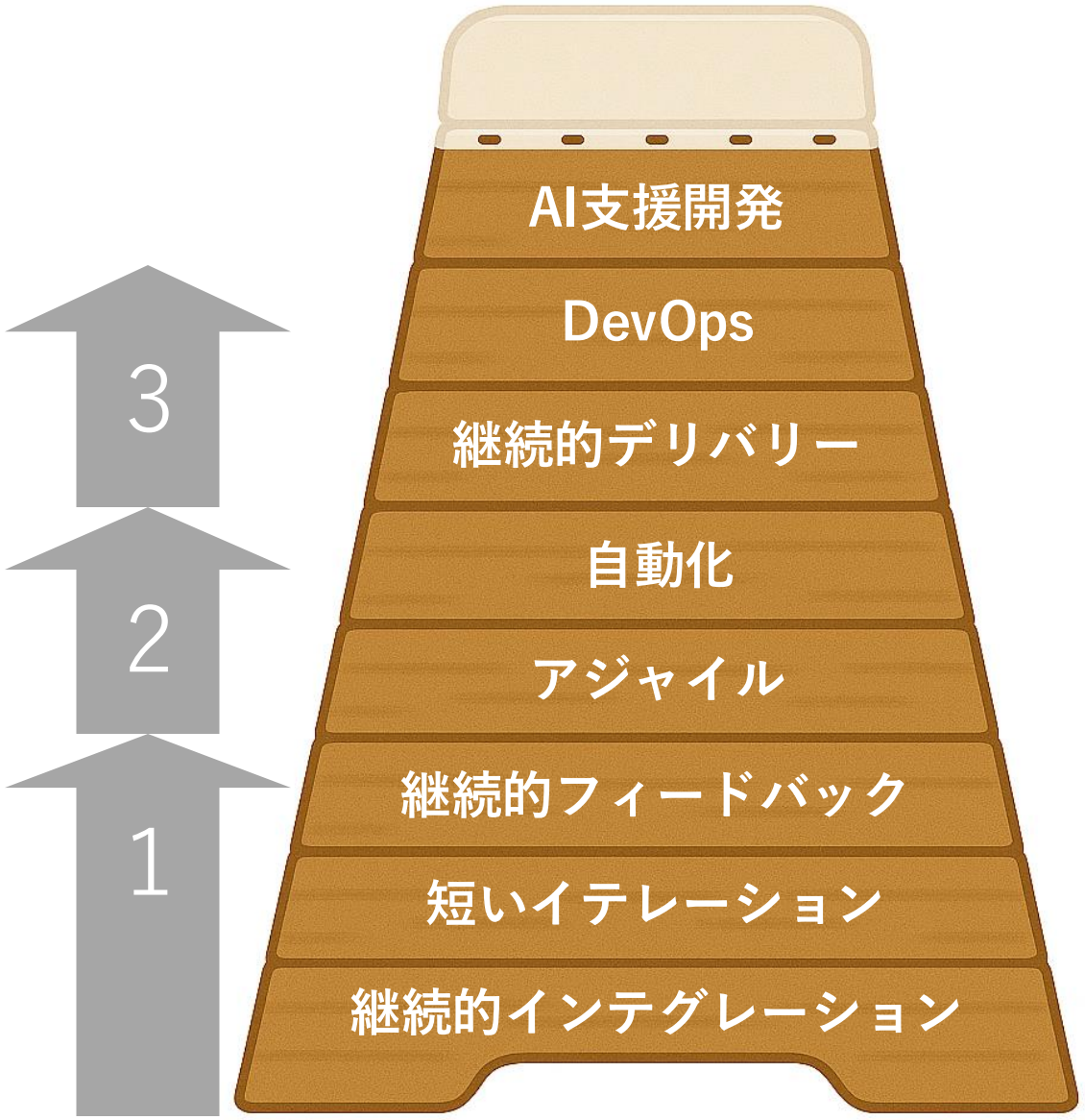
技術的発展の階層構造

- **foundation 2: アジャイル**（2000年代～）
  - クリスタル、Extreme Programming、Scrum
  - アジャイルソフトウェア開発宣言（Agile Manifesto）
- **foundation 1: WFのリスク軽減手法**（1990年代～）
  - インクリメンタル、スパイラル、RUP
  - 継続的インテグレーション(CI)
  - 短いイテレーション、継続的フィードバック



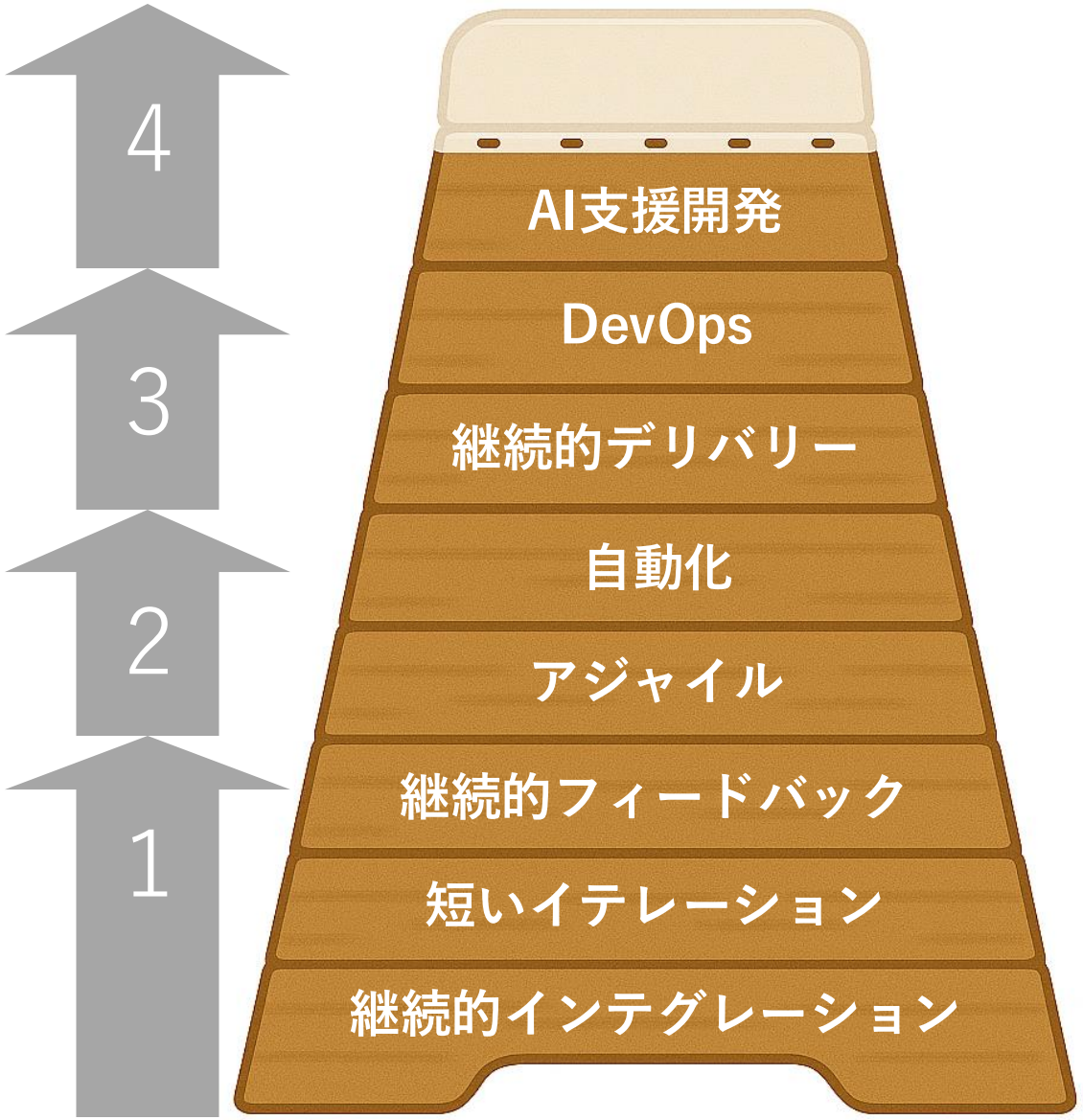
技術的発展の階層構造

- **foundation 3: DevOps**（2010年代～）
  - **アジャイルを土台に**、開発(Dev)と運用(Ops)の協力
  - 自動化を中心とした継続的デリバリー(CD)
  - フロー、フィードバック、**継続的学習と実験**
- **foundation 2: アジャイル**（2000年代～）
  - クリスタル、Extreme Programming、Scrum
  - アジャイルソフトウェア開発宣言（Agile Manifesto）
- **foundation 1: WFのリスク軽減手法**（1990年代～）
  - インクリメンタル、スパイラル、RUP
  - 継続的インテグレーション(CI)
  - 短いイテレーション、継続的フィードバック



技術的発展の階層構造

- **foundation 4: AIアシスト開発**（2020年代〜）
  - 人間とAIの協働で、ソフトウェアを効率的に創る
  - DevOpsの土台があって初めて効果を発揮
- **foundation 3: DevOps**（2010年代〜）
  - **アジャイルを土台に**、開発(Dev)と運用(Ops)の協力
  - 自動化を中心とした継続的デリバリー(CD)
  - フロー、フィードバック、**継続的学習と実験**
- **foundation 2: アジャイル**（2000年代〜）
  - クリスタル、Extreme Programming、Scrum
  - アジャイルソフトウェア開発宣言（Agile Manifesto）
- **foundation 1: WFのリスク軽減手法**（1990年代〜）
  - インクリメンタル、スパイラル、RUP
  - 継続的インテグレーション(CI)
  - 短いイテレーション、継続的フィードバック



技術的発展の階層構造

- **foundation 4: AIアシスト開発**（2020年代〜）
  - 人間とAIの協働で、ソフトウェアを効率的に創る
  - DevOpsの土台があって初めて効果を発揮
- **foundation 3: DevOps**（2010年代〜）
  - **アジャイルを土台に**、開発(Dev)と運用(Ops)の協力
  - 自動化を中心とした継続的デリバリー(CD)
  - フロー、フィードバック、**継続的学習と実験**
- **foundation 2: アジャイル**（2000年代〜）
  - クリスタル、Extreme Programming、Scrum
  - アジャイルソフトウェア開発宣言（Agile Manifesto）
- **foundation 1: WFのリスク軽減手法**（1990年代〜）
  - インクリメンタル、スパイラル、RUP
  - 継続的インテグレーション(CI)
  - 短いイテレーション、継続的フィードバック

技術というより、文化や作法
これらは積み重ね
一足飛びには手に入らない

Findy

ソフトウェア関連の論文・文献

ICTの進化

西暦

主な出来事

Manufacturing Industry in Japan

The New New Product Development Game (1986)

SCRUM Development Process（1995）

XP Explained : Embrace Change（1999）

The Scrum Guide（2010～）

アジャイルソフトウェア開発宣言 (2001)

Agile

DevOps (2009～)

GenAI (2021～)

1970 1981 1987 1989/02/01 1994/10/31 2001/5/18 2003/9/1 2005/10 2009/11/27 2011/7/16 2014/8/18 2017/6/22 2019/9/17 2021/3/15 2023/11/21 2025/6/25

1977 1979 1984 1988/3/1 1995/10/1 2000/12/1 2003/9/1 2005/10/7 2008/12/30 2010/12/12 2013/1/10 2016/10/6 2018/4/1 2019/9/30 2021/12/1 2023/11/28 2024/12/25

1975 1978 1979 1982 1989 1990/10/10 1992/3/13 1996/4/1 2002 2004/11/16 2010/7/27 2012/3/15 2013/8/1 2018/3/27 2021/8/1 2022/12/17 2024/7/11

1982 1996/4/1 2002/7/8 2004/7/16 2006/10/19 2010～ 2017～ 2021/12/10 2024/9/13

メインフレーム時代：
IBM System 360とVT100（1970～1980）

PC/AT互換機が誕生（1981～）

PC-9800シリーズ（1982～2003）

Apple Macintosh（1984）

ワークステーションの時代：
Sun SPARCstation
(1989～1994)

数字送信の開始による
ポケベルブーム（日本）
(1992～1996)

テレホーダイ
(1995～2023)

初代iPhone発表
(2007/1/9)

初代iPad発表
(2010/4/3)

Samsung
Galaxy S II
(2011/5/2)

Netflixが日本に上陸(2015/9/2)
※米国では2007/1にサービスイン

ARMアーキテクチャのSoC
Apple M1 生産（2020）

MacBook Pro M4 Max
(2024)

パソコン通信 / 携帯電話・PHS普及時代 / 携帯電話の多機能化、ブロードバンド時代 / スマートフォン普及時代（2010年、モバイル端末利用率がパソコン利用率を超える）

50年前 40年前 30年前 20年前 10年前

1972 1974 1976 1978 1980 1982 1984 1986 1988 1990 1992 1994 1996 1998 2000 2002 2004 2006 2008 2010 2012 2014 2016 2018 2020 2022 2024 2026

Point: 米国防総省(DoD)が与えた影響

1974年：米「TCP/IP」の最初の仕様がRFC 675として公開

1982年～：米国防総省(DoD)は全ての軍用コンピュータ網のためにTCP/IP標準を作成。その後、コンピューター産業...

UNIXを皮切りにほとんどの商用OSにTCP/IPが実装され...

★ピアソンショック（2013年8月1日）

1980年～：米国防総省(DoD)がウォーターフォールを採用

1980年後半～1990年前半：米国防総省(DoD)の発注したソフトウェアに問題が多発。米会計局でも多くの遅延／途中での挫折が発生

1997～2...年代：米国防総省(DoD)がCMMレベル3以上を...衛関連のソフトウェア調達条件として要求し始める。

2010年～：米国議会は2010会計年度国防権限法(NDAA 2010)において...速なITシステム獲得のための新たな取得プロセスの実施」(第804条)をDoDに命じ、事実上アジャイル型開発の原則を法制化

1970～1980年代：多くの企業や政府機関が「ウォーターフォールモデル」を公式な開発プロセスとして採用。
同時に、Royceの思惑と違う、誤解された「ウォーターフォール」が広まる

1990年代：ウォーターフォールのリスクを軽減する開発手法が次々と誕生（インクリメンタル、スパイラル、RUP など）

2000/3～：IT...バブル崩壊

2001/2：アジャイルソフトウェア開発宣言

2001～：米国を中心にアジャイル手法が急速に広まり、ウォーターフォールモデルを採用し続ける。

2010年以降：...のスタートアップはクラウドファーストで事業を立ち上げ、初めからアジャイル手法を採用

1970年代後半～1980年代末：日本車と家電がアメリカ市場を席巻。これが、徐々に日米貿易摩擦の火種に……

1988～：日本を含む諸外国へ「通商法スーパー301条」発動

1991～：バブル崩壊「失われた○○年」

2008/9～：リーマンショック

2009：Flickrのエンジニアである John AllspawとPaul Hammondが初めて...

2018：マイクロソフト... GitHub を 75 億ドルで買収

2020～2023/5：...19

2～：AI（GenAI）が前提の時代へ

1979～：日本の製造業の高品質、ものづくりの強さを研究。

1980年代後半～1990年代前半：UNIX戦争

1985～1990：国家プロジェクト「Σ計画」が頓挫

© The Deming Institute

“トリニトロン”
日本製品が欧米で人気

"ウォークマン"
1号機 TPS-L2
(1979)

Toyota Corolla
Liftback SR5 001
(1980～)

世界初のポータブル
CDプレーヤー
D-50（1984）

Java v1.0 正式リリース
Javaはオブジェクト指向と仮想マシン技術を普及させ、後の言語設計にも影響を与えた
(1996)

Linux 0.01 リリース
UNIXが商業化・断片化していく中、Linuxはオープンソースの力で統一的な開発基盤となり、世界中の技術革新を支えた
(1991/9/17)

Windows95 リリース
コンシューマ向けOSにTCP/IPが標準搭載されワークステーション並みに
(1995)

Bugzilla
リリース
(2000)

Jira
リリース
(2002)

Subversionリリース
(2000)

CVS 誕生
(1990)

(2004)

(2006)

(2004)

(2005)

欧州の自動車メーカが中心となって公開（2005）

1990～2000：第一次ブラウザ戦争。Internet Explorerの躍進、Netscape Navigatorの消滅

2009～2020：第二次ブラウザ戦争 Google Chromeの躍進

2025/1/3：Agile Alliance、PMBOKなどを策定するPMIに加盟

ソフトウェア関連の論文・文献

ICTの進化

西暦

主な出来事

Waterfall（1976〜）

1970 1976 1981 1987 1989/02/01 1994/10/31 1996/8 2001/5/18 2003/9/1 2005/10 2009/11/27 2011/7/16 2014/8/18 2017/6/22 2019/9/17 2021/3/15 2023/11/21 2025/6/25

1977 1979 1984 1988/3/1 1995/10/1 2000/12/1 2003/9/1 2005/10/7 2008/12/30 2010/10/12 2013/1/10 2016/10/6 2018/4/1 2019/9/30 2021/12/1 2023/11/28 2024/12/25

1978/5/1 1986 1989 1995 1999 2002 2004/11/16 2010/7/27 2012/3/15 2013/8/1 2018/3/27 2021/8/1 2022/12/17 2024/7/11

1975 1978 1979 1982 1990/10/10 1992/3/13 1996/4/1 2002/11/8 2004/7/16 2006/10/19 2010〜 2011/10/18 2014/4/22 2017/10/14 2018/11/22 2021/12/10 2024/9/13

パソコン通信 | 携帯電話・PHSの普及、インターネット黎明期 | 携帯電話の多機能化、ブロードバンド時代 | スマートフォン普及時代（2010年、モバイル端末利用率がパソコン利用率を超える）

メインフレーム時代：IBM System 360とVT100（1970〜1980）

PC/AT互換機が誕生（1981〜）

PC-9800シリーズ（1982〜2003）

Apple Macintosh（1984〜）

ワークステーションの時代：Sun SPARCstation（1989〜1994）

数字送信の開始によるポケベルブーム（日本）（1992〜1996）

テレホーダイ（1995〜2023）

初代iMac（1998〜）

F501i HYPERiモード開始（1999/2/22）

家庭向けADSL・FTTHの普及、ブロードバンド時代に突入（2001〜）

初代iPhone発表（2007/1/9）

初代iPad発表（2010/4/3）

Samsung Galaxy S II（2011/5/2）

Netflixが日本に上陸(2015/9/2)※🇺🇸では2007/1にサービスイン

ARMアーキテクチャのSoC Apple M1 生産（2020）

MacBook Pro M4 Max（2024）

50年前 | 40年前 | 30年前 | 20年前 | 10年前

1972 1974 1976 1978 1980 1982 1984 1986 1988 1990 1992 1994 1996 1998 2000 2002 2004 2006 2008 2010 2012 2014 2016 2018 2020 2022 2024 2026

1974年：🇺🇸「TCP/IP」の最初の仕様がRFC 675として公開

1982年〜：🇺🇸米国防総省(DoD)は全ての軍用コンピュータ網のためにTCP/IP標準を作成。その後、コンピューター産業に開放され、4.2 BSD UNIXを皮切りにほとんどの商用OSにTCP/IPが実装された。

★ピアソンショック（2013年8月1日）

Point: 米国防総省(DoD)が与えた影響

1980年〜：🇺🇸米国防総省(DoD)がウォーターフォールを採用

1980年後半〜1990年前半：🇺🇸米国防総省(DoD)の発注したソフトウェアに問題が多発。米会計局でも多くの遅延／途中での挫折が発生

1997〜2010年代：🇺🇸米国防総省(DoD)がCMMレベル3以上を防衛関連のソフトウェア調達条件として要求し始める。

2010年〜：🇺🇸米国議会は2010会計年度国防権限法(NDAA 2010)において「迅速なITシステム獲得のための新たな取得プロセスの実施」(第804条)をDoDに命じ、事実上アジャイル型開発の原則を法制化

1970〜1980年代：🌎多くの企業や政府機関が「ウォーターフォールモデル」を公式な開発プロセスとして採用。同時に、Royceの思惑と違う、誤解された「ウォーターフォール」が広まる

1990年代：🇺🇸ウォーターフォールのリスクを軽減する開発手法が次々と誕生（インクリメンタル、スパイラル、RUP など）

2000年代〜：🇺🇸米国を中心にでアジャイル手法が急速に広まり、ウォーターフォールは特にWeb系・スタートアップ企業ではほぼ使われなくなる。

2000年代〜：🇯🇵日本は大企業を中心に、ウォータフォールモデルを採用し続ける。

2020〜2023/5：🌎COVID-19

1970年代後半〜1980年代末：🇯🇵日本車と家電がアメリカ市場を席巻。これが、徐々に日米貿易摩擦の火種に……

1988〜：🇺🇸日本を含む諸外国へ「通商法スーパー301条」発動

2000/3〜：🇺🇸ITバブル崩壊

2010年以降〜：🇯🇵日本のスタートアップはクラウドファーストで事業を立ち上げ、初期からアジャイル手法を採用

1979〜：🇺🇸日本の製造業の高品質、ものづくりの強さを研究。

1991〜：🇯🇵バブル崩壊「失われた○○年」

2001/2/11：🌎アジャイルソフトウェア開発宣言

2000年代後半〜：🌎クラウドファースト・クラウドネイティブ時代に突入

2021〜：🌎AI（GenAI）が前提の時代へ

2008/9〜：🇺🇸リーマンショック

2018：🇺🇸マイクロソフト、GitHub を 75 億ドルで買収

1980年代後半〜1990年代前半：🌎UNIX戦争

1985〜1990：🇯🇵国家プロジェクト「Σ計画」が頓挫

2009：🇺🇸Flickrのエンジニアである John AllspawとPaul Hammondが初めてDevOpsに繋がる伝説的な講演を行う

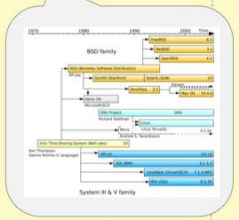

“トリニトロン”日本製品が欧米で人気


"ウォークマン"1号機 TPS-L2（1979）


Toyota Corolla Liftback SR5 001（1980〜）


世界初のポータブルCDプレーヤー D-50（1984）

© The Deming Institute



Java v1.0 正式リリース Javaはオブジェクト指向と仮想マシン技術を普及させ、後の言語設計にも影響を与えた（1996）

Linux 0.01 リリース UNIXが商業化・断片化していく中、Linuxはオープンソースの力で統一的な開発基盤となり、世界中の技術革新を支えた（1991/9/17）

Windows95 リリース コンシューマ向けOSにTCP/IPが標準搭載されワークステーション並みに（1995）

Bugzilla リリース（2000）

Jira リリース（2002）

Subversionリリース（2000）

CVS 誕生（1990）

flickr リリース（2004/2）

AWS サービス開始（2006）

Google Cloud サービス開始（2008）

Azure サービス開始（2010）

Selenium（2004）

REDMINE（2006）

trac（2004）

git（2005）

GitHub リリース（2008）

AUTOMOTIVE SPICE®

欧州の自動車メーカが中心となって公開（2005）

GitLab（2011）

Jenkins 誕生（2011）

GitHub Actions（2018）

DATADOG（2018）

GenAI

Findy Team+（2021/10）

GitHub Copilot（2021/6/29）

Gemini 1（2023/12/6）

ChatGPT 一般公開（2022/11/30）

Devin（2024/3/12）

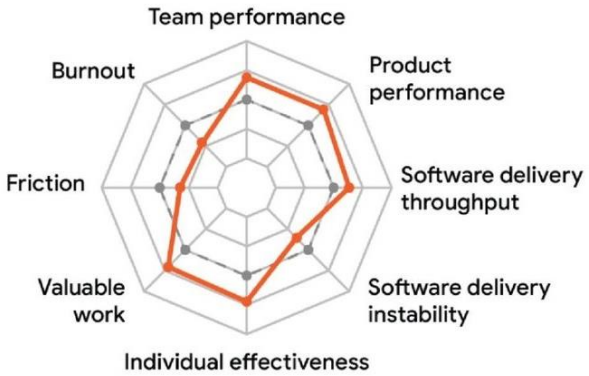
Agile Alliance + Project Management Institute.

2025/2〜：🌎Agile AllianceがPMBOKなどを策定するPMIに加盟

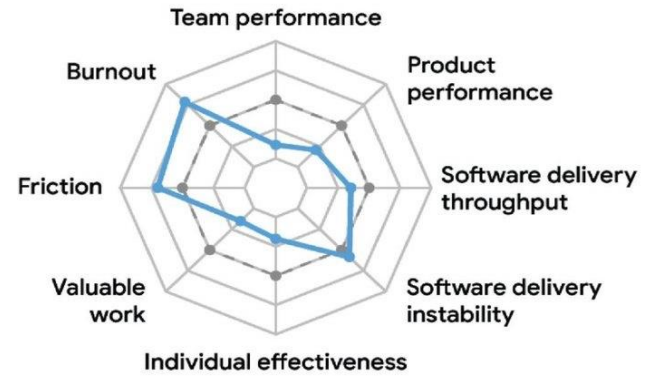
1990〜2000：🌎第一次ブラウザ戦争。Internet Explorerの躍進、Netscape Navigatorの消滅

2009〜2020：🌎第二次ブラウザ戦争 Google Chromeの躍進、 Internet Explorerの衰退

Cluster 7:
Harmonious high-achiever



Cluster 1:
Foundational challenges



Findy

Findy

業界の現状（日本）

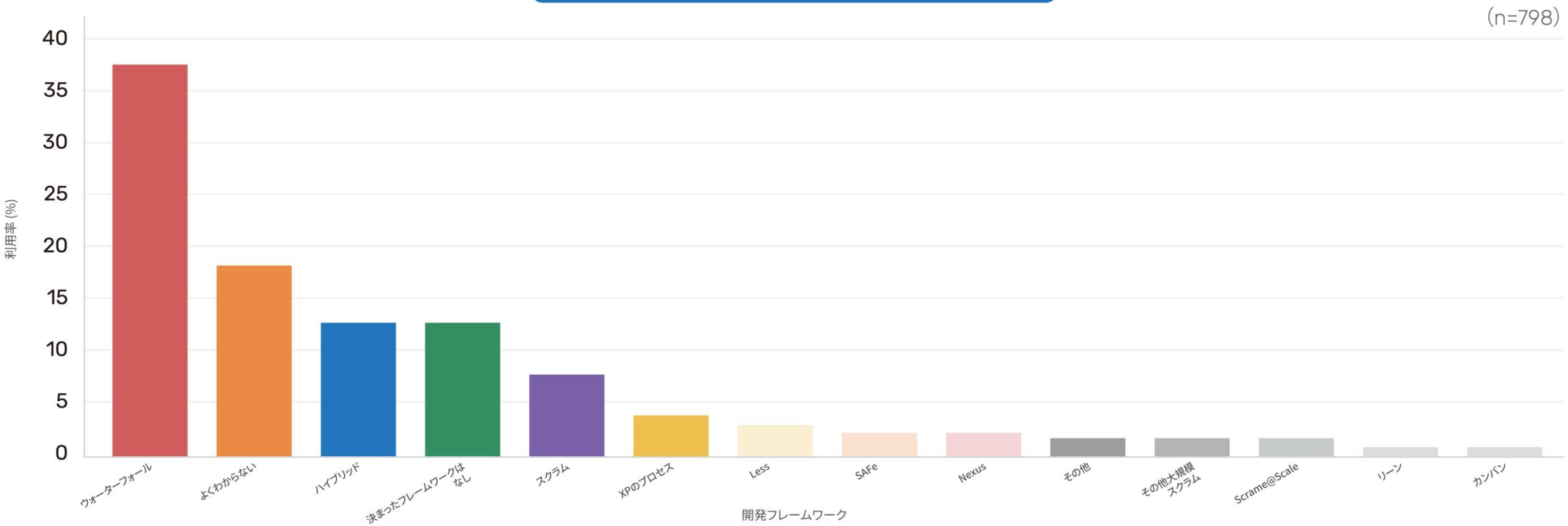
ソフトウェア開発における「開発生産性」に関する実態調査レポート

- 調査対象： ソフトウェア開発（組み込み開発を含む）に直接関わるエンジニア、プロダクトマネージャー、プロジェクトマネージャー、エンジニアリングマネージャー、開発責任者など
- 調査方法： インターネット調査
- 調査期間： 2025年4月2日(水)～2025年5月21日(水)
- 調査主体： ファインディ株式会社
- 実査委託先： GMOリサーチ&AI株式会社
- 有効回答数： 798名

※本調査はファインディ株式会社の利用ユーザーに対する調査ではないことをご留意ください。

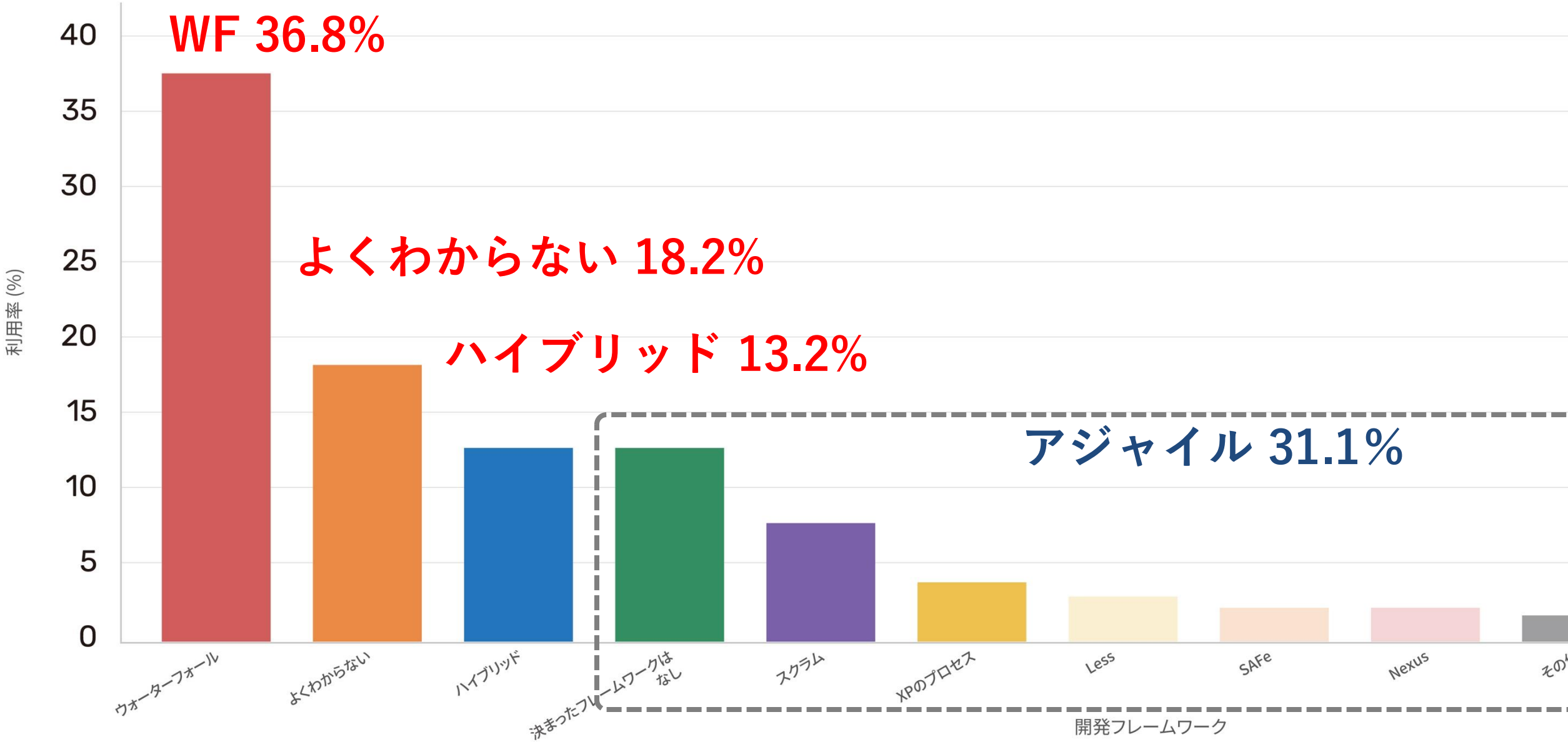
Findy

開発フレームワーク利用状況



Findy

開発フレームワーク利用状況



開発フレームワーク利用状況

開発フレームワーク	回答者数	利用率	統計的有意性	分類
ウォーターフォール	294名	36.8%	p < 0.001	従来型手法
開発フレームワークはよくわからない	145名	18.2%	p < 0.001	組織課題
ウォーターフォールとアジャイルのハイブリッド	105名	13.2%	p < 0.001	混合手法
【アジャイル開発】決まったフレームワークはない	105名	13.2%	p < 0.001	アジャイル系
【アジャイル開発】スクラム	54名	6.8%	p < 0.001	アジャイル系
【アジャイル開発】XPのプロセス	30名	3.8%	p < 0.001	アジャイル系
【アジャイル開発】大規模スクラム：LeSS	19名	2.4%	p < 0.05	アジャイル系
【アジャイル開発】大規模スクラム：SAFe	10名	1.3%	p < 0.05	アジャイル系
【アジャイル開発】大規模スクラム：Nexus	10名	1.3%	p < 0.05	アジャイル系
【アジャイル開発】大規模スクラム：Scrum@Scale	6名	0.8%	n.s.	アジャイル系
【アジャイル開発】大規模スクラム：その他	6名	0.8%	n.s.	アジャイル系
リーン	3名	0.4%	n.s.	アジャイル系
カンバン	2名	0.3%	n.s.	アジャイル系
その他	8名	1.0%	n.s.	その他

ソフトウェア関連の論文・文献

ICTの進化

西暦

主な出来事

Manufacturing Industry in Japan

The New New Product Development Game (1986)

SCRUM Development Process（1995）

XP Explained : Embrace Change（1999）

The Scrum Guide（2010～）

アジャイルソフトウェア開発宣言 (2001)

Agile

DevOps (2009～)

GenAI (2021～)

1970 1981 1987 1989/02/01 1994/10/31 2001/5/18 2003/9/1 2005/10 2009/11/27 2011/7/16 2014/8/18 2017/6/22 2019/9/17 2021/3/15 2023/11/21 2025/6/25

1977 1979 1984 1988/3/1 1995/10/1 2000/12/1 2003/9/1 2005/10/7 2008/12/30 2010/12/12 2013/1/10 2016/10/6 2018/4/1 2019/9/30 2021/12/1 2023/11/28 2024/12/25

1975 1978 1979 1982 1989 1990/10/10 1992/3/13 1996/4/1 2002 2004/11/16 2010/7/27 2012/3/15 2013/8/1 2018/3/27 2021/8/1 2022/12/17 2024/7/11

1982 1996/4/1 2002/7/8 2004/7/16 2006/10/19 2010～ 2017～ 2021/12/10 2024/9/13

メインフレーム時代：
IBM System 360とVT100（1970～1980）

PC/AT互換機が誕生（1981～）

PC-9800シリーズ（1982～2003）

Apple Macintosh（1984）

ワークステーションの時代：
Sun SPARCstation
(1989～1994)

数字送信の開始による
ポケベルブーム（日本）
(1992～1996)

テレホーダイ
(1995～2023)

初代iPhone発表
(2007/1/9)

初代iPad発表
(2010/4/3)

Samsung
Galaxy S II
(2011/5/2)

Netflixが日本に上陸(2015/9/2)
※米国では2007/1にサービスイン

ARMアーキテクチャのSoC
Apple M1 生産（2020）

MacBook Pro M4 Max
(2024)

パソコン通信 / 携帯電話・PHS普及時代 / 携帯電話の多機能化、ブロードバンド時代 / スマートフォン普及時代（2010年、モバイル端末利用率がパソコン利用率を超える）

50年前 40年前 30年前 20年前 10年前

1972 1974 1976 1978 1980 1982 1984 1986 1988 1990 1992 1994 1996 1998 2000 2002 2004 2006 2008 2010 2012 2014 2016 2018 2020 2022 2024 2026

Point: 米国防総省(DoD)が与えた影響

1974年：米「TCP/IP」の最初の仕様がRFC 675として公開

1982年～：米国防総省(DoD)は全ての軍用コンピュータ網のためにTCP/IP標準を作成。その後、コンピューター産業...

UNIXを皮切りにほとんどの商用OSにTCP/IPが実装され...

★ピアソンショック（2013年8月1日）

1980年～：米国防総省(DoD)がウォーターフォールを採用

1980年後半～1990年前半：米国防総省(DoD)の発注したソフトウェアに問題が多発。米会計局でも多くの遅延／途中での挫折が発生

1997～2...年代：米国防総省(DoD)がCMMレベル3以上を...衛関連のソフトウェア調達条件として要求し始める。

2010年～：米国議会は2010会計年度国防権限法(NDAA 2010)において...速なITシステム獲得のための新たな取得プロセスの実施」(第804条)をDoDに命じ、事実上アジャイル型開発の原則を法制化

1970～1980年代：多くの企業や政府機関が「ウォーターフォールモデル」を公式な開発プロセスとして採用。
同時に、Royceの思惑と違う、誤解された「ウォーターフォール」が広まる

1990年代：ウォーターフォールのリスクを軽減する開発手法が次々と誕生（インクリメンタル、スパイラル、RUP など）

2000/3～：IT...バブル崩壊

2001/2：アジャイルソフトウェア開発宣言

2001～：米国を中心にアジャイル手法が急速に広まり、ウォーターフォールモデルを採用し続ける。

2010年以降：...のスタートアップはクラウドファーストで事業を立ち上げ、初めからアジャイル手法を採用

1970年代後半～1980年代末：日本車と家電がアメリカ市場を席巻。これが、徐々に日米貿易摩擦の火種に……

1988～：日本を含む諸外国へ「通商法スーパー301条」発動

1991～：バブル崩壊「失われた○○年」

2008/9～：リーマンショック

2009：Flickrのエンジニアである John AllspawとPaul Hammondが初めて...

2018：マイクロソフト... GitHub を 75 億ドルで買収

2020～2023/5：...19

2～：AI（GenAI）が前提の時代へ

1979～：日本の製造業の高品質、ものづくりの強さを研究。

1980年代後半～1990年代前半：UNIX戦争

1985～1990：国家プロジェクト「Σ計画」が頓挫

© The Deming Institute

“トリニトロン”
日本製品が欧米で人気

"ウォークマン"
1号機 TPS-L2
(1979)

Toyota Corolla
Liftback SR5 001
(1980～)

世界初のポータブル
CDプレーヤー
D-50（1984）

Java v1.0 正式リリース
Javaはオブジェクト指向と仮想マシン技術を普及させ、後の言語設計にも影響を与えた
(1996)

Linux 0.01 リリース
UNIXが商業化・断片化していく中、Linuxはオープンソースの力で統一的な開発基盤となり、世界中の技術革新を支えた
(1991/9/17)

Windows95 リリース
コンシューマ向けOSにTCP/IPが標準搭載されワークステーション並みに
(1995)

Bugzilla
リリース
(2000)

Jira
リリース
(2002)

Subversionリリース
(2000)

CVS 誕生
(1990)

(2004)

(2006)

(2004)

(2005)

欧州の自動車メーカが中心となって公開（2005）

1990～2000：第一次ブラウザ戦争。Internet Explorerの躍進、Netscape Navigatorの消滅

2009～2020：第二次ブラウザ戦争 Google Chromeの躍進

2025/1/3：Agile Alliance、PMBOKなどを策定するPMIに加盟

ソフトウェア関連の論文・文献

ICTの進化

西暦

主な出来事

Waterfall（1976〜）

1970 1976 1981 1987 1989/02/01 1994/10/31 1996/8 2001/5/18 2003/9/1 2005/10 2009/11/27 2011/7/16 2014/8/18 2017/6/22 2019/9/17 2021/3/15 2023/11/21 2025/6/25

1977 1979 1984 1988/3/1 1995/10/1 2000/12/1 2003/9/1 2005/10/7 2008/12/30 2010/10/12 2013/1/10 2016/10/6 2018/4/1 2019/9/30 2021/12/1 2023/11/28 2024/12/25

1978/5/1 1986 1989 1995 1999 2002 2004/11/16 2010/7/27 2012/3/15 2013/8/1 2018/3/27 2021/8/1 2022/12/17 2024/7/11

1975 1978 1979 1982 1990/10/10 1992/3/13 1996/4/1 2002/11/8 2004/7/16 2006/10/19 2010〜 2011/10/18 2014/4/22 2017/10/14 2018/11/22 2021/12/10 2024/9/13

パソコン通信 | 携帯電話・PHSの普及、インターネット黎明期 | 携帯電話の多機能化、ブロードバンド時代 | スマートフォン普及時代（2010年、モバイル端末利用率がパソコン利用率を超える）

メインフレーム時代：IBM System 360とVT100（1970〜1980）

PC/AT互換機が誕生（1981〜）

PC-9800シリーズ（1982〜2003）

Apple Macintosh（1984〜）

ワークステーションの時代：Sun SPARCstation（1989〜1994）

数字送信の開始によるポケベルブーム（日本）（1992〜1996）

テレホーダイ（1995〜2023）

初代iMac（1998〜）

F501i HYPERiモード開始（1999/2/22）

家庭向けADSL・FTTHの普及、ブロードバンド時代に突入（2001〜）

初代iPhone発表（2007/1/9）

初代iPad発表（2010/4/3）

Samsung Galaxy S II（2011/5/2）

Netflixが日本に上陸(2015/9/2)※🇺🇸では2007/1にサービスイン

ARMアーキテクチャのSoC Apple M1 生産（2020）

MacBook Pro M4 Max（2024）

50年前 | 40年前 | 30年前 | 20年前 | 10年前

1972 1974 1976 1978 1980 1982 1984 1986 1988 1990 1992 1994 1996 1998 2000 2002 2004 2006 2008 2010 2012 2014 2016 2018 2020 2022 2024 2026

1974年：🇺🇸「TCP/IP」の最初の仕様がRFC 675として公開

1982年〜：🇺🇸米国防総省(DoD)は全ての軍用コンピュータ網のためにTCP/IP標準を作成。その後、コンピューター産業に開放され、4.2 BSD UNIXを皮切りにほとんどの商用OSにTCP/IPが実装された。

★ピアソンショック（2013年8月1日）

Point: 米国防総省(DoD)が与えた影響

1980年〜：🇺🇸米国防総省(DoD)がウォーターフォールを採用

1980年後半〜1990年前半：🇺🇸米国防総省(DoD)の発注したソフトウェアに問題が多発。米会計局でも多くの遅延／途中での挫折が発生

1997〜2010年代：🇺🇸米国防総省(DoD)がCMMレベル3以上を防衛関連のソフトウェア調達条件として要求し始める。

2010年〜：🇺🇸米国議会は2010会計年度国防権限法(NDAA 2010)において「迅速なITシステム獲得のための新たな取得プロセスの実施」(第804条)をDoDに命じ、事実上アジャイル型開発の原則を法制化

1970〜1980年代：🌎多くの企業や政府機関が「ウォーターフォールモデル」を公式な開発プロセスとして採用。同時に、Royceの思惑と違う、誤解された「ウォーターフォール」が広まる

1990年代：🇺🇸ウォーターフォールのリスクを軽減する開発手法が次々と誕生（インクリメンタル、スパイラル、RUP など）

2000年代〜：🇺🇸米国を中心にでアジャイル手法が急速に広まり、ウォーターフォールは特にWeb系・スタートアップ企業ではほぼ使われなくなる。

2000年代〜：🇯🇵日本は大企業を中心に、ウォータフォールモデルを採用し続ける。

2020〜2023/5：🌎COVID-19

1970年代後半〜1980年代末：🇯🇵日本車と家電がアメリカ市場を席巻。これが、徐々に日米貿易摩擦の火種に……

1988〜：🇺🇸日本を含む諸外国へ「通商法スーパー301条」発動

2000/3〜：🇺🇸ITバブル崩壊

2010年以降〜：🇯🇵日本のスタートアップはクラウドファーストで事業を立ち上げ、初期からアジャイル手法を採用

1979〜：🇺🇸日本の製造業の高品質、ものづくりの強さを研究。

1991〜：🇯🇵バブル崩壊「失われた○○年」

2001/2/11：🌎アジャイルソフトウェア開発宣言

2000年代後半〜：🌎クラウドファースト・クラウドネイティブ時代に突入

2021〜：🌎AI（GenAI）が前提の時代へ

2008/9〜：🇺🇸リーマンショック

2018：🇺🇸マイクロソフト、GitHub を 75 億ドルで買収

1980年代後半〜1990年代前半：🌎UNIX戦争

1985〜1990：🇯🇵国家プロジェクト「Σ計画」が頓挫

2009：🇺🇸Flickrのエンジニアである John AllspawとPaul Hammondが初めてDevOpsに繋がる伝説的な講演を行う

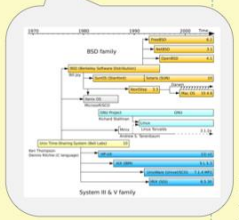

“トリニトロン”日本製品が欧米で人気


"ウォークマン"1号機 TPS-L2（1979）


Toyota Corolla Liftback SR5 001（1980〜）


世界初のポータブルCDプレーヤー D-50（1984）

© The Deming Institute



Java v1.0 正式リリース Javaはオブジェクト指向と仮想マシン技術を普及させ、後の言語設計にも影響を与えた（1996）

Linux 0.01 リリース UNIXが商業化・断片化していく中、Linuxはオープンソースの力で統一的な開発基盤となり、世界中の技術革新を支えた（1991/9/17）

Windows95 リリース コンシューマ向けOSにTCP/IPが標準搭載されワークステーション並みに（1995）

Bugzilla リリース（2000）

Jira リリース（2002）

Subversionリリース（2000）

CVS 誕生（1990）

flickr リリース（2004/2）

AWS サービス開始（2006）

Google Cloud サービス開始（2008）

Azure サービス開始（2010）

Selenium（2004）

REDMINE（2006）

trac（2004）

git（2005）

GitHub リリース（2008）

AUTOMOTIVE SPICE®

欧州の自動車メーカが中心となって公開（2005）

GitLab（2011）

Jenkins 誕生（2011）

GitHub Actions（2018）

DATADOG（2018）

GenAI

Findy Team+（2021/10）

GitHub Copilot（2021/6/29）

Gemini 1（2023/12/6）

ChatGPT 一般公開（2022/11/30）

Devin（2024/3/12）

Agile Alliance + Project Management Institute.

2025/2〜：🌎Agile AllianceがPMBOKなどを策定するPMIに加盟

1990〜2000：🌎第一次ブラウザ戦争。Internet Explorerの躍進、Netscape Navigatorの消滅

2009〜2020：🌎第二次ブラウザ戦争 Google Chromeの躍進、 Internet Explorerの衰退

Findy

「何を測るべきか」の共通認識の欠如

開発生産性指標の認知度と活用率

- 今回の調査では、開発生産性の指標について「認知度」と「活用状況」の2つの観点から質問しました。具体的には、広く知られる用語や指標について「知っているか」と「実務で活用しているか」を調べました。

> *1. 以下の指標やフレームワークのうち、あなたが**「知っている（名前を聞いたことがある、概要を知っている等）」もの**をすべて選んでください。［複数回答］*
> *2. 以下の指標やフレームワークのうち、あなたの組織やチームで**「実際に活用している」もの**をすべて選んでください。［複数回答］*

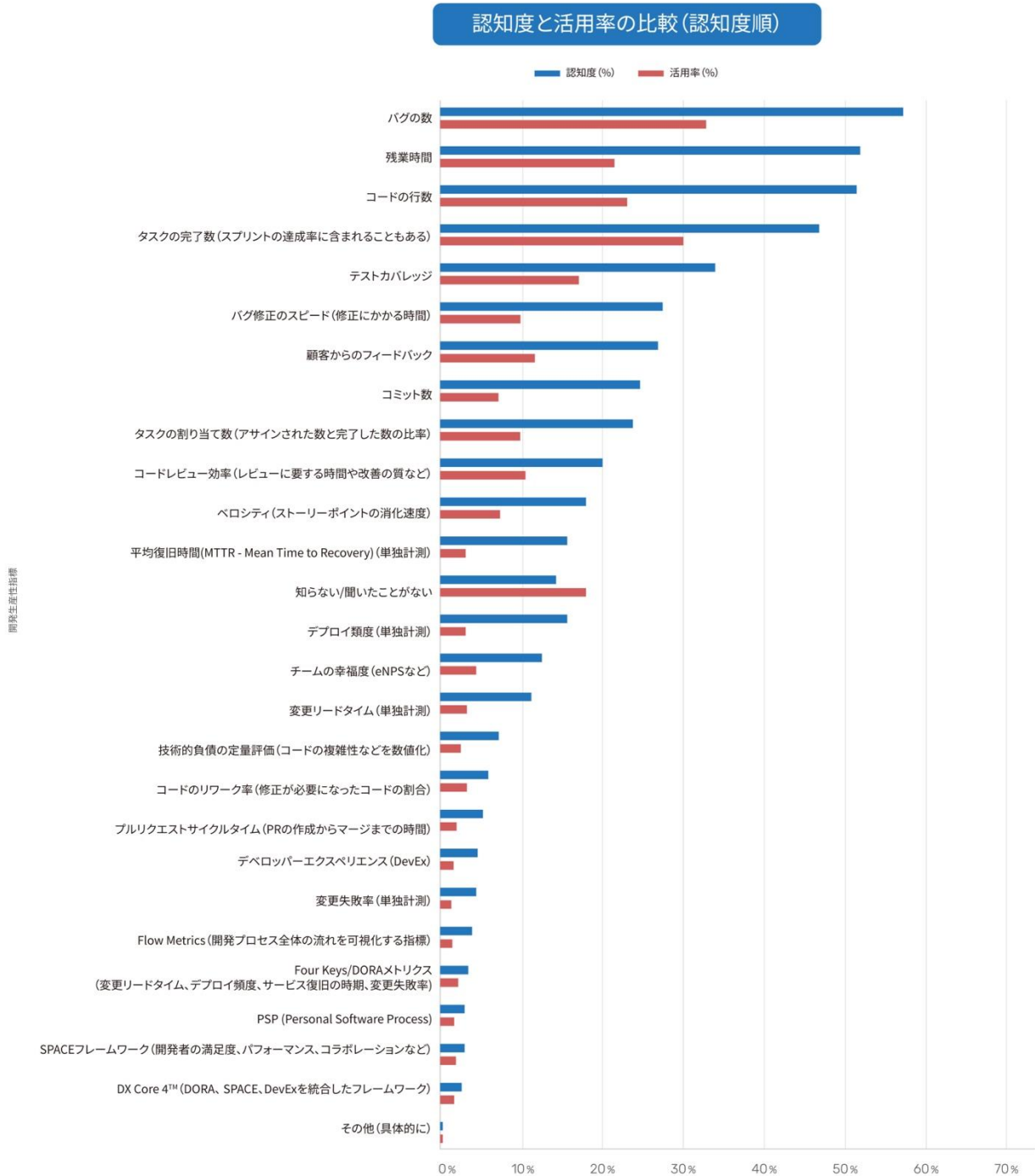
- 調査の包括性を担保するため、あえて適切でない指標や重複する用語も選択肢に含めました。これにより、単なる認知度だけでなく、指標に対する理解度や認識の正確性も測定できる設計としました。

開発生産性にふさわしい指標（14項目）

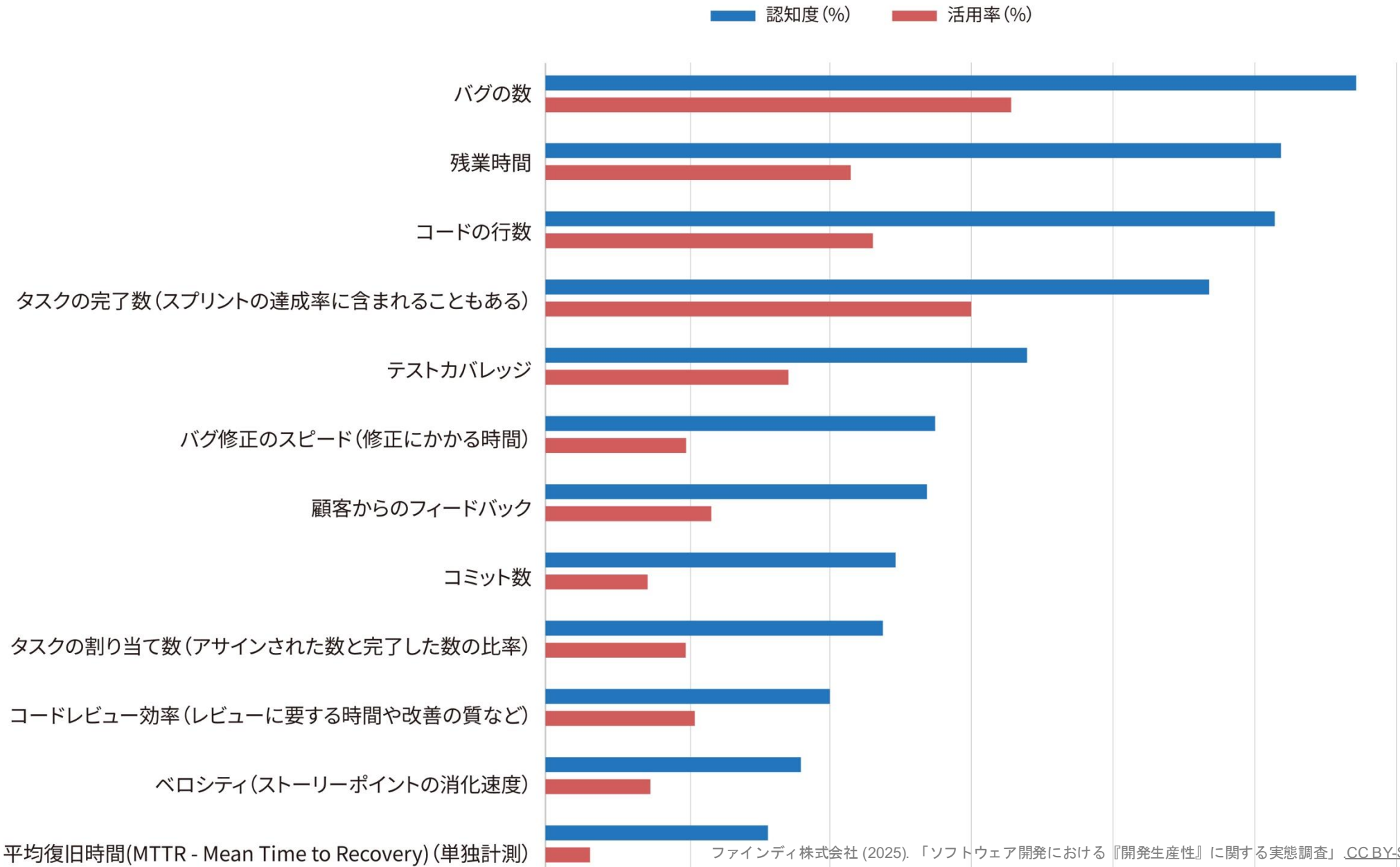
- 現代的な開発生産性測定において科学的根拠を持つ指標群
  1. Four Keys / DORAメトリクス – 実証研究に基づく包括的生産性指標
  2. 変更リードタイム – 要求から価値提供までの時間効率
  3. デプロイ頻度 – 継続的価値提供能力の測定
  4. 平均復旧時間（MTTR） – システム回復力と運用効率
  5. 変更失敗率 – 品質と速度の最適バランス
  6. SPACEフレームワーク – 多次元的生産性評価手法
  7. デベロッパーエクスペリエンス（DevEx） – 開発者効率性の統合指標
  8. DX Core 4™ – 複数フレームワークの統合測定
- リソース効率ではなく、作業の流れ（フロー）に焦点を当てた指標群
  9. Flow Metrics – 開発プロセス全体の流れの可視化と最適化
  10. プルリクエストサイクルタイム – コード統合プロセスの効率性
  11. コードレビュー効率 – 品質担保における時間効率
  12. バグ修正のスピード – 問題解決プロセスの効率性
- 長期的な生産性維持に関わる指標
  13. 技術的負債の定量評価 – 将来の生産性への影響度測定
  14. コードのリワーク率 – プロセス品質と手戻り作業の削減度

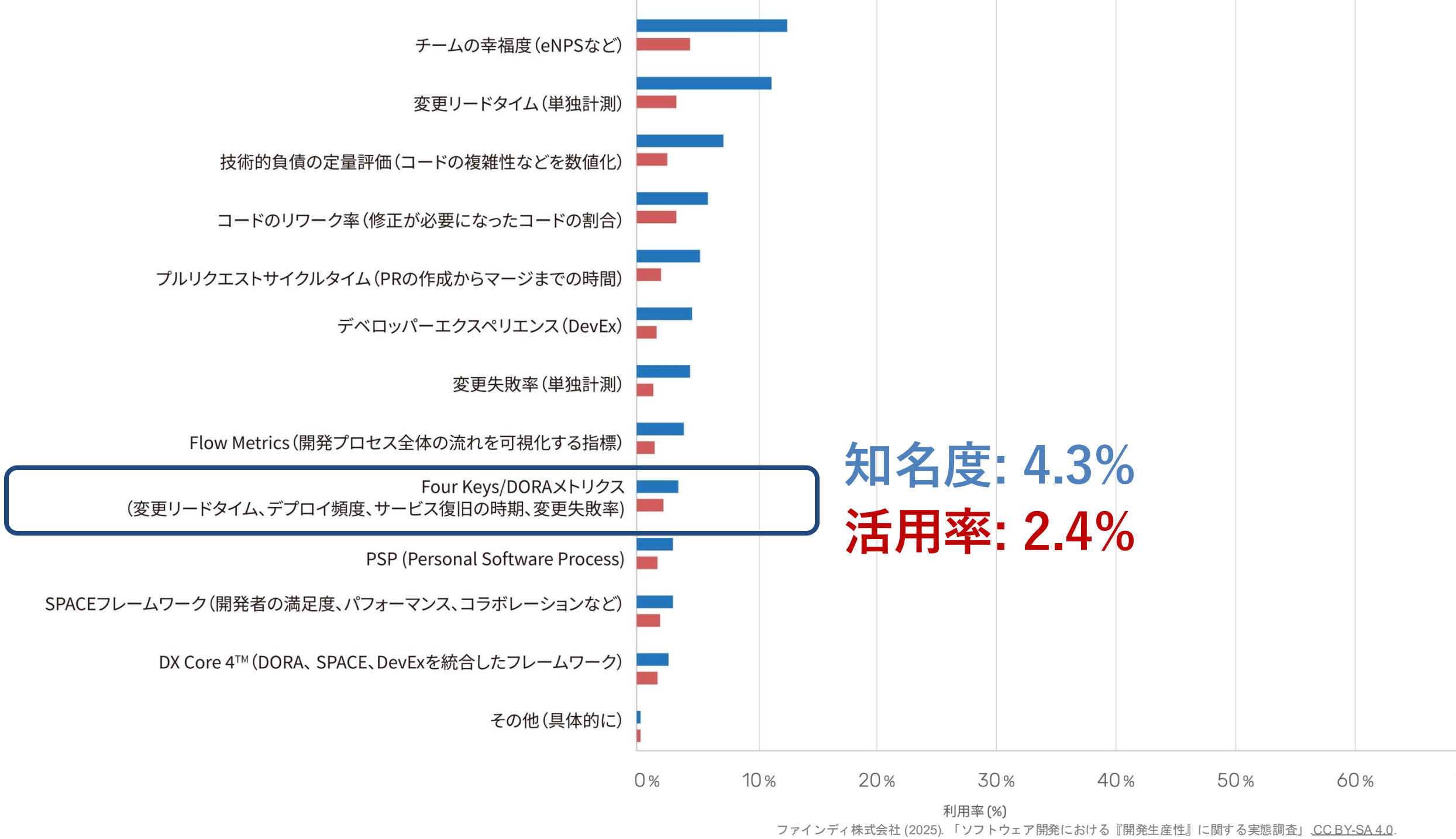
開発生産性としてはふさわしくない指標（11項目）

- 作業量と開発量を測定する指標
  - これらは結果のみに着目しており、開発プロセスの効率や価値創出を正しく評価できない
    1. コードの行数 – 1000行のコードを書くより10行で解決できる方が生産的な場合が多い
    2. コミット数 – 活動量を示すのみで、品質や生み出される価値とは無関係
    3. ベロシティ（ストーリーポイントの消化速度） – チーム内の作業量予測ツールであり、生産性指標としては適切ではない
- リソースの使用効率を測る指標
  - 真の生産性向上には結びつかないことが多い
    4. 残業時間 – 投入時間の増加は生産性の低下を示唆する場合が多い
    5. タスクの完了数 – 量的成果であり、価値や効率性を反映しない
    6. タスクの割り当て数（比率） – リソース効率重視は、必要な"ゆとり"（Slack）を排除する
- 品質や満足度を測る指標
  - 重要な指標ではあるが、生産性を直接測るものではない
    7. バグの数 – 品質指標であり、生産性とは異なる概念
    8. テストカバレッジ – 品質管理手法の一つ
    9. チームの幸福度（eNPS） – 生産性に影響するが間接的指標
    10. 顧客からのフィードバック – 価値検証の手段であり生産性測定ではない
    11. PSP（Personal Software Process） – 個人レベルのプロセス改善手法



認知度と活用率の比較（認知度順）

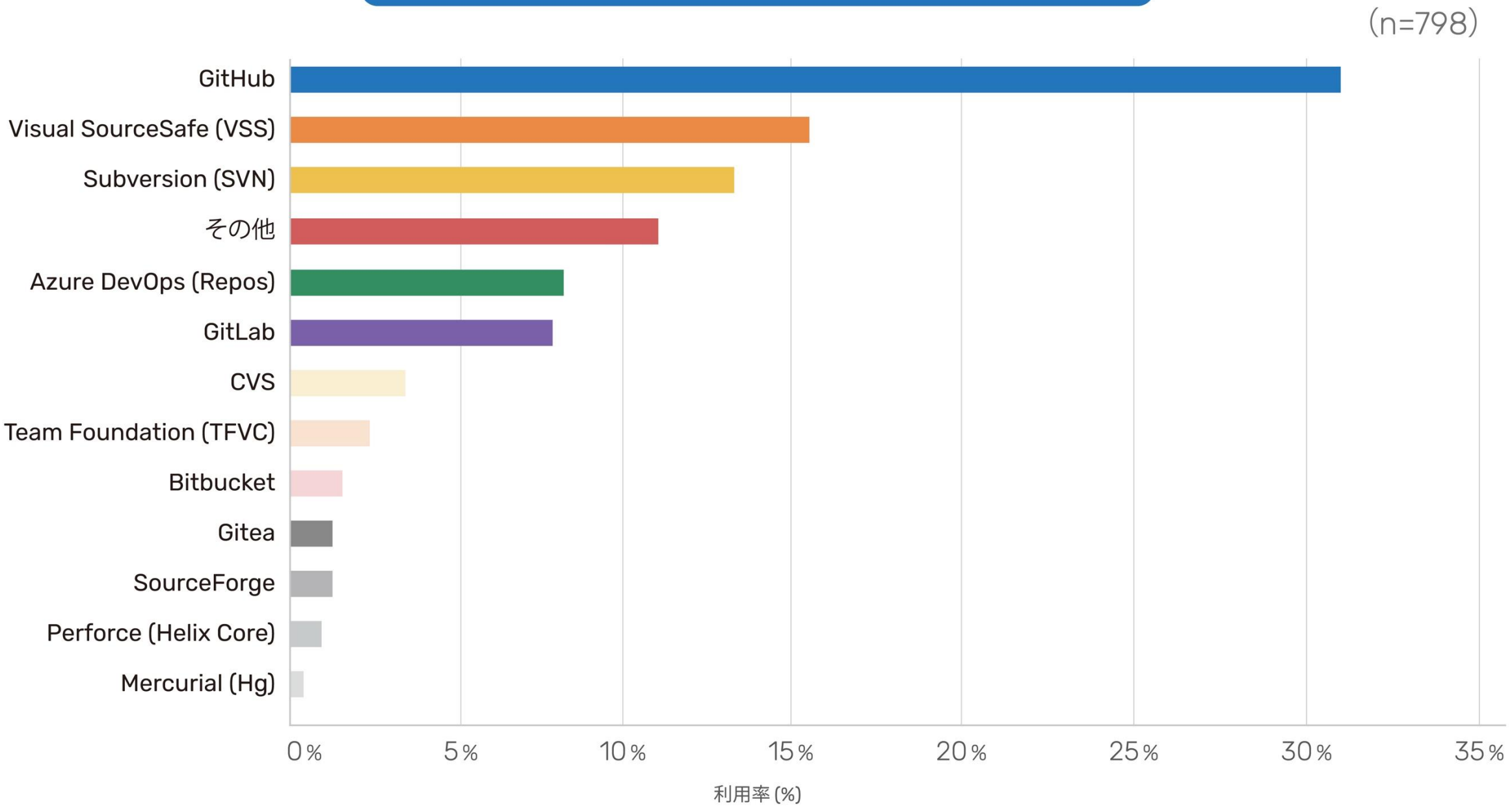




Findy

ソースコード管理ツールとAI活用の関係

ソースコード管理ツール利用状況



ソースコード管理ツール利用状況

ツール	利用率	サポート状況	最終更新	セキュリティリスク
Visual SourceSafe (VSS)	15.8%(126人)	サポート完全終了	2005年10月	極めて高い
CVS	3.6%(29人)	事実上の開発停止	2008年5月8日	高い
Subversion (SVN)	13.7%(109人)	限定的メンテナンス	2024年12月8日	中程度

- 従来型ツールの利用は「古い」以上に重大なリスクです。Visual SourceSafeは2005年に更新終了していますが、126人（15.8%）が依然使用していました。
- これらの利用者やSubversion（13.7%）、CVS（3.6%）の組織は、CopilotなどAI時代の支援ツールを活用できず、競争力格差拡大のリスクに直面しています。

従来型ツールの利用による技術的制約とは

■ API仕様の違いによる機能制限

- Visual SourceSafe: COM APIベースのアクセス方式（2005年設計のため、現代的なREST API仕様には非対応）
- Subversion: WebDAV/HTTP APIを実装（一部のJSON形式APIには対応しているが、現代的なWebhook機能は制限的）

→ 結果: **AIツールによるプログラム的なアクセスにおいて、一部機能が制限される場合があります**

■ メタデータ構造の違いによるAI学習への影響

- 従来型ツール: ファイル変更履歴を中心とした情報構造（コミット単位の情報は限定的）
- モダンツール: コミット意図、レビュー履歴、イシュー関連付けなどの豊富なコンテキスト情報を構造化

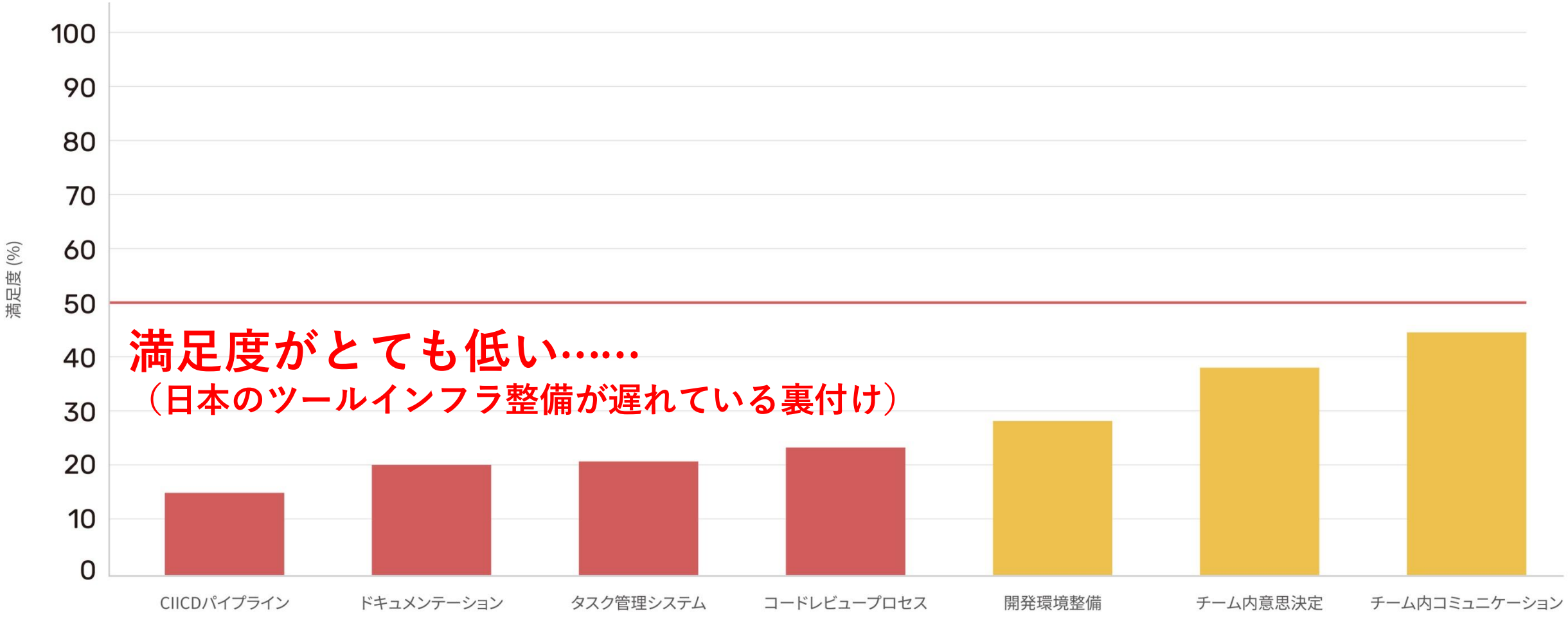
→ 結果: **AIによるコードコンテキストの理解において、活用できる情報量に制約が生じる場合があります**

■ 開発エディタとの統合における制約

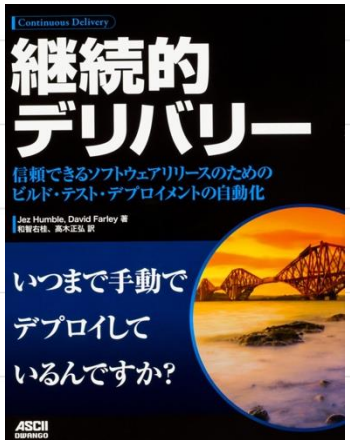
- VS Code、JetBrains IDEs: Git統合を前提とした拡張機能エコシステムが充実
- Cursor: Gitリポジトリ構造に最適化されたAI機能を提供

→ 結果: **従来型ツールでは最新の開発環境の一部機能を十分に活用できない場合があります**

開発環境・プロセス項目


満足度 (%)
100
90
80
70
60
50
40
30
20
10
0
満足度がとても低い……
（日本のツールインフラ整備が遅れている裏付け）
CIICDパイプライン
ドキュメンテーション
タスク管理システム
コードレビュープロセス
開発環境整備
チーム内意思決定
チーム内コミュニケーション

開発環境・プロセス項目



CDは約15年前に登場していた技術だが……

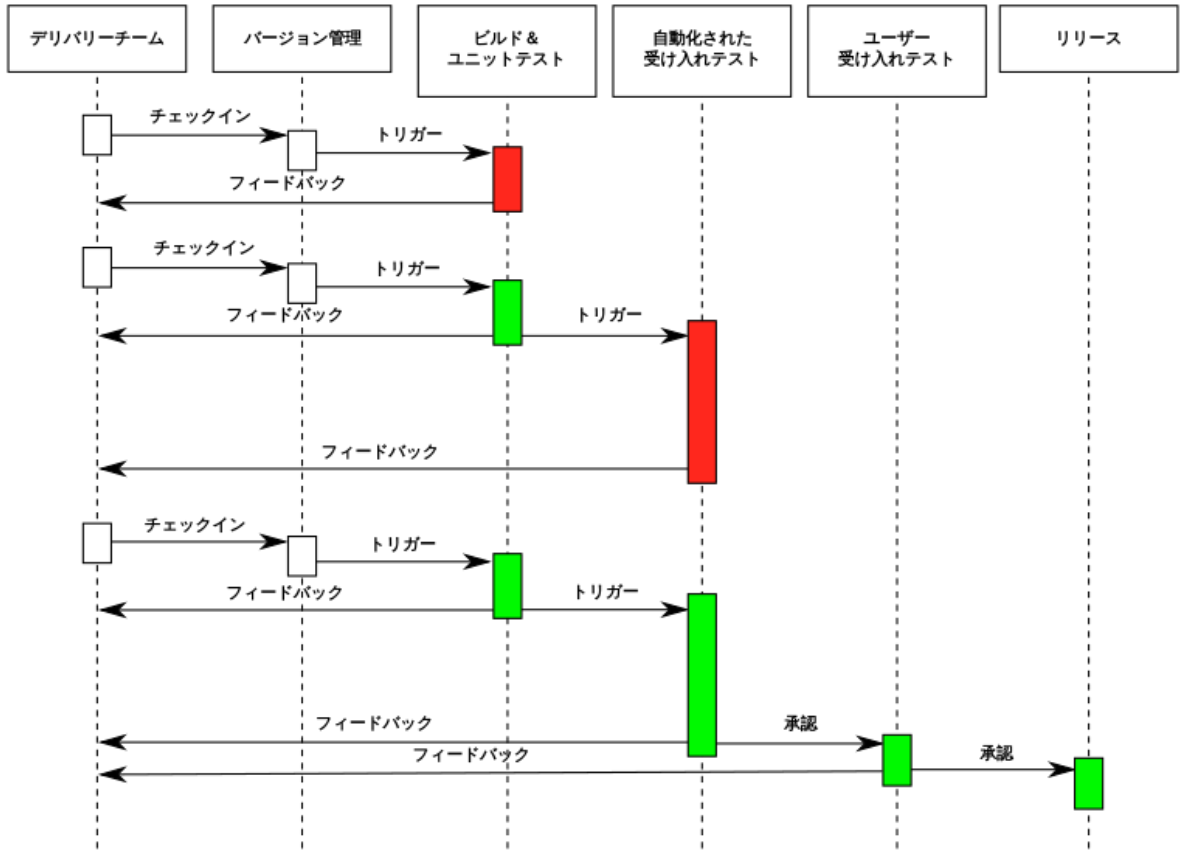
（2010年、日本語訳は2012年）

満足度がとても低い……

（日本のツールインフラ整備が遅れている裏付け）

3. 継続的デリバリー (Continuous Delivery / CD) の効果の証明

- Four Keys で測定されるパフォーマンス（速度と安定性の両方）を高める上で、バージョン管理、自動テスト、自動デプロイといった CD の具体的なプラクティスが直接的に貢献することを科学的に証明しました。
- CD が単なる理想論ではなく、ビジネス成果に繋がる実践であることの説得力を高め、その普及を後押ししました。



Findy

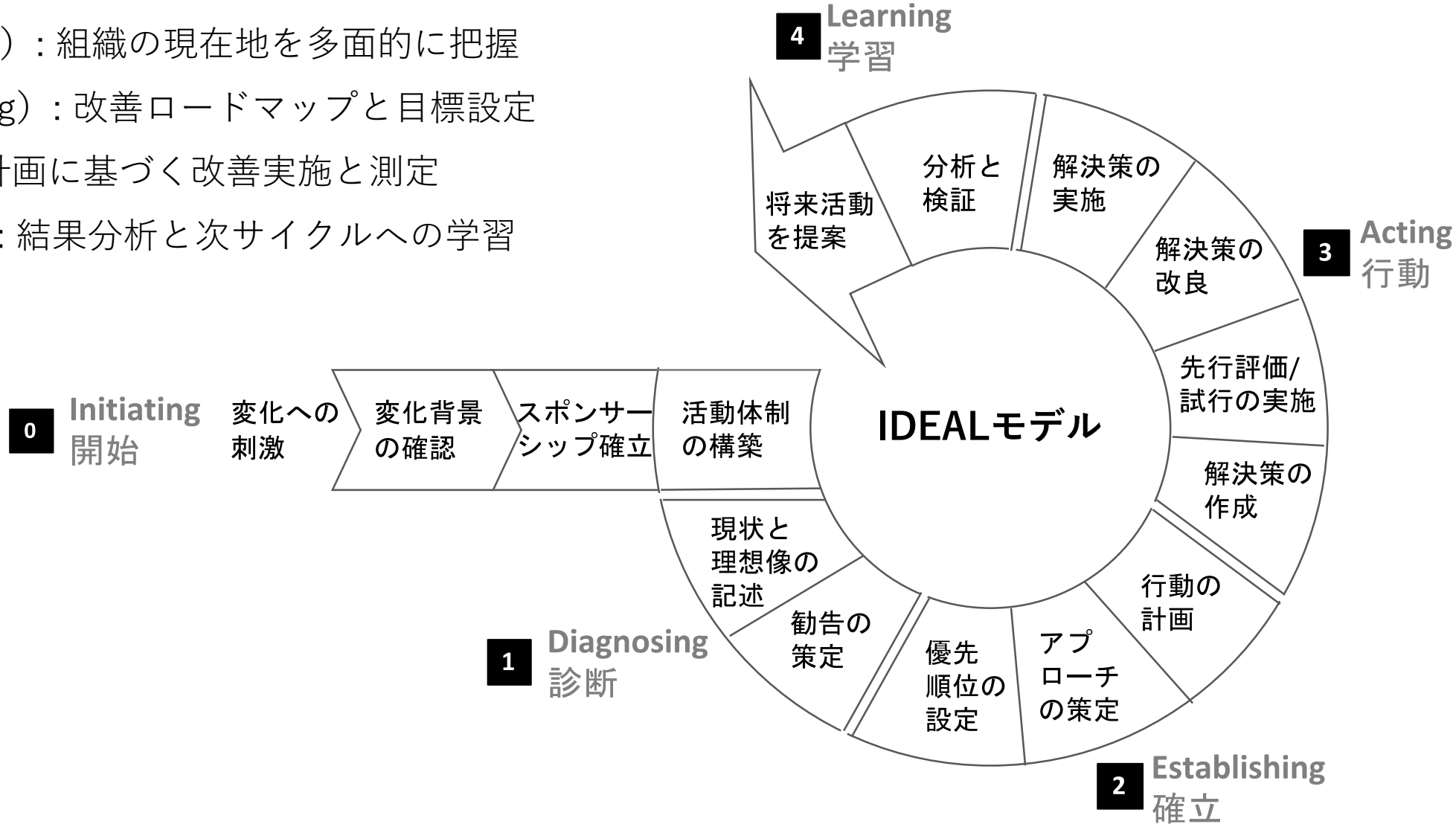
IDEALモデルの再発見

IDEALモデルとは

- 1996年 Carnegie Mellon大学 Software Engineering Institute（SEI）が開発
- ソフトウェアプロセス改善のための体系的フレームワーク
- CMMIなどの成熟度モデルと組み合わせて使用される前提で設計

IDEALモデルとは

- 開始（**I**nitiating）：スポンサー確保、推進体制構築
- 診断（**D**iagnosing）：組織の現在地を多面的に把握
- 確立（**E**stablishing）：改善ロードマップと目標設定
- 行動（**A**cting）：計画に基づく改善実施と測定
- 学習（**L**earning）：結果分析と次サイクルへの学習



IDEALモデル - なぜ知られていないのか

1. 認知度の問題: 専門家しか知らない
   - SEI（Software Engineering Institute）の技術報告書として発行
   - アカデミック/プロセス改善専門家の領域に留まる
   - アジャイル・DevOpsコミュニティでは言及されず（OODAとかの方が有名）
   - 日本語での体系的解説がほぼ存在しない
2. イメージの問題: 「古典的で重い手法」という誤解
   - 1996年公開 → CMMI（Capability Maturity Model Integration）時代の「硬い方法論」と混同
   - しかし実態は軽量で実践的なフレームワーク
   - PDCAより具体的、Six Sigmaより組織的
3. 統合の欠如: Four Keysとの接続が見えなかった
   - IDEAL: 組織的改善プロセス
   - Four Keys: 測定指標（2018年普及、ただし日本での認知度は依然低い）
   - どちらも個別に存在するが、統合例がほとんど示されていない
     → 測定に基づく科学的改善の実現方法が不明瞭
   - 両者とも「知る人ぞ知る」状態で、組み合わせの威力が伝わっていない

今こそ注目すべき: 本質は現代の継続的改善と完全一致。Four Keysという測定基盤と組み合わせることで、IDEALの真価が発揮される。

なぜIDEALモデルなのか

- 改善フレームワークとして思い浮かぶもの: PDCA、Six Sigma/DMAIC、PCMなど
- 今回の目的に照らして必要な要件:
  1. 組織的合意形成プロセスが明示されているか
  2. Four Keys等、必要な指標を柔軟に設定できるか
  3. 文化・ケイパビリティを技術と統合して扱えるか
- これらを満たすのは？ → IDEALしかない

主要フレームワークとの比較

- 記号の意味: ◎=要件を満たす, ○=概ね満たす, △=一部のみ, ×=欠如
- 各モデルの前提:
  - **PDCA**: Plan-Do-Check-Act（改善の基本サイクル）
  - **Six Sigma**: DMAICサイクルを用いた品質改善手法
  - **PCM**: Project Cycle Management（プロジェクト型の段階管理手法、主に国際開発分野で活用）
  - **IDEAL**: （既出）
- 評価軸（必須条件）:
  - 組織的合意形成: 改善スポンサー／ステークホルダーの巻き込み手順が明示されているか
  - 測定指標の柔軟性: 必要な指標を自由に設定・測定できるか
  - 文化・ケイパビリティ: 組織文化・スキルの診断と開発を扱えるか
  - 継続的改善: 計測から改善までをループさせる仕組みが組み込まれているか

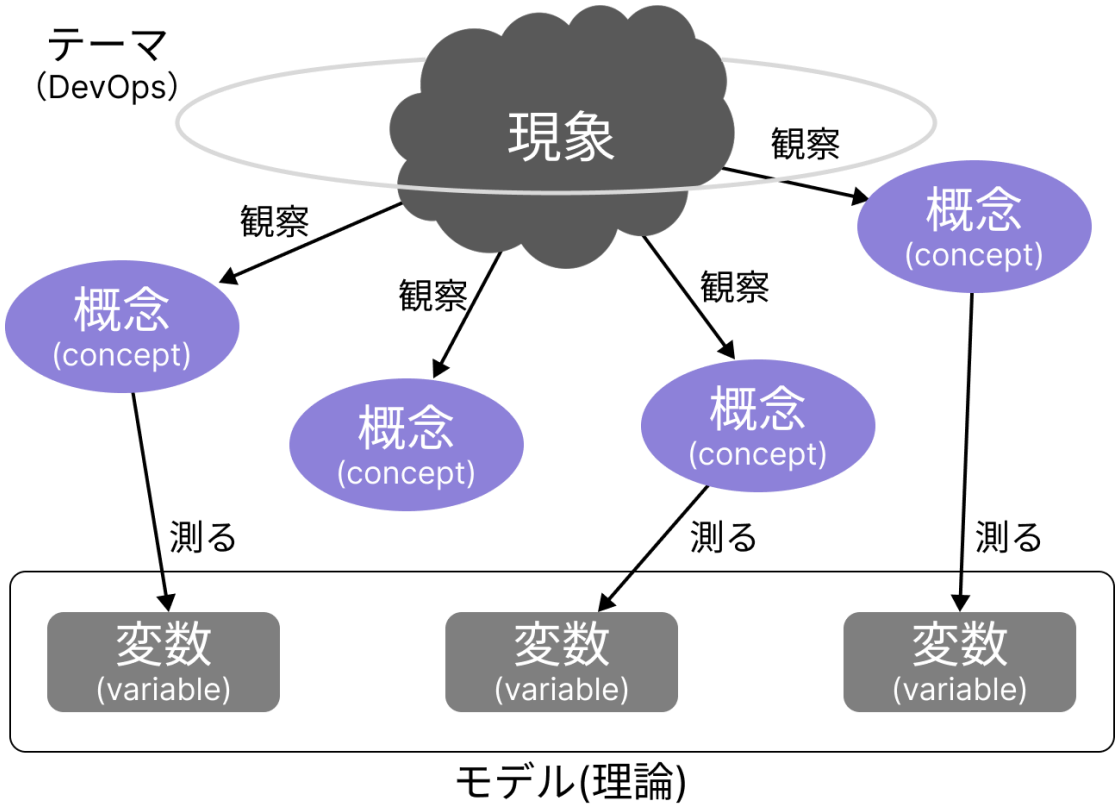
評価軸	求める状態	PDCA	Six Sigma	PCM	IDEAL
組織的合意形成	スポンサーと推進体制を明示	△ スポンサー定義なし	△ 改善チャンピオン依存	○ 参加型設計は可能	◎ Initiatingで必須成果物化
測定指標の柔軟性	Four Keys等を自由に設定可能	△ 各自で追加設計が必要	× 品質指標に限定	△ 重視していない	◎ 指標を自由に設定可能
文化・ケイパビリティ	技術×文化を統合診断	× プロセス寄り	△ 品質文化中心	△ 能力開発が弱い	◎ Diagnosingで文化診断を標準化
継続的改善	定量指標でサイクルを回す	△ 計測は任意	◎ DMAICが強力	△ プロジェクト単位で止まる	◎ Learning→Initiatingでループ

他モデルとの比較とIDEAL選定理由

- **主な差分:**
  - **組織的合意**: 他は「個人の頑張り」に依存。IDEALだけがInitiatingで「成果物」として強制する
  - **測定の自由度**: Six Sigmaは品質指標で固定。IDEALはFour Keysでもカスタム指標でも自由に組み込める
  - **文化の扱い**: IDEALのみが技術・プロセス・文化を同列に診断し、改善施策を形式知化できる
  - **持続性**: Six SigmaのDMAICは強力だが品質改善専用。IDEALは組織変革全体をカバーし、Learningで次サイクルへ接続
- **結論**:
  - 必須要件を全て満たすのはIDEALモデルのみ。

科学的リサーチプロセスは次の4段階で進みます

Step	説明
1 現象の観察	「DevOpsの実践は本当にビジネスの成果に繋がるのか？」といった問いからDORAの研究が始まりました。
2 概念の特定	DORAは、継続的デリバリー、継続的インテグレーション、自動化テストの実施などを概念として特定しました。
3 変数への変換	概念を測定可能な形に変換します。例えば、デプロイ頻度、変更リードタイム、障害復旧時間、変更失敗率などの具体的指標として定義します。
4 モデルの構築	収集したデータを統計的に分析し、相関または因果関係を明らかにします。DORAの研究では、自動化テストの実施が変更リードタイムを短縮することや、チームの独立性がサービスの信頼性向上につながることを実証しました。

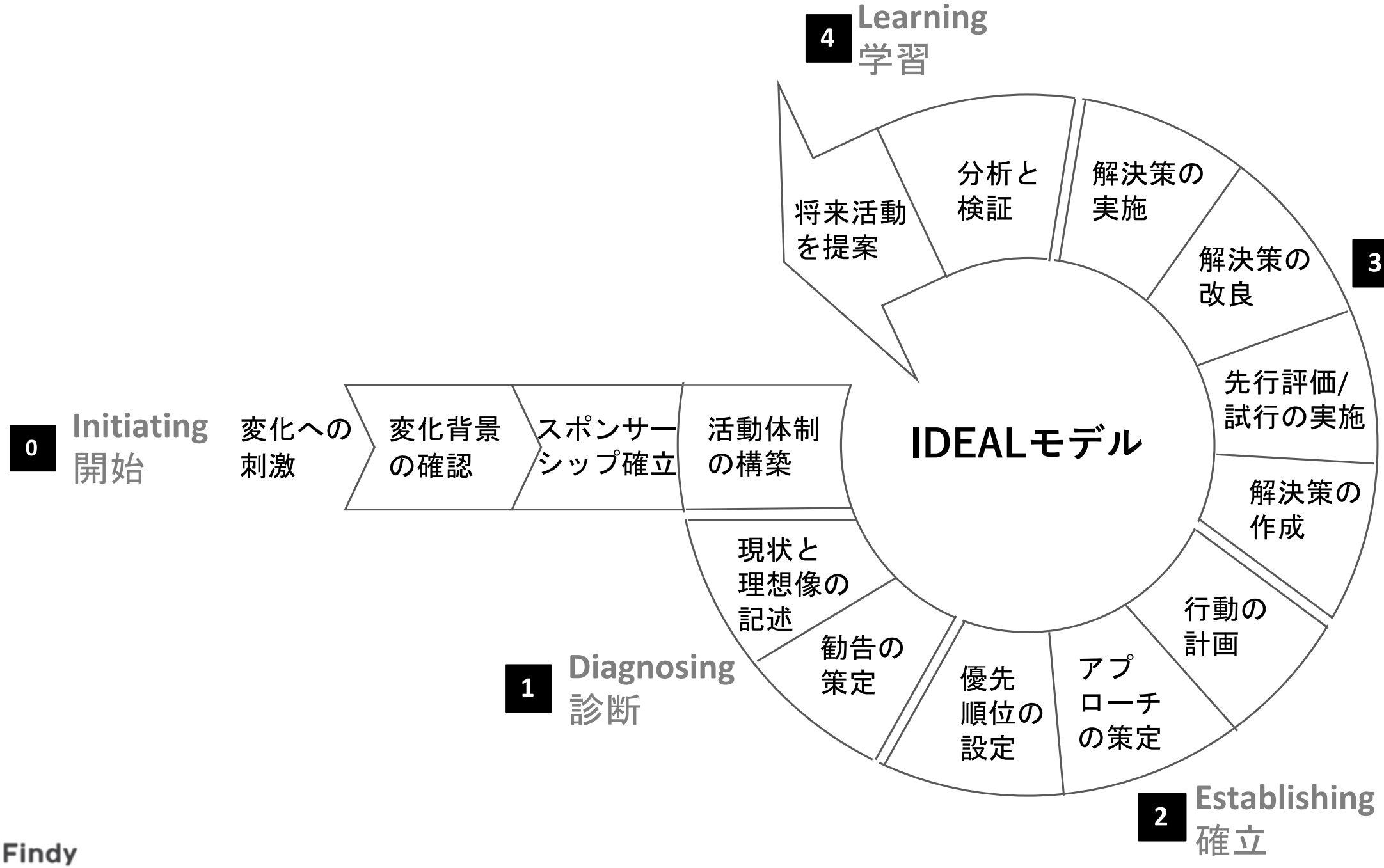


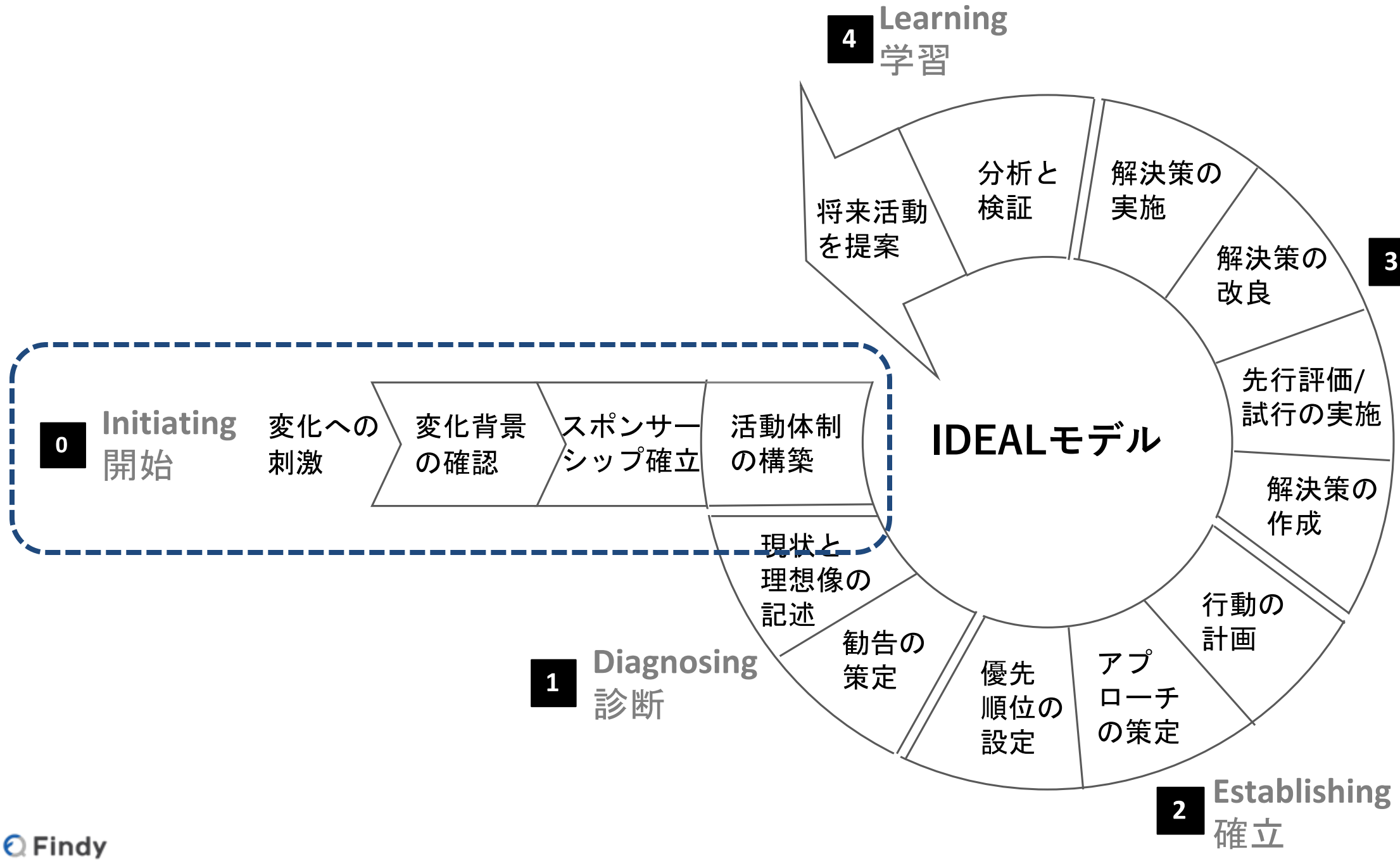
IDEALは科学的リサーチと親和性が高い

向後, 千春. (2016). 18歳からの「大人の学び」基礎講座: 学ぶ, 書く, リサーチする, 生きる. 北大路書房. 図3-3に筆者が加筆

Findy

実践方法 - 5段階の詳細





開始（Initiating）の特徴

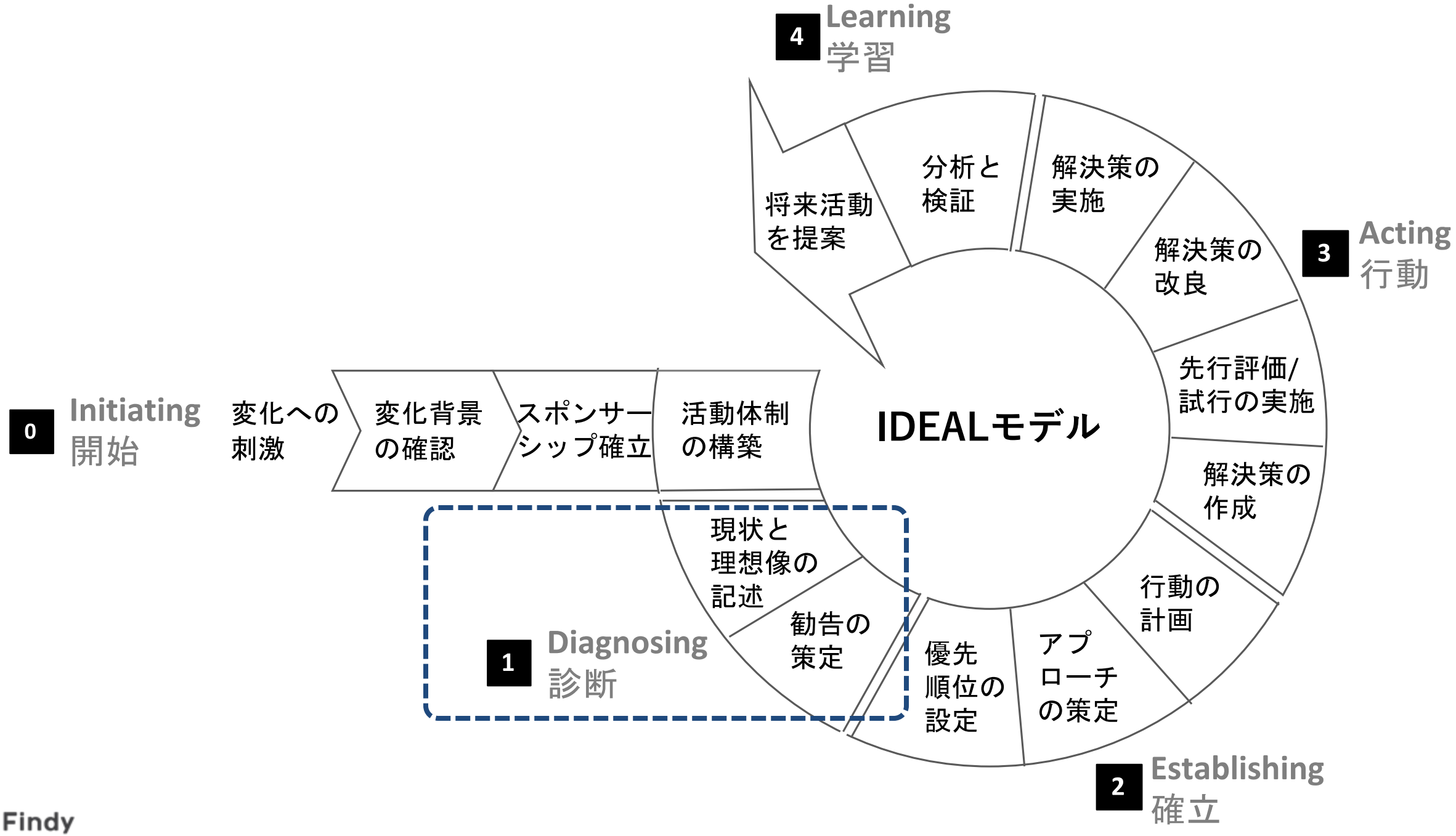
- **開始（Initiating）とは**
  - IDEALモデルの第1段階：改善活動の土台作り
  - 目的：組織的な支援体制を確立し、改善を成功に導く
- **開始（Initiating）で実施すること**
  - スポンサーの明確化（経営層の誰が支援するか）
  - ステークホルダーの特定と巻き込み戦略
  - 改善活動の優先度づけと組織アジェンダへの位置づけ

  → これらを成果物として明文化し、合意形成する

開始（Initiating）の特徴

- **開始（Initiating）とは**
  - IDEALモデルの第1段階：改善活動の土台作り
  - 目的：組織的な支援体制を確立し、改善を成功に導く
- **開始（Initiating）で実施すること**
  - スポンサーの明確化（経営層の誰が支援するか）
  - ステークホルダーの特定と巻き込み戦略
  - 改善活動の優先度づけと組織アジェンダへの位置づけ

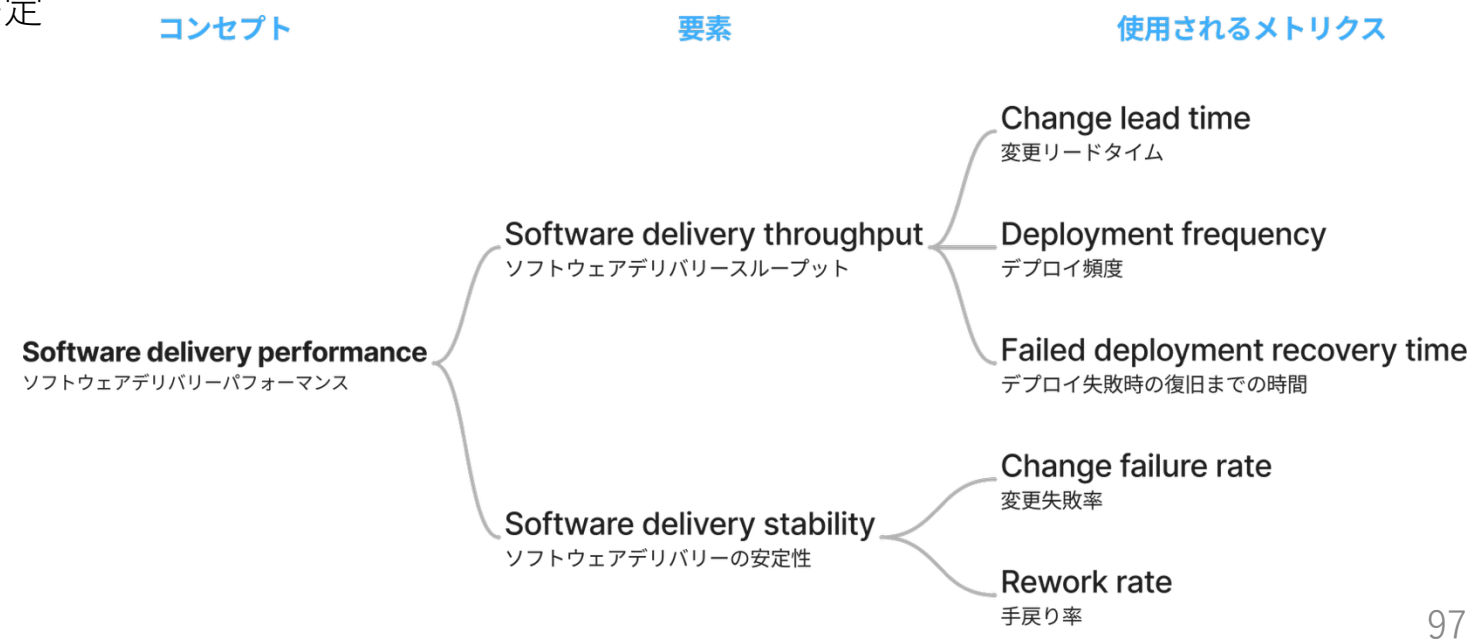
    → これらを成果物として明文化し、合意形成する

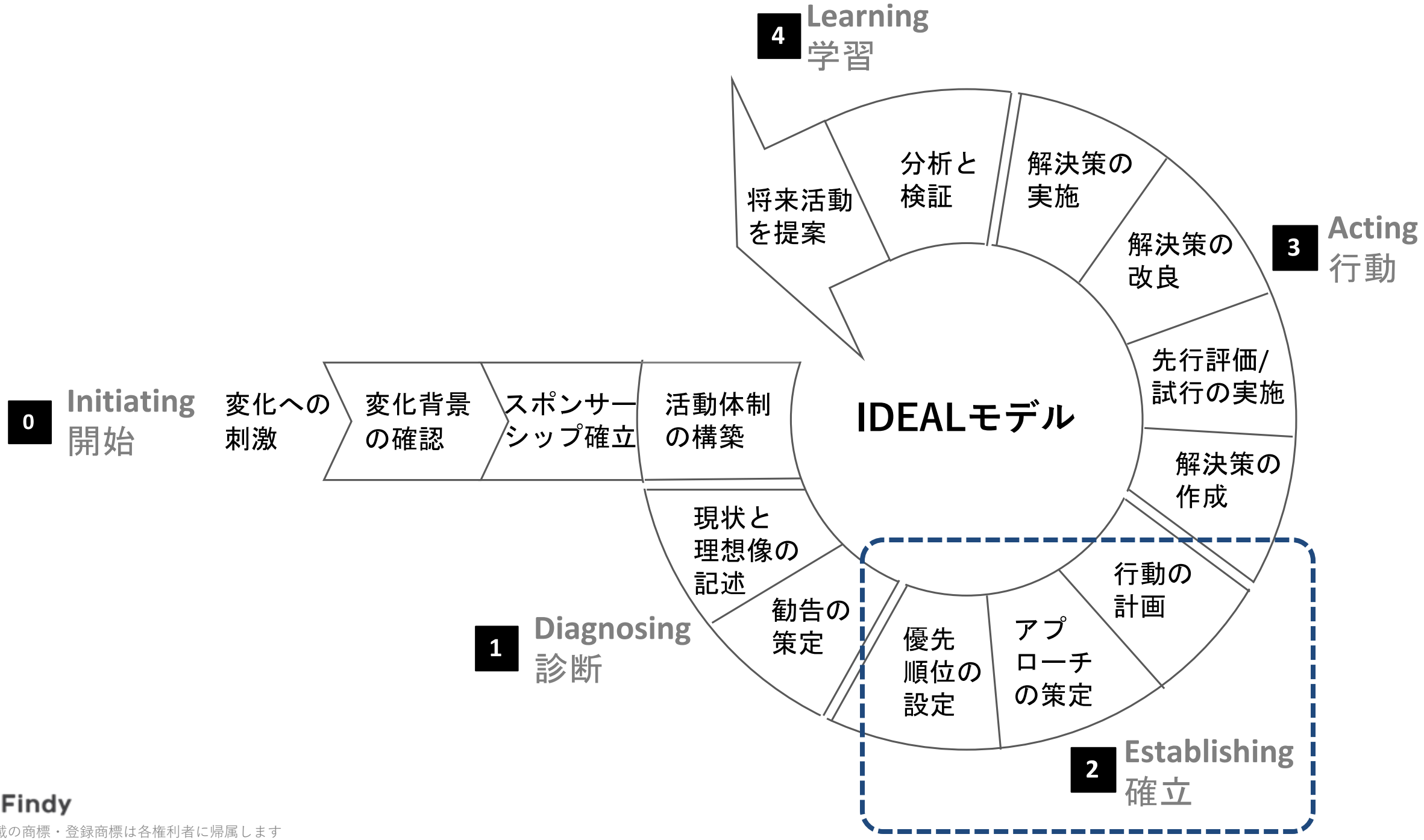


診断（Diagnosing）の特徴

- **診断（Diagnosing）とは**
  - IDEALモデルの第2段階：組織の現在地を正確に把握
  - 目的：改善すべき領域を多面的に特定する
- **診断（Diagnosing）で実施すること**
  - 計量的診断：DORA Metrics 測定による定量評価
  - 定性的診断：ケイパビリティアセスメント（技術・文化・組織）

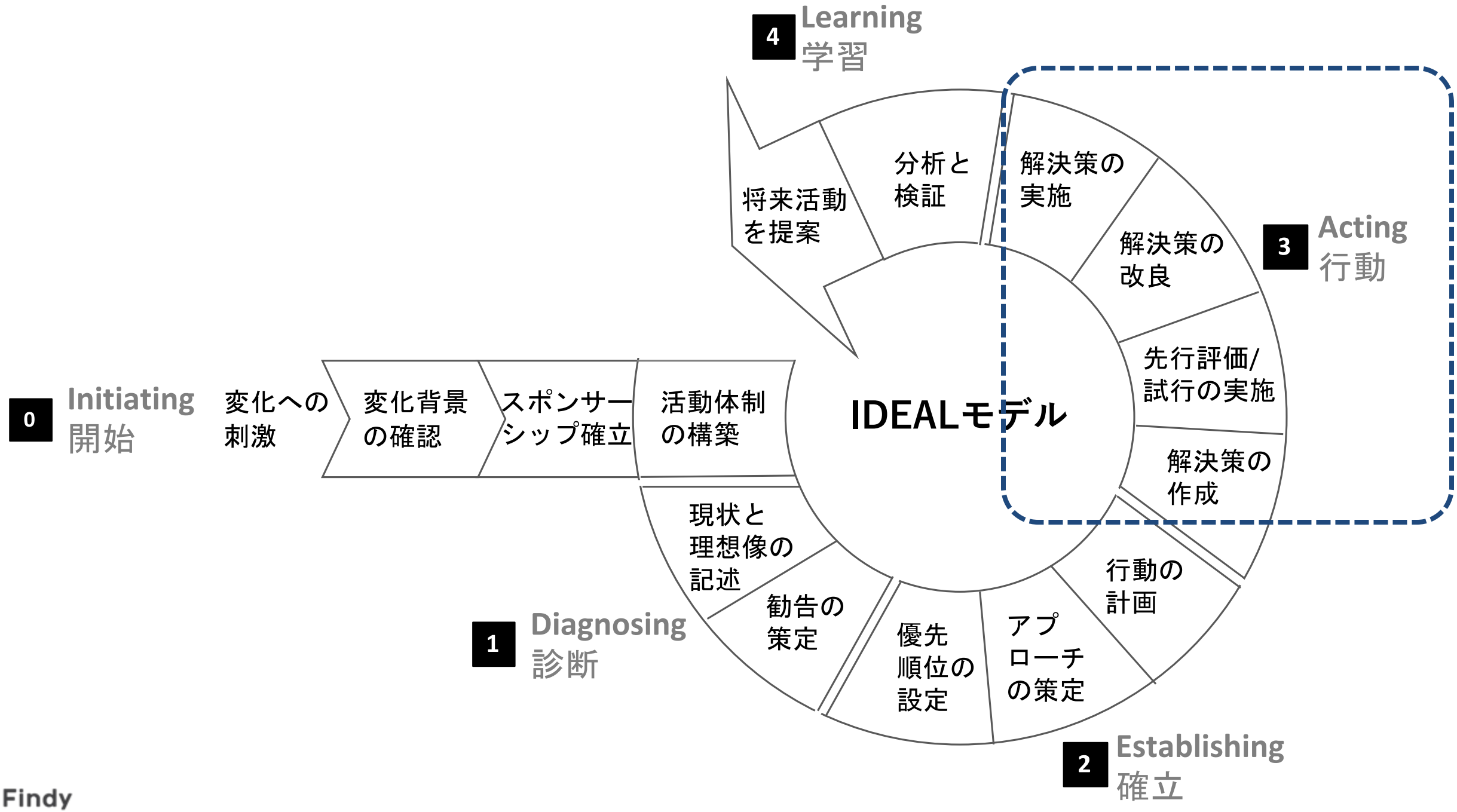
    → これがSix SigmaやPDCAとの明確な違い
  - 多面的診断により改善の優先順位を科学的に決定





確立（Establishing）の特徴

- 確立（Establishing）とは
  - IDEALモデルの第3段階：改善の具体的な計画と目標を設定
  - 目的：診断結果に基づき実行可能なロードマップを確立する
- 確立（Establishing）で実施すること
  - 優先改善領域の特定（技術・文化・組織の統合計画）
  - 測定計画の策定とツールチェーン設計（CI/CD等の自動化基盤が前提）

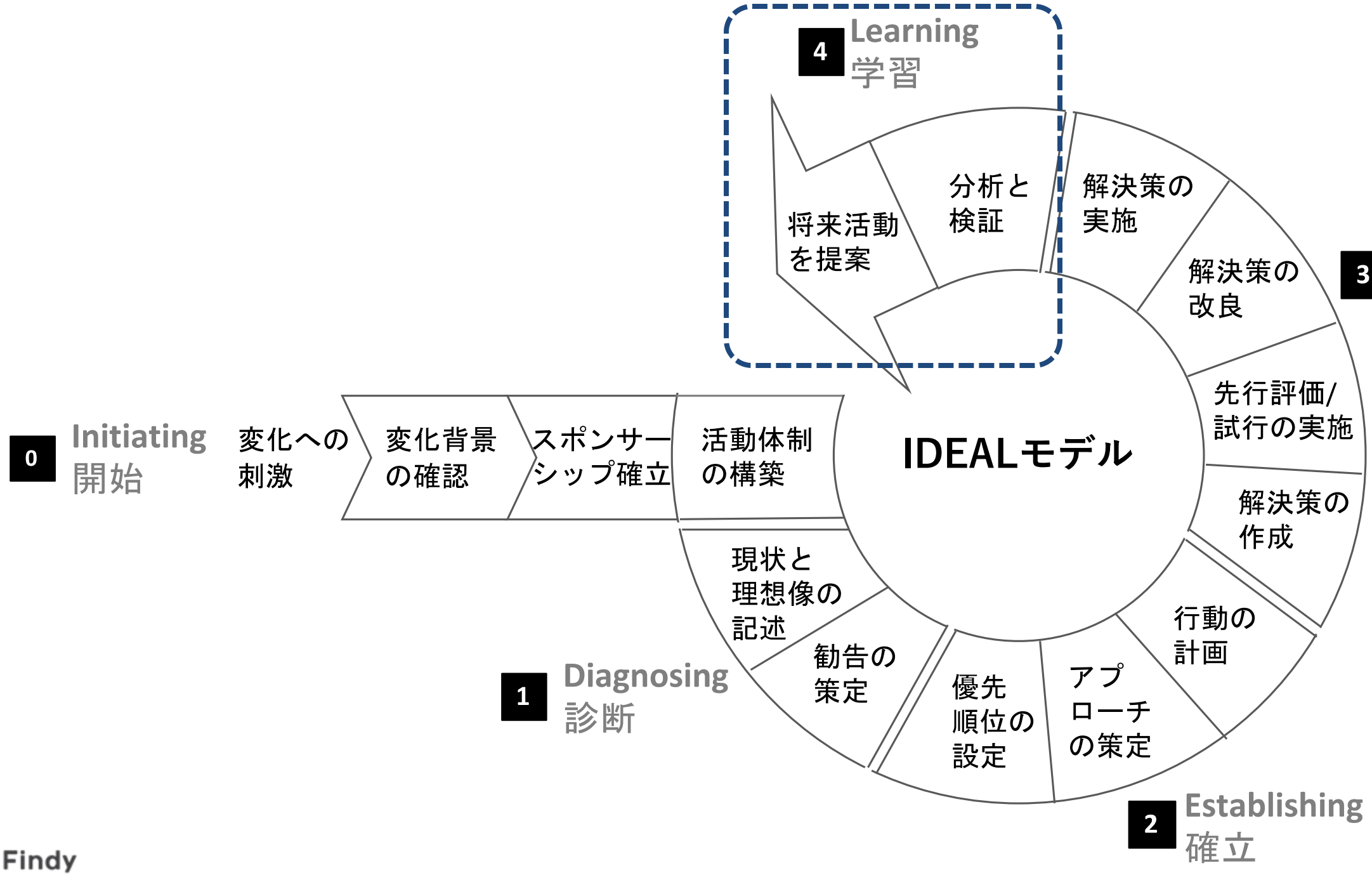


行動（Acting）の特徴

- 行動（Acting）とは
  - IDEALモデルの第4段階：改善計画を実行し成果を測定する
  - 目的：計画に基づく改善を実施し、Four Keysで継続的に効果を検証する
- 行動（Acting）で実施すること
  - イテレーションごとの振り返りと改善実施
  - Four Keysによる継続測定
- 当社の改善実績
  - サイクルタイム改善[7][8]:
    - 平均サイクルタイム: 最初のコミットからマージまで3.6時間
    - プルリクエストレビュー時間: 最短1分以内、目標10分程度
    - テストカバレッジ: 90%以上（自然に達成）
  - Four Keys改善（デプロイ頻度向上）[10]:
    - リリースプロセスの自動化により当日デプロイが標準に
    - GitHub Actionsによる6ステップ自動化（ワークフロー起動→自動PR生成→マージ→本番デプロイ）

行動（Acting）の実践 - 小さなバッチサイズの徹底

- リーンの基本法則: 変更バッチサイズを最小化
- プルリクエスト粒度の原則[8]:
  - 「一つのことだけに注力」した変更
  - 「10のPRを1回レビュー」より「1のPRを10回レビュー」
  - 迷ったら小さく作成し、後から追加
- フィーチャーフラグ（Feature Flag）運用
- CI/CD高速化の実践[7]:
  - パッケージキャッシング
  - Nxによる変更検知
  - 最大40並列テスト実行
  - 高性能なCI実行環境の活用



学習（Learning）の特徴

- 学習（Learning）とは
  - IDEALモデルの第5段階：改善結果を分析し次のサイクルへ活かす
  - 目的：知見を形式知化し、継続的改善サイクルを回す
- 学習（Learning）で実施すること
  - 改善結果の分析（Four Keys × 事業KPIの相関分析）
  - ROI（Return on Investment）可視化による経営層への説明責任
  - 知見のテンプレート化と組織資産化
  - 次のInitiatingサイクルへのフィードバック
- 重要性
  - この段階が機能しないと単発の改善で終わってしまう
  - 継続的な測定と分析により、次のサイクルでの改善精度が向上する

学習（Learning）の実践 - 文化・組織改善の測定

- SPACEフレームワークによる開発者エクスペリエンスの測定[11]
  - SPACEの5次元: Satisfaction（満足度）、Performance（パフォーマンス）、Activity（活動量）、Communication（協働）、Efficiency（効率）
  - 測定: 月次開発者サーベイ
  - 分析: Four Keysとの相関
    - 例: Satisfaction and well-beingスコア↑ → 変更失敗率↓
  - 施策例: 失敗を学習機会として扱う制度化
  - 成果: Satisfactionスコア 3.2 → 4.1（6ヶ月）
- デリゲーションポーカーによる意思決定権限の明確化
  - 手法: Management 3.0のデリゲーションポーカー
  - 7レベルの委譲度: Tell（命令）→ Sell（説得）→ Consult（相談）→ Agree（合意）→ Advise（助言）→ Inquire（問い合わせ）→ Delegate（委譲）

学習（Learning）の実践 - 人材育成とナレッジ共有の測定

- 教育メソッドの体系化[9]
  - 原則:
    - 「目の前のことに集中」
    - 「3段階学習（基礎→手数→質と理解）」
    - 「正しい方法を正しい手順で作業」
  - 測定: PR作成数 4個/日→9個/日（3ヶ月）
  - テンプレート化: Learningで蓄積、次のInitiatingで再利用

IDEALを筋トレに例えると

0
開始フェーズ

強い開発チームになるための土台作り
- スポンサーを明確にして、自分たちを**応援してくれる体制**を組織的に構築する

1
診断フェーズ

開発チームの体力や筋力のレベルを評価する
- ツールの自動化で開発パフォーマンス（Four Kyes）や重要指標を確認。ケイパビリティ・データを対比させ改善が必要な箇所を把握する。

2
確立フェーズ

開発チームの筋トレ計画を立てる
- 診断フェーズで得られた結果に基づいて、プロセス改善活動の優先順位と取り組み方を策定し、具体的な改善計画を作成する。改善計画は専門家が支援。

3
行動フェーズ

実際に筋トレを開始する
- 確立フェーズで作成された改善計画に従って解決策を作り、その先行評価・試行・パッケージ化・展開・長期支援移行を実施する。

4
学習フェーズ

効果を評価し、必要に応じて計画を調整する
- 行動フェーズの結果を受けて、これまでの活動を分析してその妥当性を確認し、次のサイクルの準備を行う。
- 診断フェーズに戻る

Findy

まとめ

まとめ - 場当たり的改善からの脱却

- 問題の本質
  - 開発生産性指標25個が乱立、体系的フレームワークの欠如
  - 改善への意欲はあるが、実行と成果に結びついていない
  - AI時代の到来で、科学的な測定基盤の重要性がさらに増大
- 解決策
  - IDEAL: 組織的合意形成から継続的改善まで、プロセス改善の体系的フレームワーク
  - Four Keys: スピードと安定性を両立する科学的測定指標
  - 統合の威力: IDEALの5段階にFour Keysを組み込むことで、測定に基づく持続可能な改善サイクルを実現
- 実践で得られた成果
  - サイクルタイム3.6時間、当日デプロイが標準、PRレビュー最短1分
  - 技術指標だけでなく、文化・組織（SPACE、デリゲーションポーカー、教育メソッド）も測定・改善可能
- 持ち帰っていただきたいこと
  - 改善は「何を測るか」だけでなく「どう進めるか」が重要。IDEALモデル×Four Keysは、場当たり的改善から脱却し、組織に根付く改善文化を築く実践的アプローチ。

参考文献

[1] Robert McFeeley. 1996. IDEAL: A User's Guide for Software Process Improvement. Technical Report CMU/SEI-96-HB-001. Carnegie Mellon University, Software Engineering Institute.

[2] Nicole Forsgren, Jez Humble, and Gene Kim. 2018. Accelerate: The Science of Lean Software and DevOps. IT Revolution Press.

[3] Google Cloud. 2024. 2024 State of DevOps Report. https://cloud.google.com/devops/state-of-devops

[4] Abi Noda, Margaret-Anne Storey, Nicole Forsgren, and Michaela Greiler. 2023. DevEx: What Actually Drives Productivity. ACM Queue 21, 2 (2023).

[5] EuroSPI. SPI Manifesto. https://conference.eurospi.net/index.php/en/manifesto

[6] ファインディ株式会社. 2025. ソフトウェア開発における『開発生産性』に関する実態調査. https://findy.co.jp/3036/

[7] 戸田. 2024. Findyの爆速開発を支えるテクニック. Findy Tech Blog. https://tech.findy.co.jp/entry/2024/05/27/090000

[8] 戸田. 2024. Findyの爆速開発を支えるPull requestの粒度. Findy Tech Blog. https://tech.findy.co.jp/entry/2024/10/04/070000

[9] 戸田. 2025. Findyの爆速成長を支えるエンジニア教育メソッド. Findy Tech Blog. https://tech.findy.co.jp/entry/2025/02/18/070000

[10] 新福. 2024. ファインディでのGitHub Actions自動化の事例. Findy Tech Blog. https://tech.findy.co.jp/entry/2024/09/24/090000

[11] Nicole Forsgren, Margaret-Anne Storey, Chandra Maddila, Thomas Zimmermann, Brian Houck, and Jenna Butler. 2021. The SPACE of Developer Productivity. ACM Queue 19, 1 (2021), 20-48. https://queue.acm.org/detail.cfm?id=3454124