Self-introduction / 自己紹介

- Kazuaki Matsuo / 松尾和昭
  - Twitter: @Kazu_cocoa, GitHub: KazuCocoa

- Cookpad Inc / クックパッド株式会社
- Software Engineer in Quality / テストエンジニア
  - Close to Software Engineer, Tools and Infrastructure
- International business development group / 海外事業部

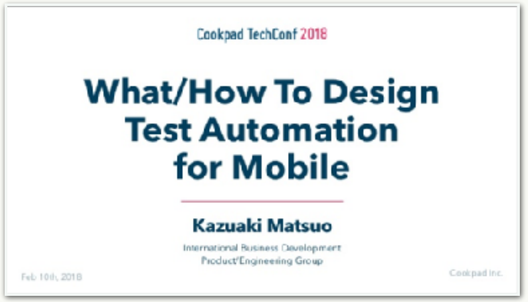
Self-introduction / 自己紹介

- Responsibility Area / 責任範囲
  - Automation for mobile (so far) / (今は)モバイル自動化
  - building cross functional workflow / 横断的なワークフロー構築
  - Hiring, Architectural stuff for Microservices

Background / 背景

- OSS
  - One of Technical Core Committees of Appium

- Presentations / 発表
  - What/How To Design Test Automation for Mobile (at CookpadTechConf2018)
  - Tasting tests at Cookpad (at try!SwiftTokyo 2017)
    - etc

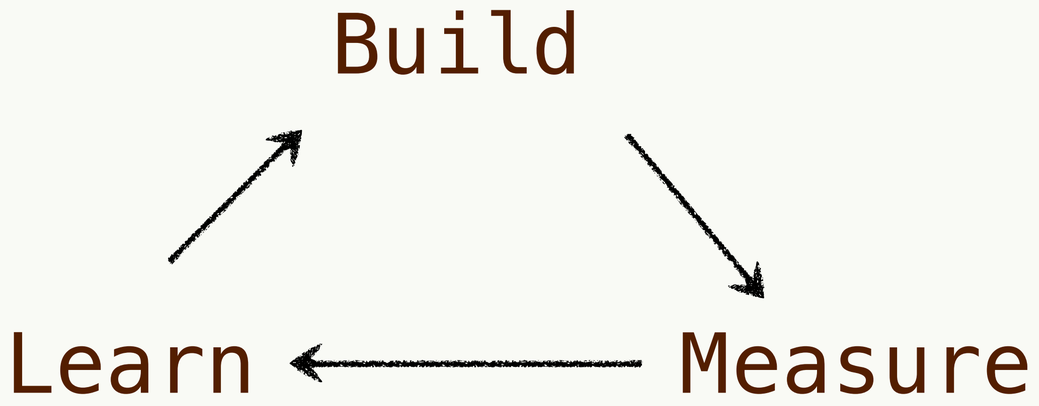


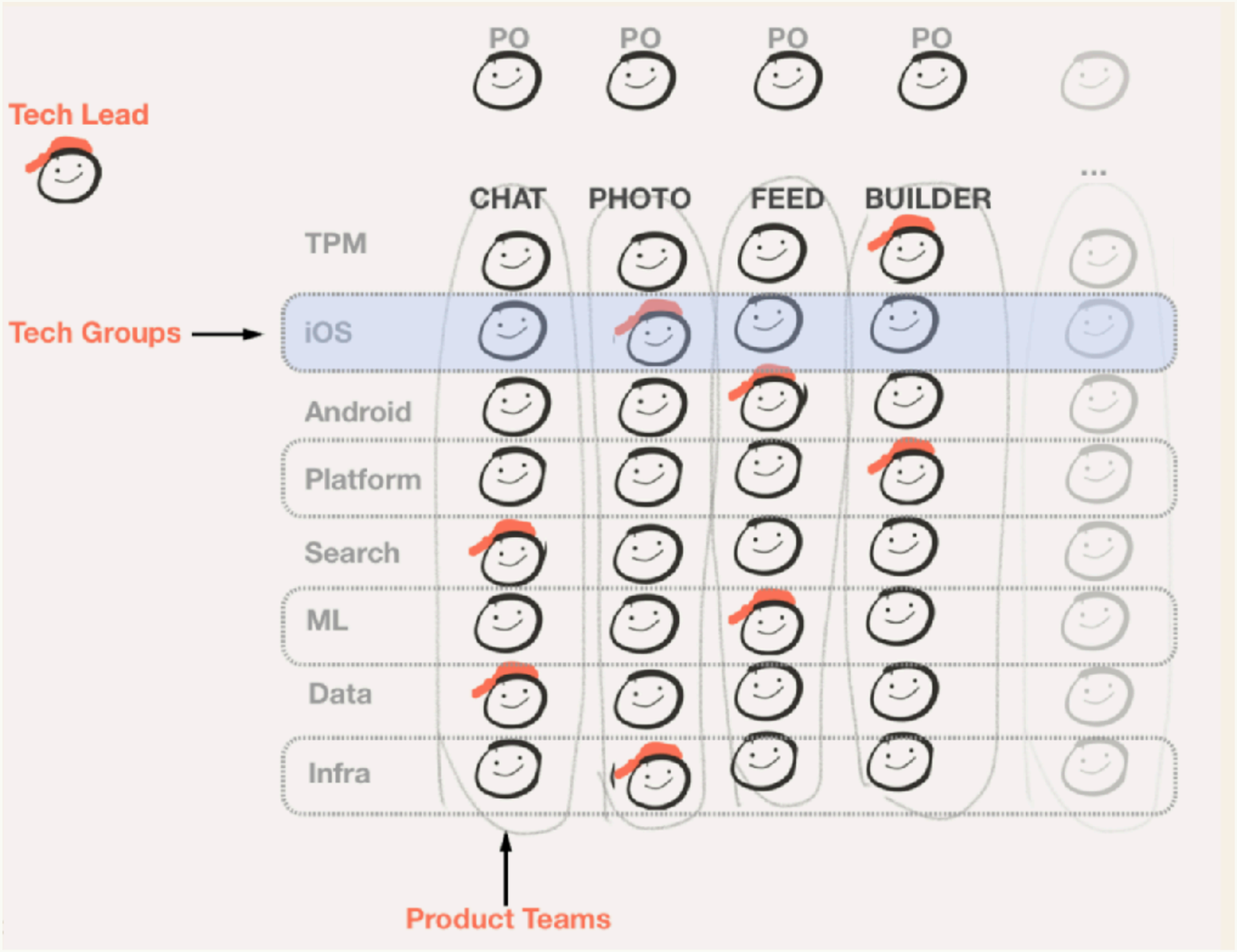
Background / 背景

- Books (as a reviewer), articles / 書籍(レビュー)
  - 初めての自動テスト (Origin: The Way of the Web Tester)
  - モバイルアプリのUIテストフレームワーク「Appium」の最新事情と、Appium Desktopを使ったテストの実行
    - https://codezine.jp/article/detail/10545
  - 開発現場で役立つテスト「超」実践講座
    - https://thinkit.co.jp/series/7223

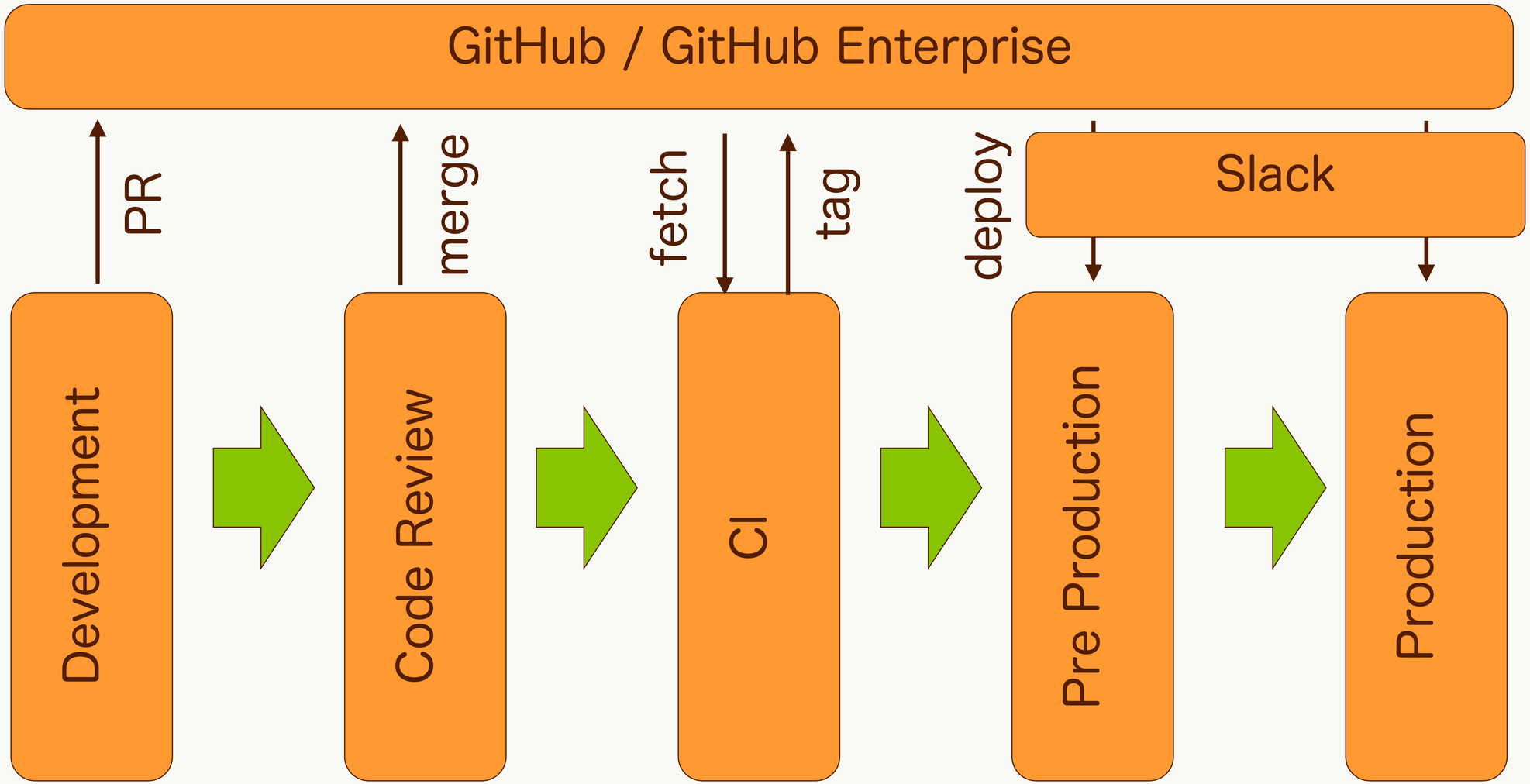
Our Development Style / 開発スタイル

- Service/Feature / サービス/機能
- Expert based / 専門性ベース





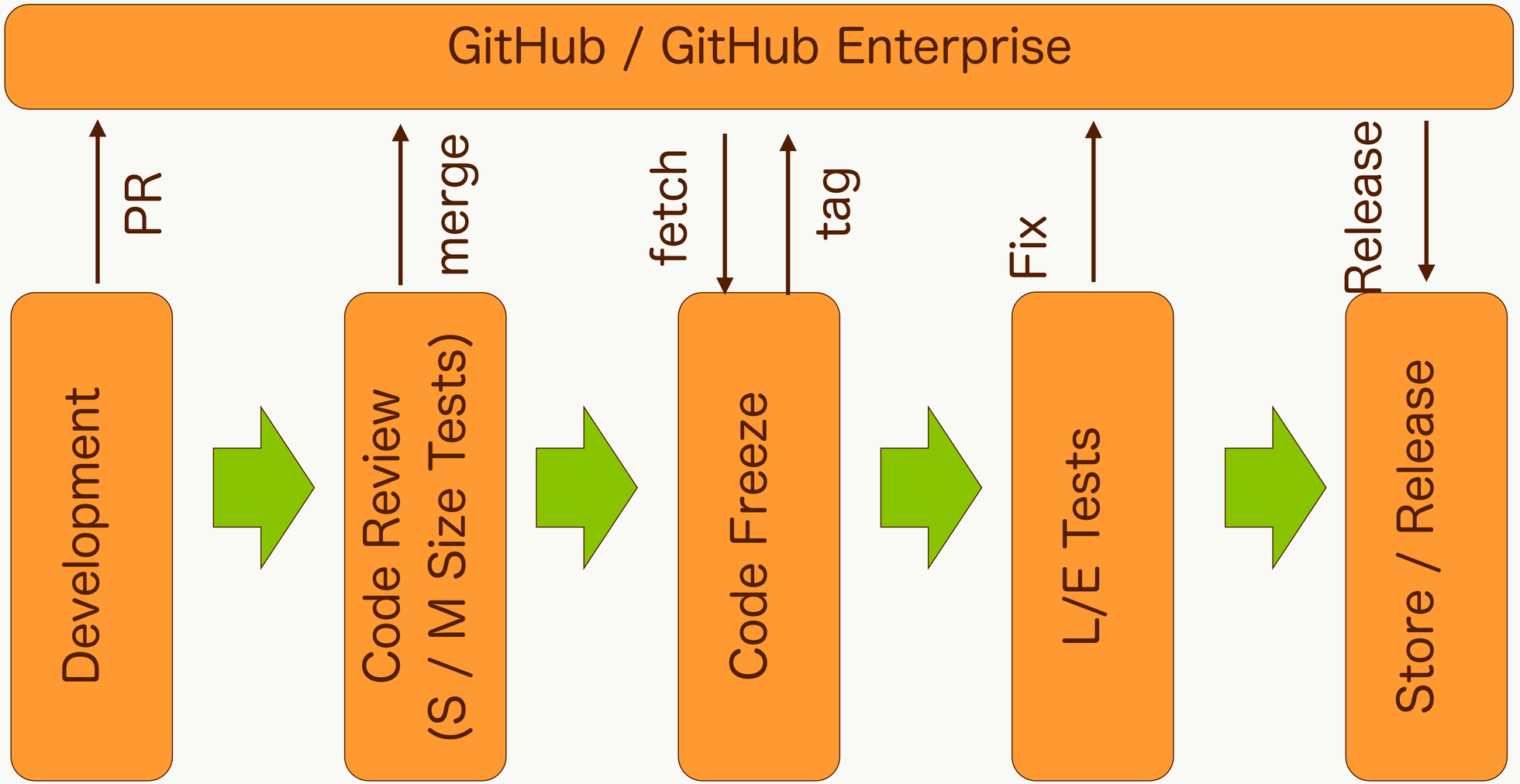
Deployment pipeline - Web



Deployment : Web

- Frequency : In past 3 months / 過去3ヶ月を対象としたリリース頻度
  - 30 times / day (Japan), 20- times / day (International)
    - Production for internal or external release / 社内外への本番環境
    - Include new features and bug fixes / 不具合修正、機能追加など諸々含む
    - Employees who have a title “Engineer” can deploy / “エンジニア”の肩書きを持つ人であれば誰でもデプロイ可能
      - CI should be green / 自動化されたテストが実行されており、それがGreenであることが必須条件
    - No Canary Release (Will implement soon) / カナリアリリースはまだ実装していない（する予定）
- Rollback / ロールバック
  - 10 times in the past 3 months / 3ヶ月のうちには10回程度(国内外合計、新規プロダクト含む)

Deployment pipeline - Mobile



Release : Mobile

- Publish to the store / ストアへの配信
  - Per a week ~ Per a month (Depends on each teams) / 週1 ~ 月1(チームなどに依存するが、大体はこの範囲内)
  - Planned test suite should finish / 計画されたテストが完了している
    - Any fixes bugs which may impact for users / ユーザに影響のある不具合があれば修正されていること
  - Members who is associated with the release gather and make sure the release version / リリース前に特定の単位のメンバが集まり、アプリの体験の統一性などを確認したりもする
- Crashes / クラッシュ頻度
  - 99+% are crash free for almost apps at least/ 大体のプラットフォームで99%以上安定している
    - By Crashlytics

問1: テストの価値を組織においてどのように定義・計測していますか？

- Automated Tests / 自動化されたテストにおいては
  - Reduce regressions / リグレッションの防止という価値観は共有されている
    - No concrete criteria based on coverages (It depends on team) / カバレッジの厳密な目標値などの設定はしていない（チームに依存）
    - Monitoring stability of CI / CIの状態は計測できるようにはなっている
- Manual Tests / 手動で行われるテストにおいては
  - It depends on each team especially prototype, usability testing, etc / ユーザインタビュー形式のプロトタイプ、ユーザビリティテストなど、サービスの価値計測のためのものは各々のチームが必要に応じて行っている
  - No standard / 統一された規格は社内では持たない
  - Share each team’s knowledge to other teams / 知見は社内のチーム間で共有され、実施されているところもある
- Negative feedback from users, the state on the app store / ユーザからのネガティブフィードバックの量、ストアのレビューは計測している
  - We use them to evaluate our service or product, but we don’t use them only for “test” itself. / サービスや製品自体の評価には使っているが、”テスト”単体の評価には利用していない

問2: 継続的テストが生産性を向上した具体例を教えてください

- Deploy some code changes in a few minutes by automation
  - “生産性”を何らかのコード変更をリリース用ブランチにマージしてから世にリリースするまでの時間という意味だと、最短で数分程度で可能になった
  - Not only automated test, but also deployments stuff
- Detect crashes which happen rarely in development, but it impacts users in production
  - 普段の開発で触れにくいが、モバイルアプリだとPush通知によって画面を開くとクラッシュする、という類のものをリリース前に検出でたなどの実績もあり、ユーザの体験を大きく損なうようなリグレッションを予防できている
  - 開発者が”ここが影響するだろう”と予測、静的解析ツールで補助できるところの外にある問題も検出できる頻度が上がった
- Reduce a time developers should check manually
  - 開発者が手動で確認する必要のある範囲、自分以外の人が実装したところを事細かく調査する時間を減らすことはできた

問3:あなたの組織ではソフトウェアテストにおいても アジャイルは必須か？それはなぜか？

- Mandatory, especially autonomous culture / teams
  - Agileのような提唱されている形を実現するかはチームに依存するが、テスト自動化や自動サブミット、モニタリングなどのプラクティスは必須
  - 自立して行動する人たちのチームにする、
- Focus on user values, developing features, build rapid release cycle
  - 人の労力を機能開発、価値検証に時間を割く、短時間のリリースを実現するため

問4:Flaky Test, 保守性など、自動テストの品質特性の課題をソフトウェアエンジニアリングで解決している事例を教えてください。そのほかの品質特性もあれば教えてください。

- Containerize environment / CI/CD、Web開発では開発環境のコンテナ化は行っている
- Design automated test based on test pyramid and size
  - 例えばテストピラミッドやサイズだと、大きいものを分割して下位に持っていくなど
- Make stable test environment using emulator/simulator/real devices properly
  - モバイルでは、テスト実行環境の不安定さを環境を変えながら安定させようとしたりしている
- Retry failed tests
  - 確率的に失敗するテストのリトライ機構
- Conduct tests in parallel and make running time short
- Split one big module to small chunks / ”アプリの分割”ということも取り組んでいる
  - Each interfaces should guided by CDC testing or type based checking