

Learning from, Understanding, and Supporting DevOps Artifacts for Docker

ICSE2020

Jordan Henkel, Christian Bird,

Shuvendu K. Lahiri, Thomas Reps

背景

- DevOpsツールの普及
 - 高品質を維持しながらシステムの変更をコミットしてから通常の運用に至るまでの時間を短縮することを目的とした一連のプラクティス
- DevOpsツールはそれぞれカスタムDSLを持っている
 - DockerのDockerfileなど
- DevOpsアーティファクトの記述方法について
 - 記述方法のベストプラクティスを開発者が知らない場合が多い
 - ビルド時間が長くなる, サイズが大きくなる, ビルドが壊れる

課題

- DevOpsアーティファクトの開発を支援するツール
 - VS Code Docker, Hadolint
 - ツールはベストプラクティスの採用には結びついていない

Dockerなどのツールに対する静的な開発者支援は、
浅い構文検証に限られている

- Dockerfileのベストプラクティス
 - https://docs.docker.jp/engine/articles/dockerfile_best-practice.html

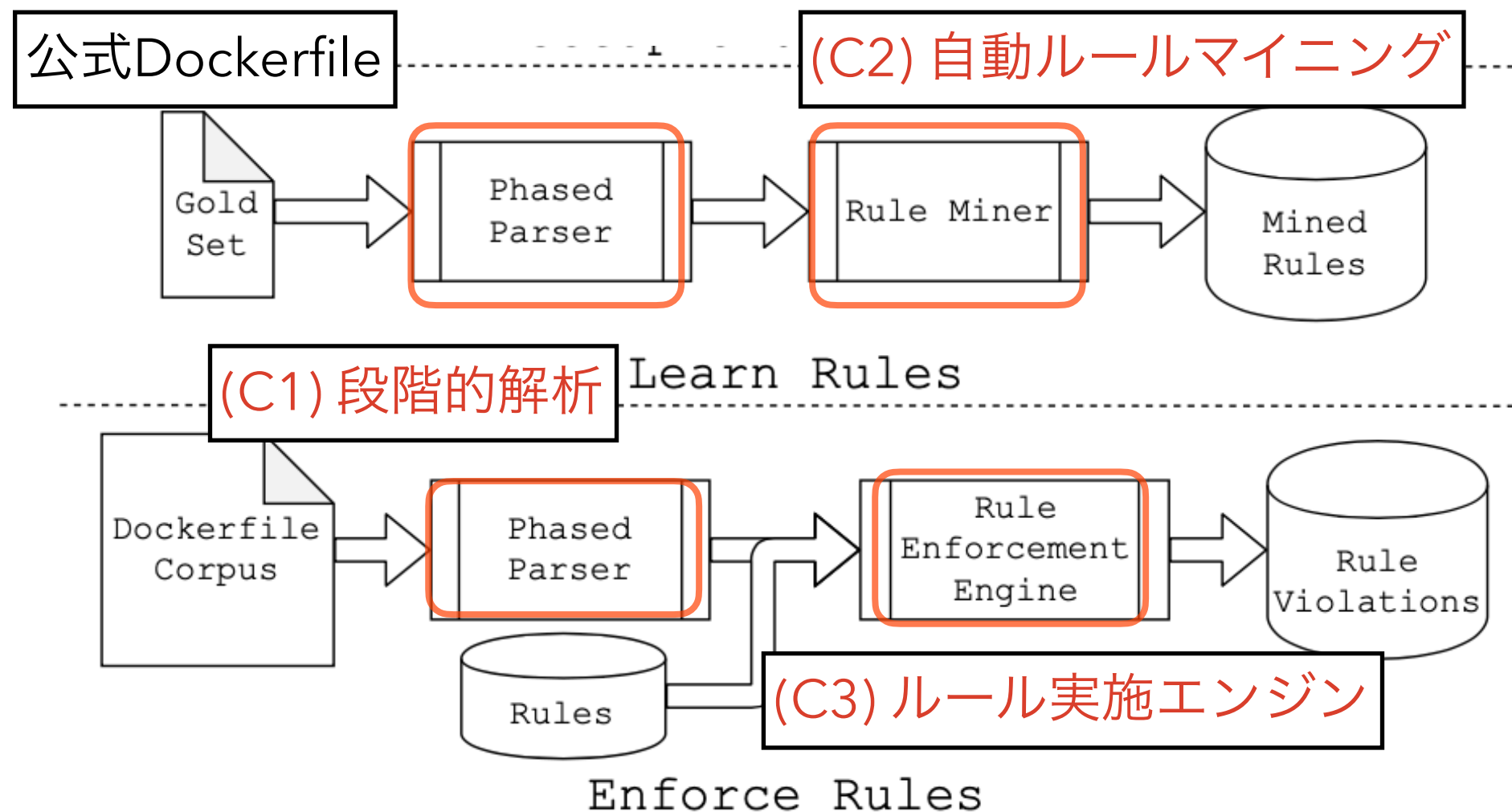
研究の目的

DevOpsの成果物の領域において、より効果的なセマンティクスを考慮したツールの必要性に対処する

- Dockerfileのルールをマイニングし,
Dockerfileを分析することのできるツールセットの提案
- 提案ツールセット : binnacle

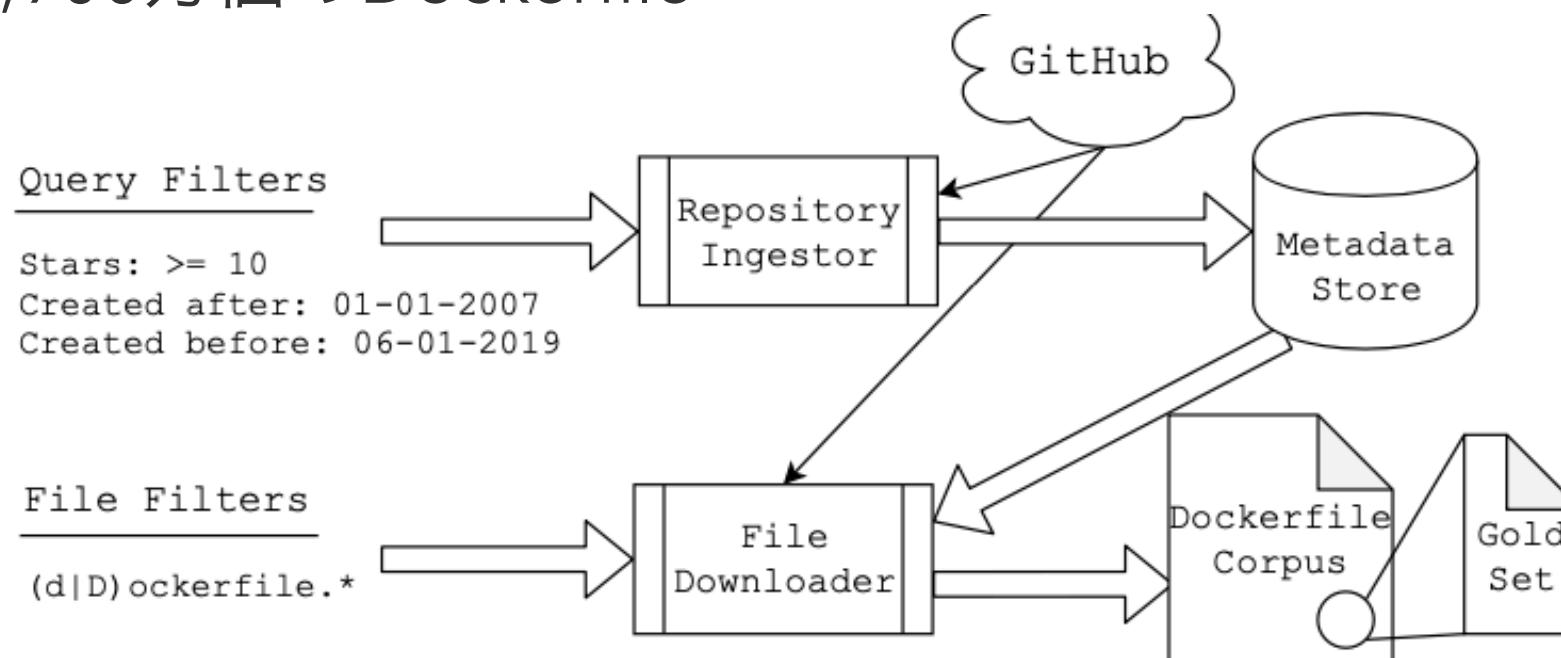
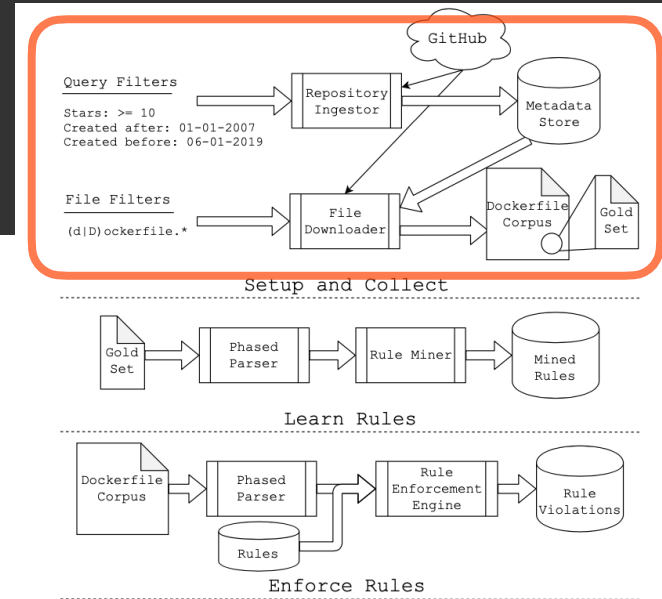
提案ツールの全体図

- 公式Dockerfileからルールを自動抽出
- Dockerfileがマイニングしたルールに準拠しているのかを判定する



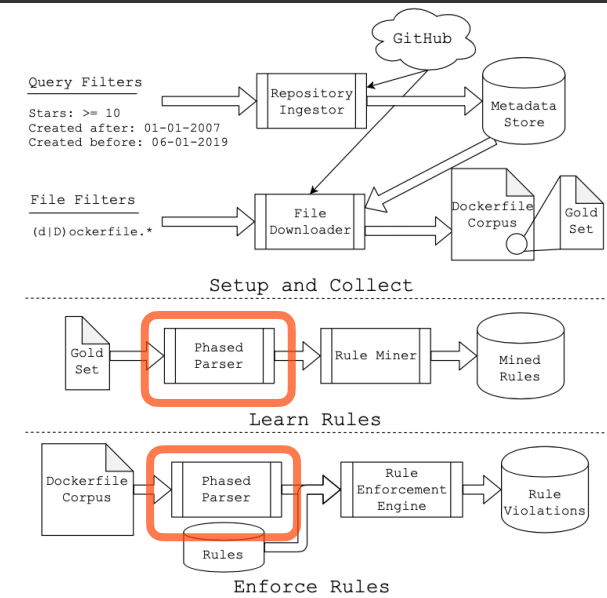
Dockerfileの収集

- 専門家に書かれたDockerfile
 - 約400個の公式Dockerfile (Goldsetと呼称)
 - 産業用リポジトリの約5,000個のDockerfile
- OSSリポジトリにあるDockerfile
 - GitHub 上のすべての公開リポジトリ & スターが 10 個以上
 - 約21,900万個のDockerfile



(C1) 段階的解析

- Dockerfileの構造化された表現を生成



Dockerfile

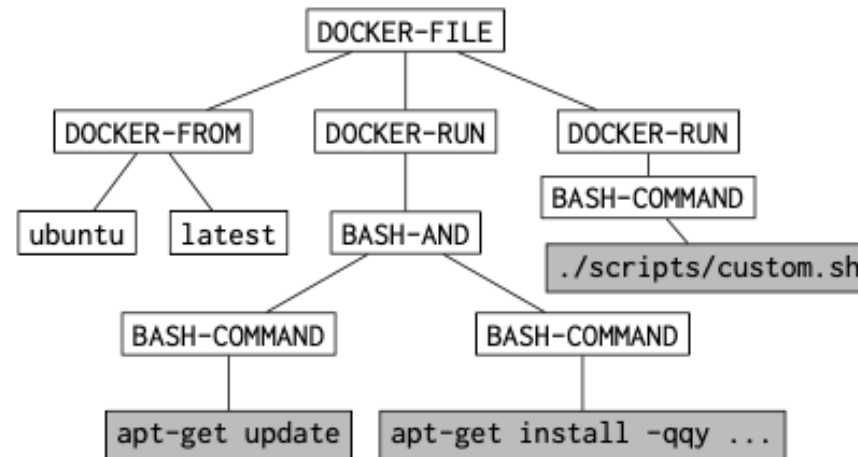
```
FROM ubuntu:latest

RUN apt-get update && \
    apt-get install -qqy ...

RUN ./scripts/custom.sh
```

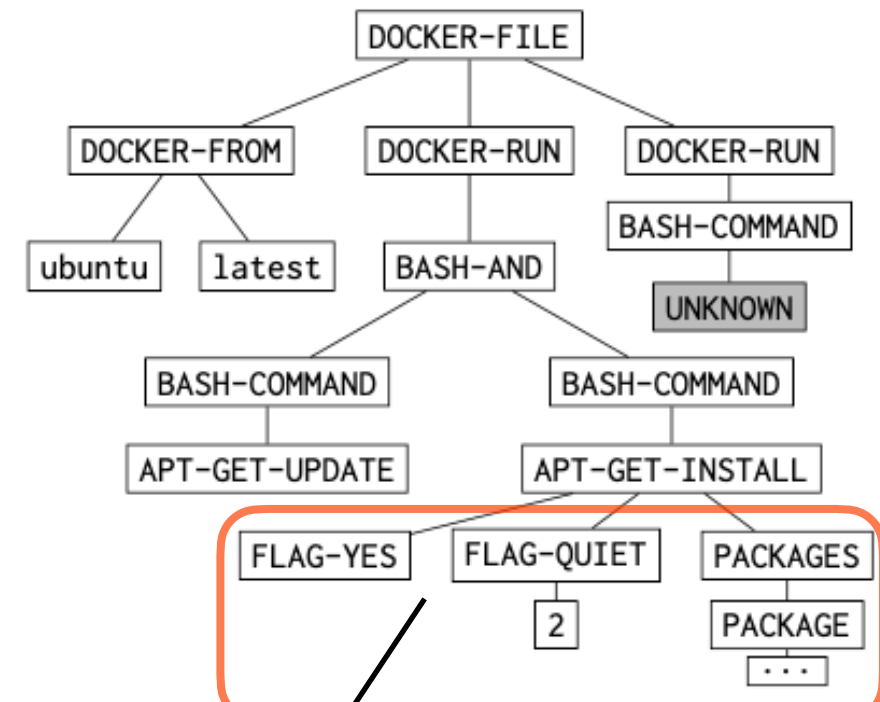
(a) An example Dockerfile

コマンドの粒度



第2段階

第3段階

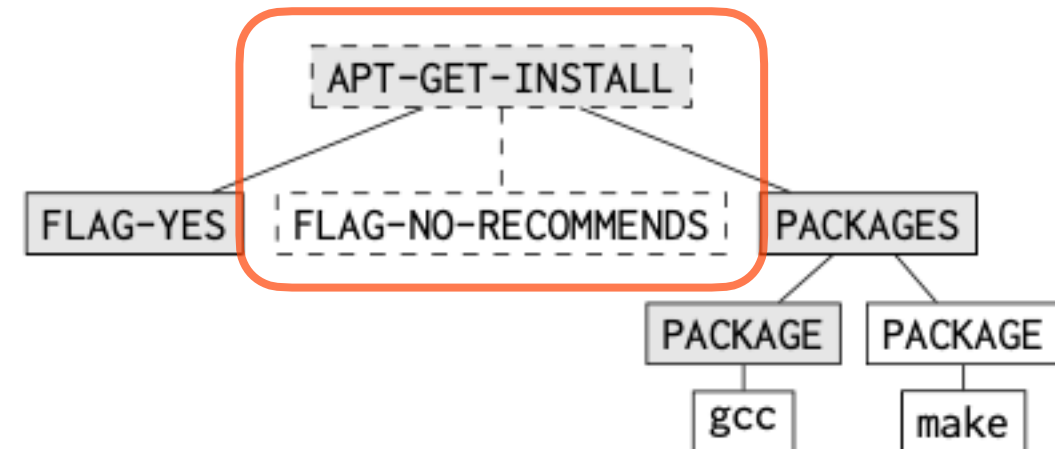
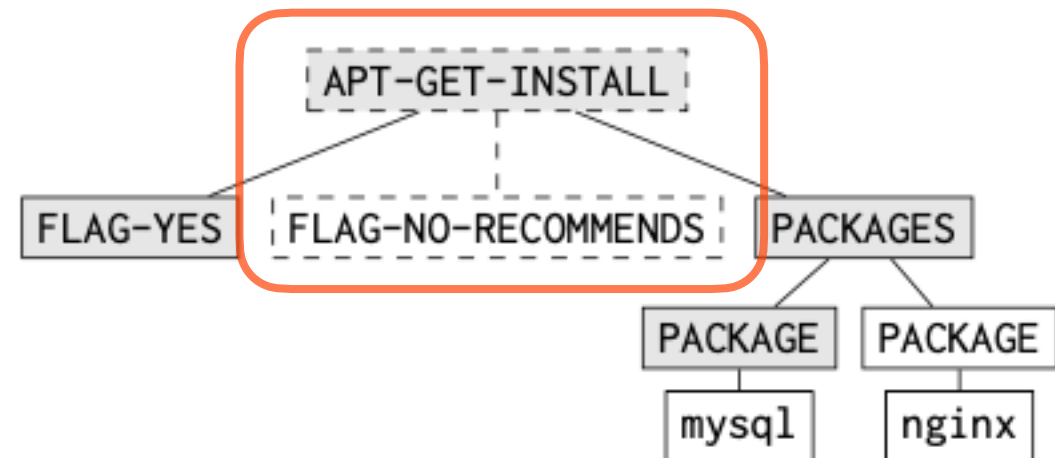
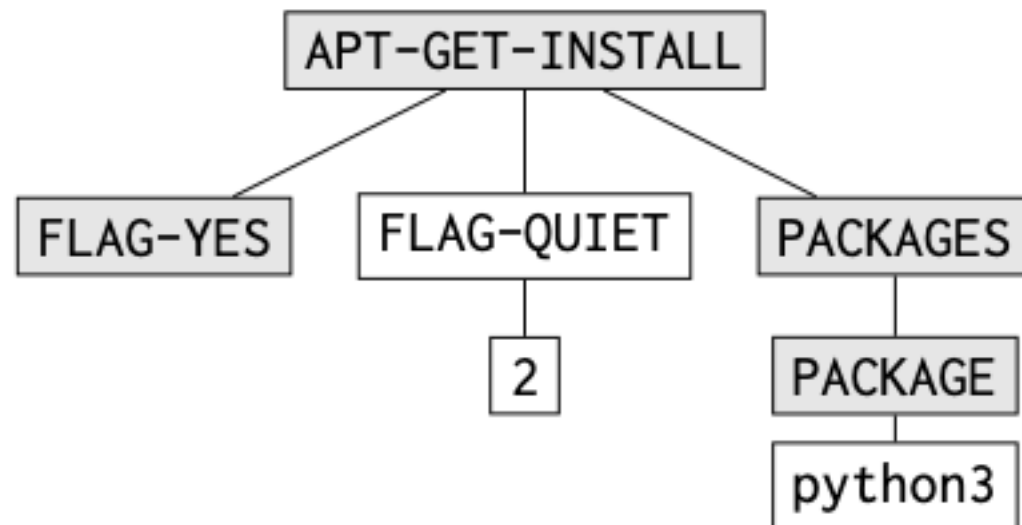
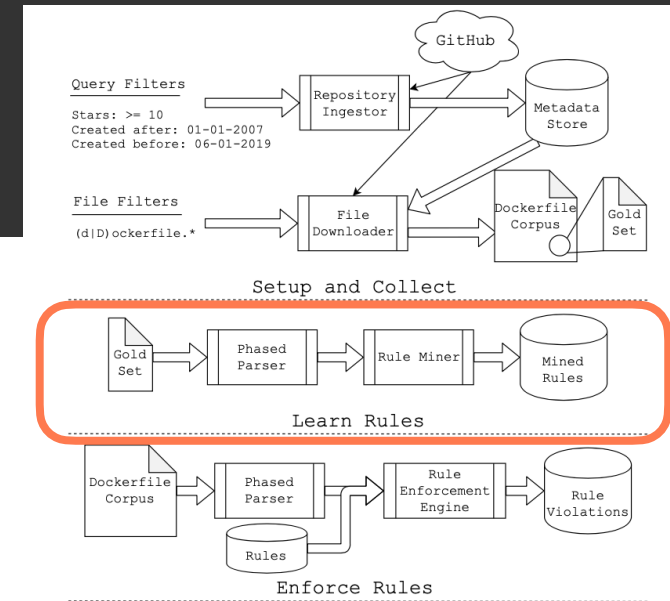


第1段階

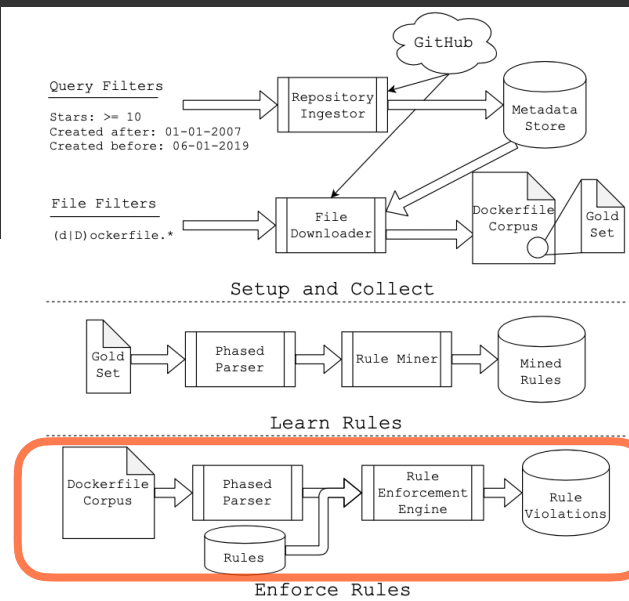
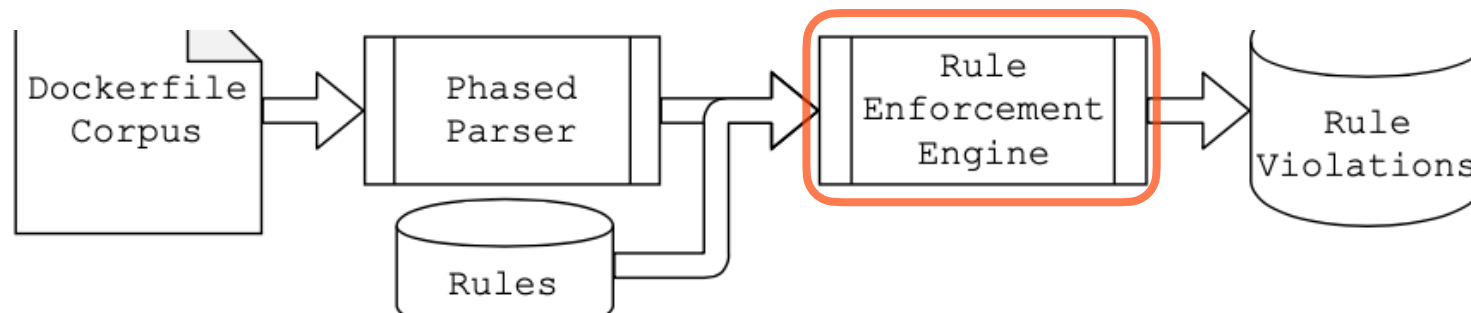
オプションの粒度でASTを作成

(C2) 自動ルールマイニング

- 頻度の高いサブツリーマイニングをベースにした自動ルールマイニングを構築



(C3) ルール実施エンジン



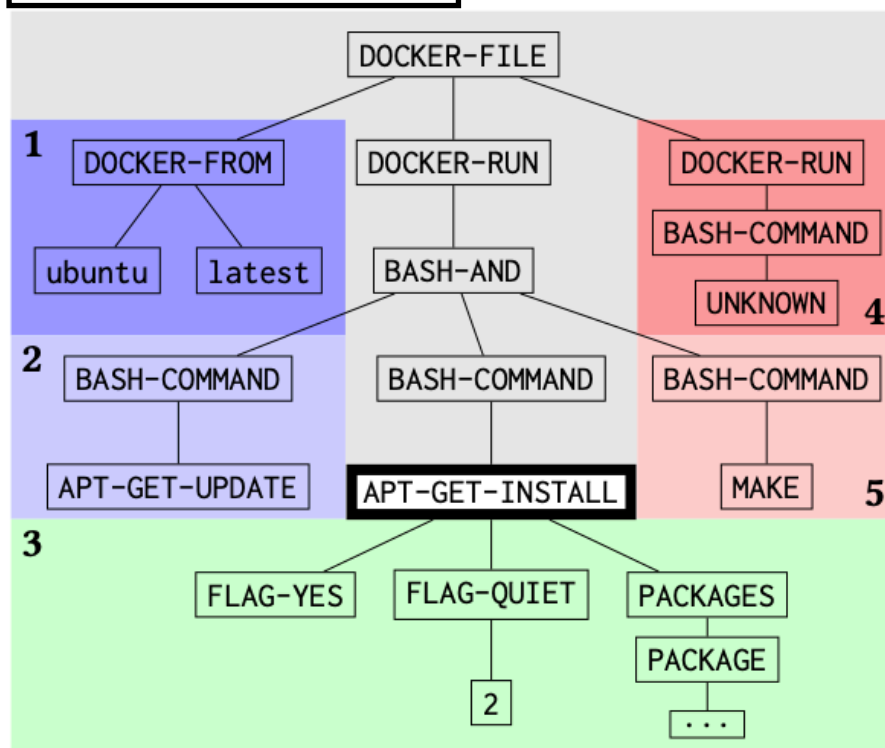
ルール1

apt get installを行う前に
apt get updateを行う

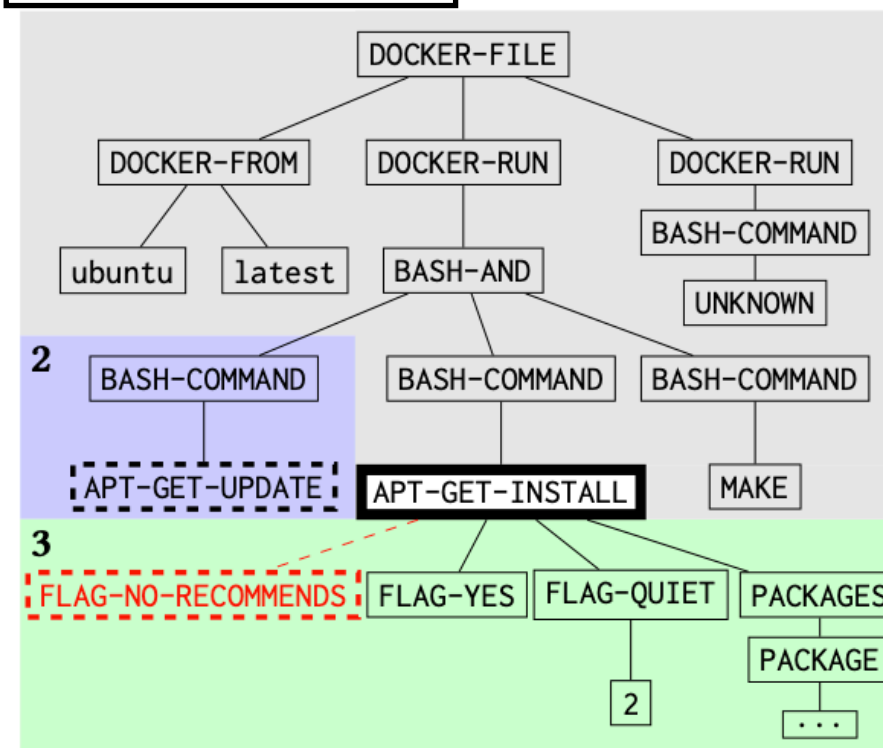
ルール2

apt get installの子は全て
flag-no-recommendsをつける

ルール1について

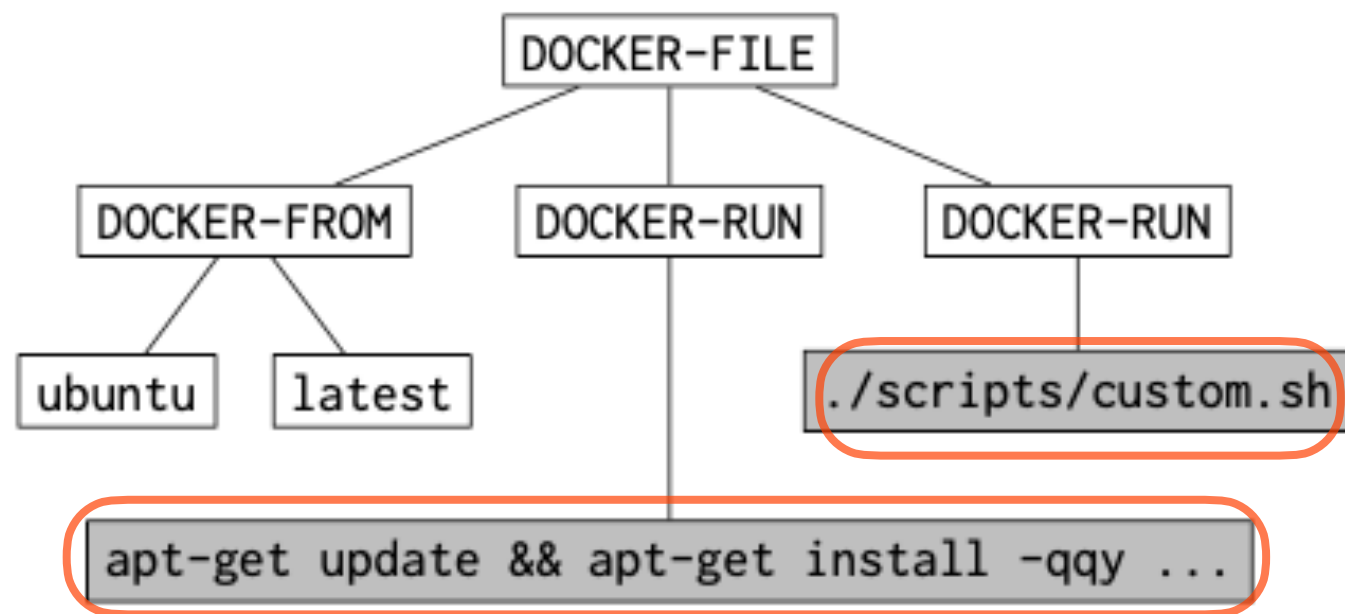


ルール2について

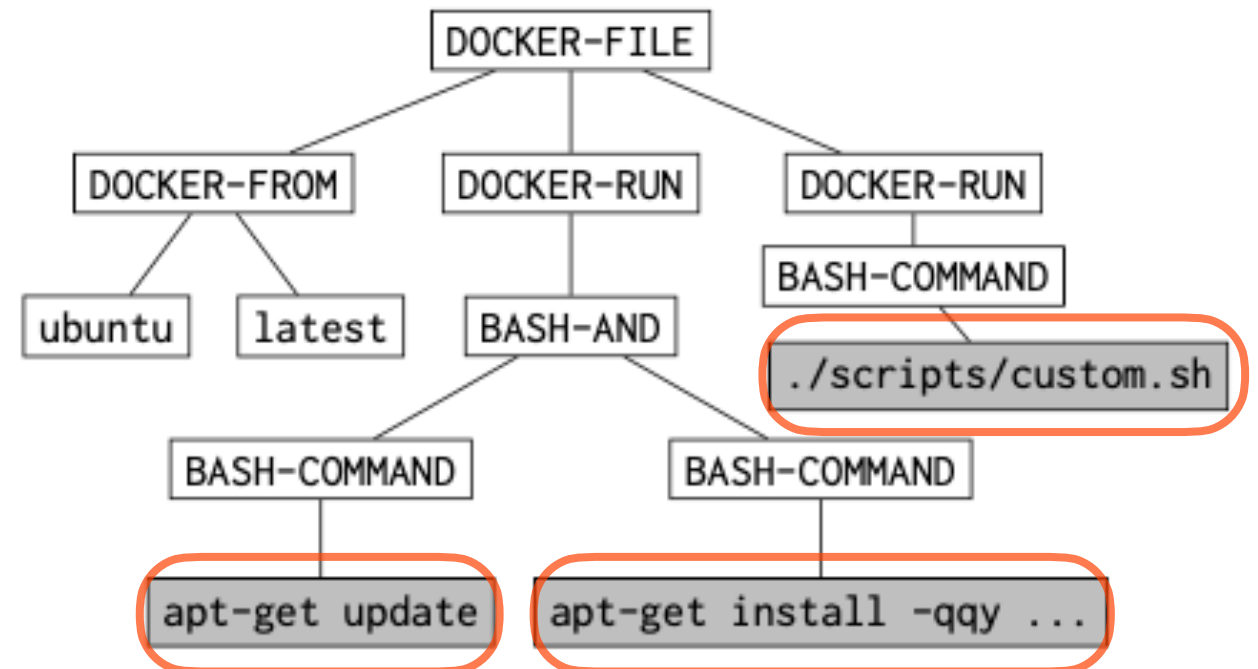


段階的解析 - 評価 -

- ツリーとして表されるDockerfileに存在する有用な情報の量を定量化するための指標が必要
- ツリー内で実質的に解釈不可能な葉ノード(EU)の割合



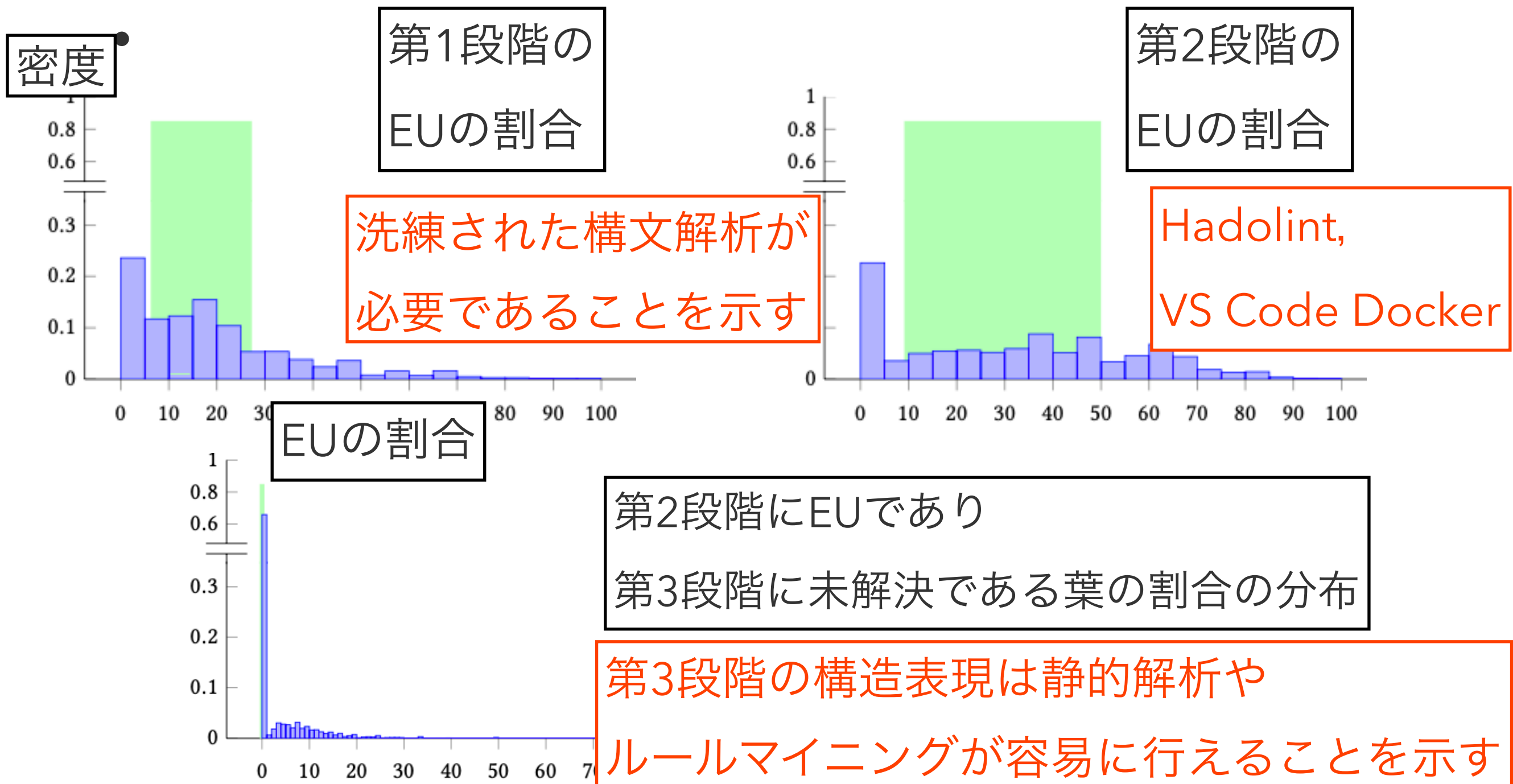
第1段階の解析後



第2段階の解析後

段階的解析 - 評価 -

- ツリー内で実質的に解釈不可能な葉ノード(EU)の割合



自動ルールマイニング - 評価 -

- データから自動的に抽出できるルール数は？
 - 26個 (従来ルールマイナーだと数千)
 - 手動で作成ゴールドルールの2/3を抽出できた
- 新しいルールを何個見つけたのか
 - 手動で見落としていた16のルールを新たに発見

自動マイニングによって、新しいルールを発見し
手動の労力を節約することができる

ルール実施エンジン - 結果 -

- ゴールドルールと照らし合わせて、ルールの違反率で評価

評価データ	違反率
専門家のDockerfile	6.57%
OSSリポジトリのDockerfile	37.27%
産業用リポジトリのDockerfile	OSSリポジトリと同程度

産業界の開発者を支援するためにも活用できる

まとめ

- DevOps アーティファクトの開発を支援する
binnacleツールセットを提案
 - 段階的解析
 - 自動ルールマイニング
 - ルール実施エンジン
- GitHub上のDockerfileは、 公式Dockerfileと比較して、
ゴールドルールの違反に関して平均して約6倍悪い

所感

- 意外とGithubリポジトリのDockerfileと公式Dockerfileに差はあるのだなと思った.
- ASTや, アソシエーションルールマイニングの話を理解するのが困難だった.

提案手法：binnacleツールセット

- DevOpsアーティファクトの学習と理解における課題
 - 入れ子になった言語の課題
 - ルールエンコーディングと自動ルールマイニングの課題
 - 静的ルール実施の課題
- 提案手法：binnacleツールセット
 - 段階的パーシング
 - ルールマイニング
 - ルールエンフォースメント