Google Cloud

Google の研究結果から考える 内製化のための Tips

Google Cloud
アプリケーション モダナイゼーション スペシャリスト
塚越 啓介

Table of Contents

01

良いドキュメントは
生産性を向上させる

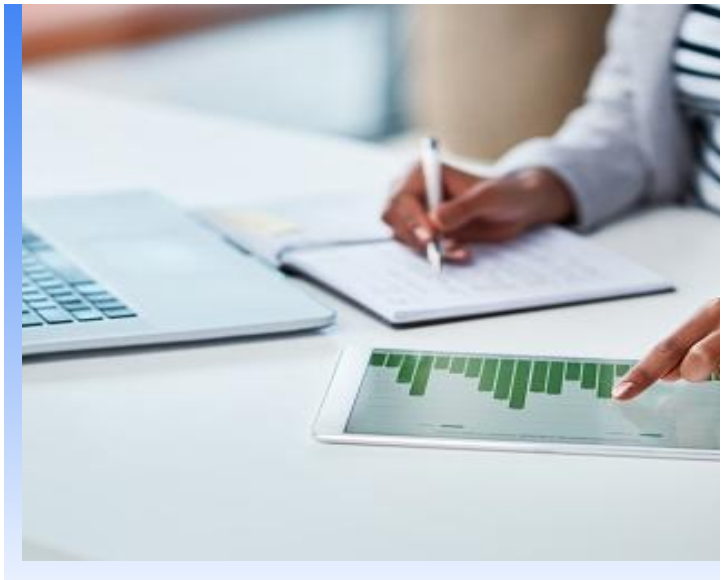
“

今回初めて、チーム内のドキュメンテーションの品質と、それが他の機能やプラクティスに及ぼす影響を測定しました。ドキュメンテーションは、DevOps 機能実装の成功の基礎となることがわかりました。
質の高いドキュメンテーションを行うチームは、セキュリティのベストプラクティスを実践する確率がそうでないチームより 3.8 倍高くなっており、クラウドを最大限に活用する確率も 2.5 倍高くなっていました。"

出典: Google Cloud Blog "2021 年の Accelerate State of DevOps Report で、燃え尽き症候群とチームのパフォーマンスについて発表
https://cloud.google.com/blog/ja/products/devops-sre/announcing-dora-2021-accelerate-state-of-devops-report

質の高いドキュメントとは

質の高いドキュメントを作成しているチームは開発効率がそうでないチームと比べて 2.4 倍も高い。
質の高いドキュメントはこれら全てに当てはまる。



- **読み手の目的を達成できる**
- **正確かつ最新である**
- **シンプルで読みやすい**
- **見つけやすく整理されている**

出典: The 2021Accelerate State of DevOps
https://services.google.com/fh/files/misc/state-of-devops-2021.pdf

質が低いドキュメントのデメリット

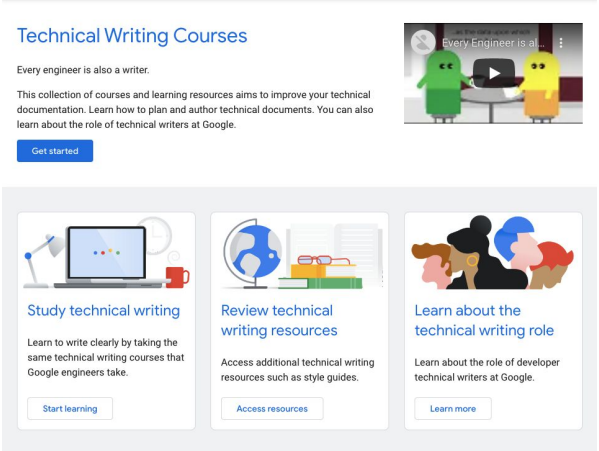
- **読み手**を想定していない文章は読みづらい
  - 引き継ぎ資料にシステムのユースケースやターゲットユーザーがなかったら・・・
- **古いドキュメント**は信頼性が低い（システムの信頼性も低下）
  - ドキュメントの最終更新日が 3 年前のだったら・・・
- **大量の複雑**なドキュメントは読む人の気力をうばい、混乱させる
  - 2000 ページの仕様書を渡されたら・・・
- どんなに優れたドキュメントでも **見つけられなければ** 意味がない
  - 社内ドキュメントを検索できなかったら・・・

質の高いドキュメントを作るための Tips

社内検索システム

出典: Google Workspaces "Cloud Search"
https://workspace.google.co.jp/intl/ja/products/cloud-search/

Technical Writing Courses



出典: Technical Writing
https://developers.google.com/tech-writing

Design Doc

- 背景とスコープ
- 目標と非目標
- 実際のデザイン
- 考慮された代替案
- 横断的な懸念事項

出典: Design Docs at Google
https://www.industrialempathy.com/posts/design-docs-at-google/#the-actual-design

02

優れたチームの特徴は
自動テスト

テストの自動化が大きな差

コードのコミットをトリガーにソフトウェアの
ビルドと一連のテストスイートを実行することが必要。
ここが大きな差として現れている。

自動化は実装するにはコストが高すぎるという声があるが、自動化は本当に健全な投資である。

	Low	Medium	High	Elite
Automated build	64%	81%	91%	92%
Automated unit tests	57%	66%	84%	87%
Automated acceptance tests	28%	38%	48%	58%
Automated performance tests	18%	23%	18%	28%
Automated security tests	15%	28%	25%	31%
Automated provisioning and deployment to testing environments	39%	54%	68%	72%
Automated deployment to production	17%	38%	60%	69%
Integration with chatbots / Slack	29%	33%	24%	69%
Integration with production monitoring and observability tools	13%	23%	41%	57%
None of the above	9%	14%	5%	4%

出典: The 2019 Accelerate State of DevOps
https://services.google.com/fh/files/misc/state-of-devops-2019.pdf

自動テストがないとどうなるか

自動テストがないほとんどの場合、テストは「開発完了」後に別のフェーズで行われる。

- 手動による回帰テストは実行するのに時間もコストもかかるため、プロセスのボトルネックとなりやすい
- 人は回帰テストのような繰り返しの作業を得意としないため、間違いが発生しやすい。
- テスト結果のフィードバックが遅くなり、不具合の発生から修正まで時間が開きやすい
- デベロッパーに自分のコードをテストする責任がなければ、テストしやすいコードの作成方法を習得するのは難しい

出典: "DevOps 技術: 継続的テスト"
https://cloud.google.com/architecture/devops/devops-tech-test-automation

テストにデベロッパーを関与させることの重要性

DORA の調査では、デベロッパーが一連の自動テストの作成と保守を主に担当し、デベロッパーが承認テストの失敗を簡単に修正できる場合に、パフォーマンスが向上している。デベロッパーがテストに関与していないと次のような問題が生じる。

- **頻繁にテストスイートが破損した状態になる。** コードを変更するとテストの更新が必要になる。デベロッパーがテスト自動化を担当していなければ、担当のチームがテストを修正するまで、ビルド　パイプラインが破損した状態になる。
- **デベロッパーが、テストしにくいコードを生成している。** デベロッパーが、テスト方法を考慮せずにコードを書きがちになる。その結果、コードの設計が不十分で、コストとメンテナンスの手間がかかるテストスイートになる。

出典: "DevOps 技術: 継続的テスト"
https://cloud.google.com/architecture/devops/devops-tech-test-automation

継続的テストの実装方法

ソフトウェアの品質を高めるには、デリバリー プロセス全体で自動テストと手動テストを継続的に実行し、開発中のシステムの機能とアーキテクチャを検証する必要がある。

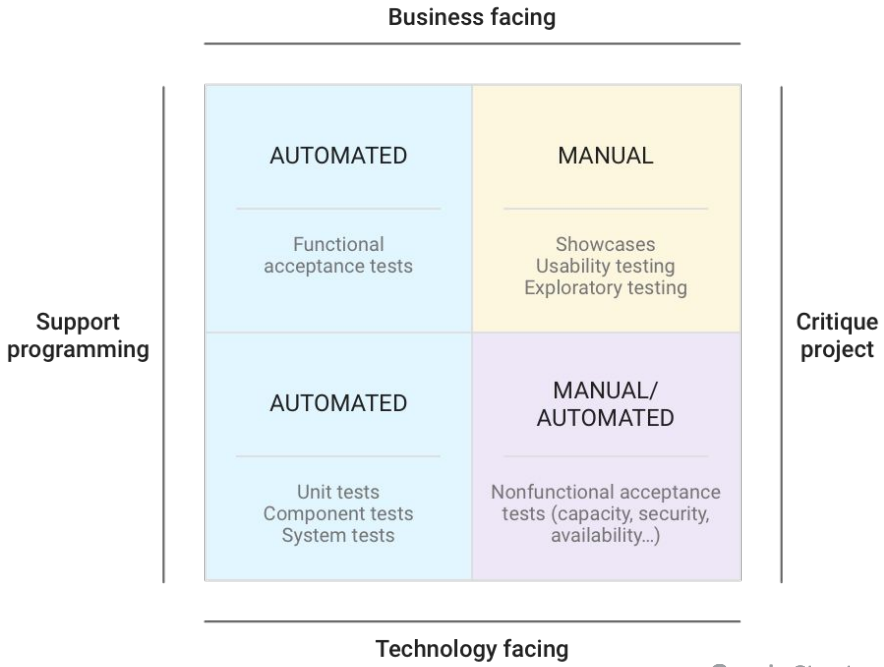
- **単体テスト:** 通常、単体テストでは 1 つのメソッド、クラス、または関数を単独でテストして、デベロッパーに対し、作成したコードが意図どおりに機能することを保証します。テスト可能なコードとメンテナンスしやすいテストにするために、コードを作成する前に単体テストを作成する。 テスト駆動型開発（TDD）と呼ばれる手法。

- **承認テスト:** 通常、実行中のアプリまたはサービス（通常は依存関係は テストダブル に置き換えられます）をテストし、上位レベルの機能が設計通りに機能すること、回帰エラーが発生していないことを保証する。承認テストのサンプルで、ユーザー ストーリーに対するビジネス上の承認基準や API の正確性を確認する。こうしたテストは、開発プロセスの一環として作成し、自動承認テストに合格するまでは、成果物を「開発完了」として宣言することはできない。

出典: "DevOps 技術: 継続的テスト"
https://cloud.google.com/architecture/devops/devops-tech-test-automation

実行する自動テストと手動テストの種類

- 自動テストスイートの目的は、エラーをできるだけ早く見つけること。
- 手動テストの目的は、探索とユーザビリティの確認。



出典: "DevOps 技術: 継続的テスト"
https://cloud.google.com/architecture/devops/devops-tech-test-automation

03

怒られることで
パフォーマンスは低下する

成果を出せるチームと出せないチームの違い

平等な発言

「全員が同じくらい話す機会があれば、チームはうまくいった」

「常に 1 人または小グループのみが発言した場合、集団知能は低下した」

ソーシャル センシティビティ

良いチームの人々は声の調子、表情、その他の非言語的手がかりに基づいて他の人がどのように感じているかを直感するのがうまかった。誰かが動揺したり、置いていかれたりした際のサポートが平均以上だった。

出典: re:Work "「効果的なチームとは何か」を知る
https://rework.withgoogle.com/jp/guides/understanding-team-effectiveness/steps/identify-dynamics-of-effective-teams/

心理的安全性

リスクある行動の結果に対する個人の認知の仕方。

「無知、無能、ネガティブ、邪魔だと思われる可能性のある行動をしても、このチームなら大丈夫だ」と信じられるかどうかを意味する。心理的安全性の高いチームのメンバーはリスクを取ることに不安を感じない。自分の過ちを認めたり、質問をしたり、新しいアイデアをだしても、誰も自分を馬鹿にしたり罰したりしないと信じられる。

re:Work

g.co/reworkjp

出典: re:Work "「効果的なチームとは何か」を知る
https://rework.withgoogle.com/jp/guides/understanding-team-effectiveness/steps/identify-dynamics-of-effective-teams/

チームメンバーの状態を確認する

- ミスをするとたいてい非難される。
- チームで課題や難しい問題を指摘し合えない
- 自分と異なるということを理由に他者を拒絶することがある。
- リスクのある行動をしたら怒られる可能性がある
- 他のメンバーに助けを求めることは難しい。
- チームメンバーと仕事をするとき、自分のスキルと才能が尊重され、活かされていると感じる。

出典: re:Work "「効果的なチームとは何か」を知る
https://rework.withgoogle.com/jp/guides/understanding-team-effectiveness/steps/identify-dynamics-of-effective-teams/

心理的安全性を高める行動

- 積極的な姿勢を示す
- 理解していることを示す
- 対人関係において相手を受け入れる姿勢を示す
- 意思決定において相手を受け入れる姿勢を示す
- 強情にならない範囲で自信や信念を持つ

出典: re:Work "心理的安全性を高めるためにマネージャーにできること
https://docs.google.com/document/d/1M2HvwvzFXfnNJkt-r9s4XXQtNI8YeVHfEpoICwzNIPo/edit

04

まとめ

内製化で開発パフォーマンスをあげるための Tips

- 良いドキュメントは生産性を向上させる
- 優れたチームの特徴は自動テスト
- 怒られることでパフォーマンスは低下する

ほかにも開発パフォーマンスを向上するためのさまざまな Tips があります

- **技術に関する能力**

  クラウドアーキテクチャ、バージョン管理、継続的インテグレーション、継続的デリバリー、トランクベース開発、テスト自動化、疎結合アーキテクチャ、コードの保守性、セキュリティ、データベースの変更管理

- **プロセスに関する能力**

  小さいバッチ単位の作業、変更承認の効率化、モニタリング、Work in Progress（WIP）の制限、作業の可視化、機能別アウトソーシングの抑制

- **文化に関する能力**

  Westrum 型の組織、学習文化、自律的であること、振り返りの重視

DevOps の能力 | Google Cloud
https://cloud.google.com/architecture/devops/capabilities?hl=ja

より詳細な情報が知りたい場合は

DORA

re:Work

re:Work

practices, research, and ideas to #makeworkbetter

g.co/rework

Tech Acceleration Program

アプリケーション開発支援プログラム

Prototype

開発予定アプリケーションの
プロトタイプ開発の
ご支援

Architecture

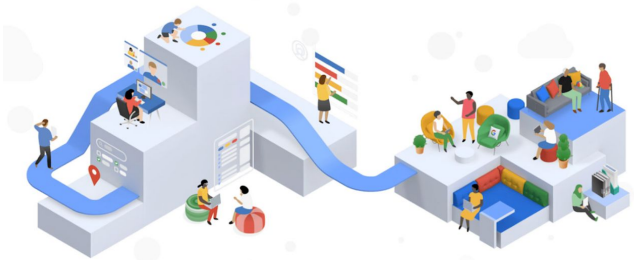
開発予定アプリケーションの
最適なアーキテクチャ設計の
ご支援

Security

クラウド開発における
セキュリティルール策定の
ご支援

内製化支援パートナー

内製化支援で DX を加速



ビジネスアセスメント	組織・体制作り支援	テーマ選定	技術アセスメント	開発支援	活用・運用支援
課題発掘から開発プロセスへの繋がりをコンサル	アジャイルな開発組織構築支援	どのようなユニットで何を開発するのか	ユーザー側の技術スキルレベルのアセスメント	PoC、Pilot 開発 DevOps との連携支援 ex) GitLab	定期的な活用支援

	ビジネスアセスメント	組織・体制作り支援	テーマ選定	技術アセスメント	開発支援	活用・運用支援
iret	○	○	○	○	○	○
CTC	○	○	○	○		
UL Systems, Inc.	○	○	○			
Cloud Ace	○	○	○	○	○	○
CREATIONLINE, INC.		○			○	
GI Cloud		○			○	○
Sreake					○	○
Centillion System		○		○	○	
DIGITAL VORN			○	○	○	○

Google Cloud パートナーが提供する内製化支援サービスにご興味のある方はこちらまで
https://goo.gle/naiseika-partners