

Should you Upgrade Official Docker Hub Images in Production Environments?

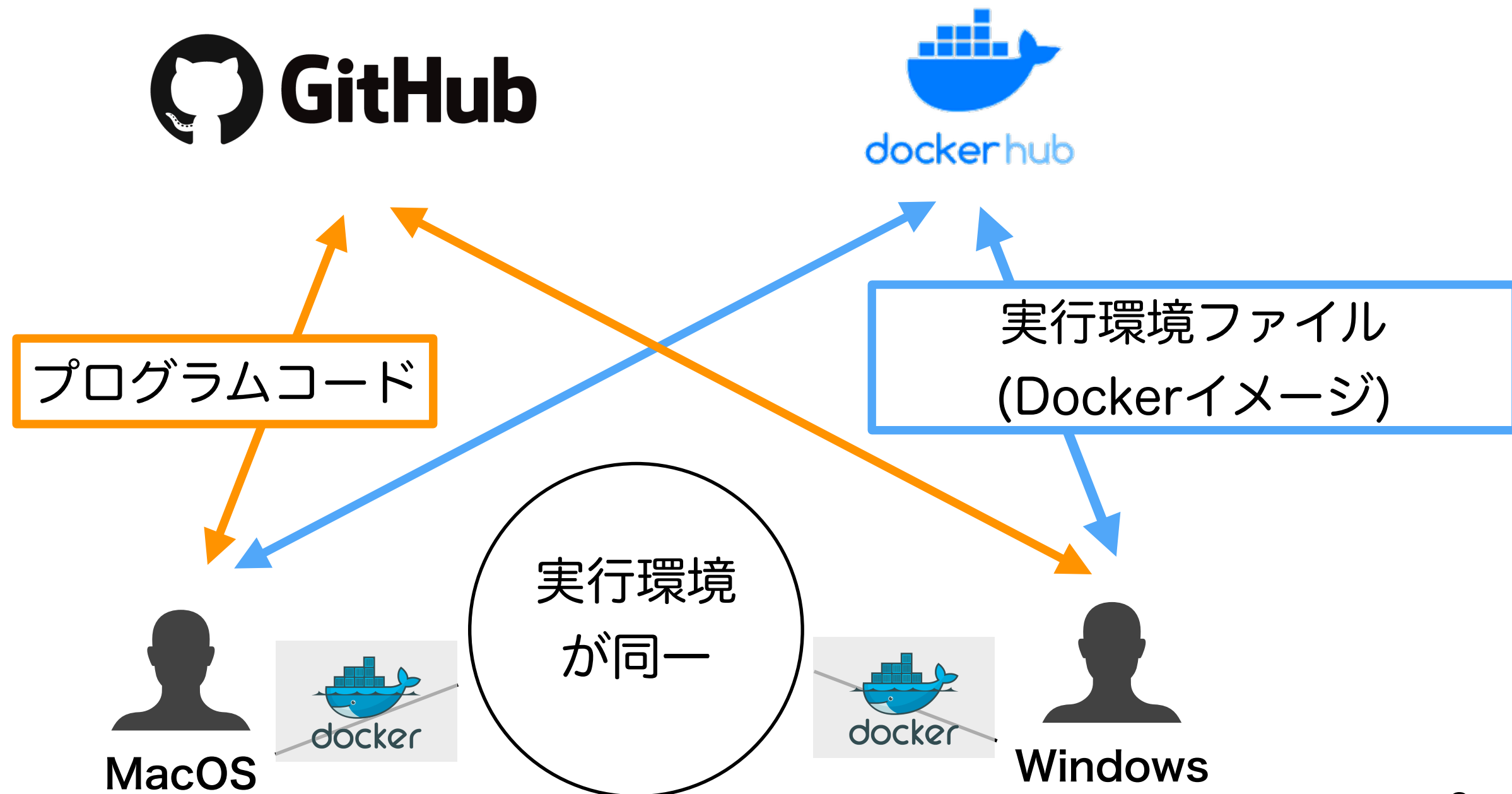
S. Gholami, H. Khazaei and C. -P. Bezemer,
2021 IEEE/ACM 43rd International Conference on Software Engineering:
New Ideas and Emerging Results (ICSE-NIER), 2021,
pp. 101-105, doi: 10.1109/ICSE-NIER52604.2021.00029.

https://www.researchgate.net/publication/348884285_Should_you_Upgrade_Official_Docker_Hub_Images_in_Production_Environments

対象技術

Docker

軽量なコンテナ型の仮想環境を
作成・配布・実行するためのプラットフォーム



背景

- Dockerコンテナはソフトウェアコードと必要なパッケージやライブラリの依存関係をすべてカプセル化したイメージで構成されている
- Dockerコンテナを本番環境にデプロイし、新しいイメージから作成されたコンテナを置き換える in-place upgrade※を適応するのが容易
- Dockerイメージをアップグレードするたびに多くのパッケージが一度に変更される可能性があり、アプリケーションのパフォーマンスやセキュリティが低下する可能性がある

※ 稼働中のシステムに直接ソフトウェアの更新や入れ替えを行うこと

※ 稼働中のシステムを一旦停止し、すべてのソフトウェアを新しいものに入れ替えて再稼働する意味

関連研究

○ Dockerイメージのセキュリティに関する研究

- イメージが親イメージからセキュリティの脆弱性を継承することを明らかに[13]
- npmパッケージの脆弱性がDockerイメージに影響を与える可能性がある[16]

○ パッケージ変更に伴うリスクに関する研究

- Node.jsやJavaなど様々な環境や言語で研究が行われている
- パッケージの変更によって、そのパッケージに依存するアプリケーションの機能が破壊、パフォーマンスが低下、セキュリティ上の脆弱性が生じるなどの可能性がある[8, 9, 12, 15]

Dockerイメージにin-place upgradeを適用することは簡単だが、
内部パッケージに起因する問題によってシステム全体を危険にさらすこと

本研究の目的

目的

Dockerイメージにin-place upgradeを適応した際に
生じるリスクを明らかにする

- ネイティブ(OS), Node, Pythonの各パッケージに焦点を当て
どのタイプ※のアプリケーションのイメージがより多くのパッケー
ジ変更を行う傾向があるかを調査する

※ Docker Hub公式リポジトリには14のカテゴリがあり, そのカテゴリごとに比較を行う
(分析対象の中でどのカテゴリにも属していない38リポジトリはその他とする)

方法論 | 流れ

○ リポジトリの選択

- Docker Hub公式リポジトリを選択
- 160のうち、Dockerイメージを含んでいる158リポジトリを調査対象とする

○ イメージタグの収集

- `source{d}[14]`を利用して、各リポジトリのイメージタグ37000個のタグを収集

○ パッケージ名と最終更新日の収集

- `source{d}[14]`を利用して、ネイティブ(OSで使用される),
Node(Node.jsアプリケーション用), Pythonの各パッケージを分析
- `docker inspect`コマンドを用いて最終更新日を収集

○ パッケージの変更点の特定

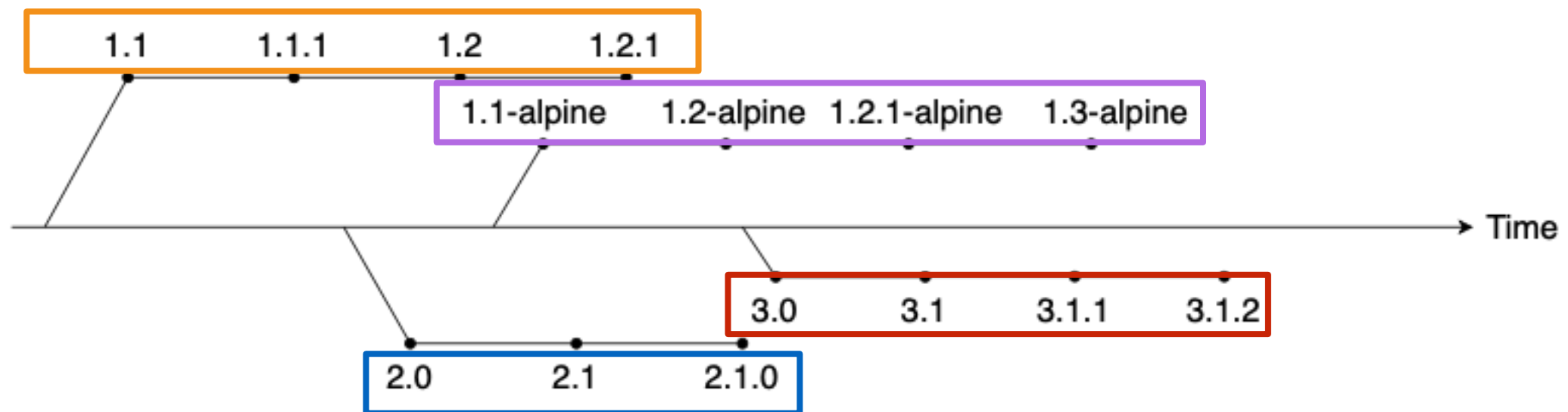
- 次ページで詳細

方法論 | パッケージの変更点の特定

○ パッケージの変更点の特定

- 異なるブランチ同士では比較しない

(明確にブランチが分かれてないリポジトリは分析対象外)



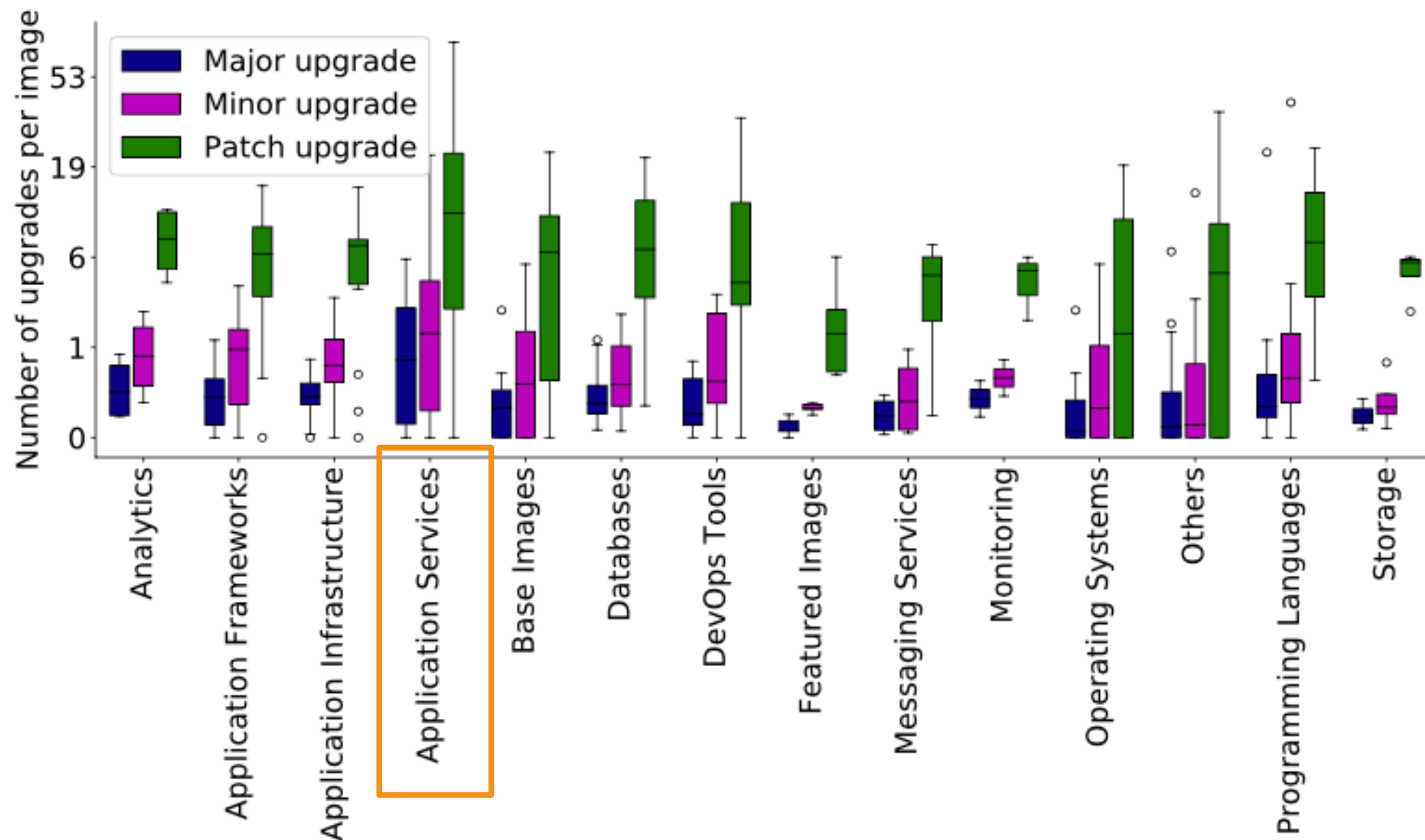
○ 変更の種類

- アップグレード or ダウングレード
- メジャー変更 (1.2.0 → 2.0)
- マイナー変更 (1.2.0 → 1.3.1)
- パッチ変更 (1.2.0 → 1.2.1)

* <https://semver.org/>

結果 | 各カテゴリごとのアップグレード回数

○ 公式Dockerイメージではイメージごとに中央値で8.6個のアップグレードがある



カテゴリ

メジャー変更

マイナー変更

パッチ変更

Application
Services

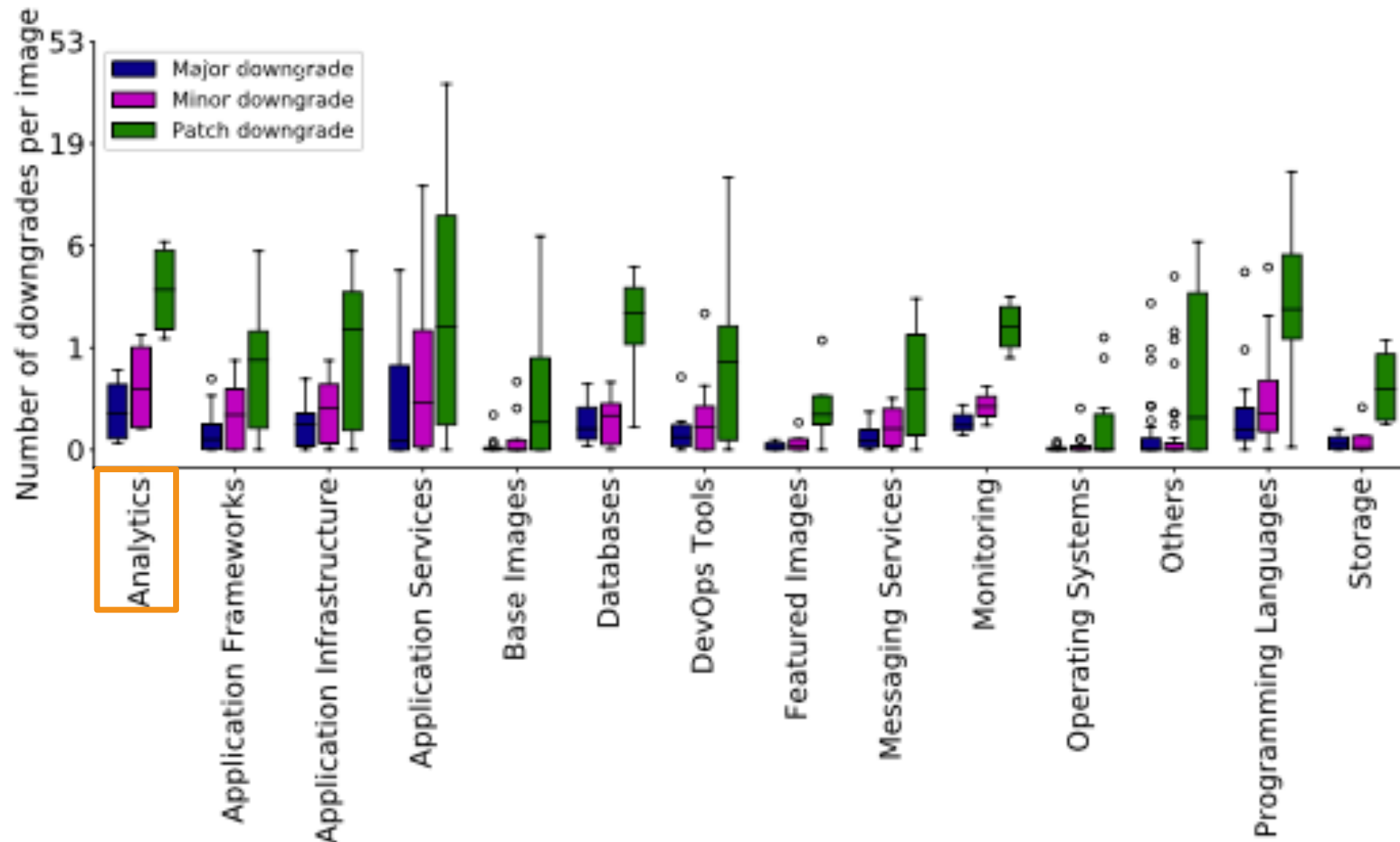
1.4

2.2

11.1

結果 | 各カテゴリごとのダウングレード回数

○ 公式Dockerイメージではイメージごとに中央値で2.1個のダウングレードがある



カテゴリ

メジャー変更

マイナー変更

パッチ変更

Analytics

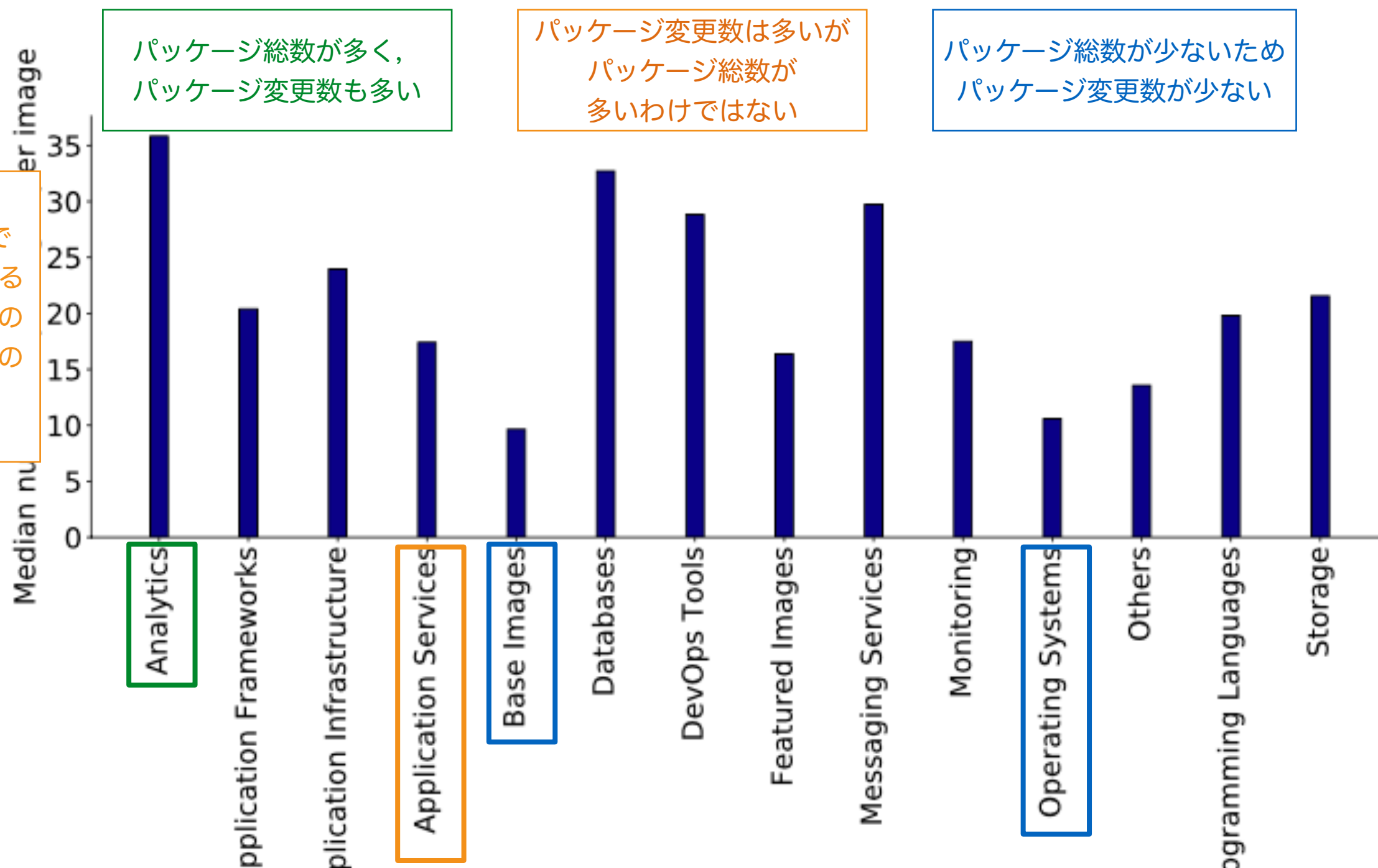
0.4

0.8

3.8

結果 | 各カテゴリごとのパッケージ数

○ 公式Dockerイメージでは中央値で最大36個のサードパーティ製パッケージが指定されている



Analyticsアプリケーションのイメージは他のタイプのイメージよりも安定していないことを示唆する

結果 | 頻繁に変更されているパッケージ

- 公式Dockerイメージ：9000個以上の異なるパッケージが使用
- 最も変更が多いパッケージ：ユーティリティーパッケージ
 - tzdata、base-files、libsystemd0、libudev1、opensslなど
- ユーティリティーパッケージのアップグレードは
非互換性を引き起こすことや新しいバージョンにバグがある可能性が高く、可能な限り避けたい

Docker Hub公式リポジトリからイメージのin-place upgradeを行う際には
開発者は慎重になる必要がある

まとめ

- ネイティブ(OS), Node, Pythonの各パッケージについて分析した
 - 公式Docker Hubリポジトリにある37000以上のイメージ
- 中央値で8.6個のイメージのパッケージがアップグレードされる
 - 対象となるアプリケーションのみがアップデートされる
 - 従来のアップデートシナリオと比較して高い値
- Analyticsアプリケーションのイメージは他イメージよりも不安定
- 最も変更頻度が高いのはユーティリティーパッケージ
 - tzdata、base-files、libsystemd0、libudev1、opensslなど
- in-place upgradeのリスクをさらに調査し、定量化する必要がある

所感

- 最も頻繁に変更されているユーティリティーパッケージのアップグレードに関してほとんど触れられていない
 - 非互換性などを引き起こさないようにするためにはどうしたらいいのかみたいな分析って何をすればいいのかな。。
- Analyticsアプリのイメージをより深く分析したらリスクを定量化する手がかりが見える？
 - なんで安定していないのか，は定性的な分析がいる？
- カテゴリごとの変更数等の具体的な数字表で見たかった