

Architecture Smells and Pareto Principle: A Preliminary Empirical Exploration

Alexandra-Maria Chaniotaki, Tushar Sharma

2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)

発表者：松尾 和輝

背景

- アーキテクチャの品質は非常に重要である.
- アーキテクチャのリファクタリングには高いリスクと労力が伴うため、余りリファクタリングが行われない.

目的

- パレートの法則※を用いて，アーキテクチャの臭いの特性を実証的に調査する．

※パレートの法則

二割の要素が，全体の八割を生み出しているというばらつきの状態を示す．



1: <https://blog.kairosmarketing.net/marketing-strategy/how-to-leverage-pareto-law-into-marketing-activities/>

リサーチクエスション

RQ1：アーキテクチャの臭いはパレートの原理に従っているか？

RQ2：プログラミング言語の違いによって、現象の現れ方は異なるのか？

RQ3：リポジトリのサイズはパレートの原則の遵守に影響するか？

プロジェクトの選定

- 選定にはRepoReapers²を使用.

GitHubリポジトリを分析し、8つの次元(アーキテクチャ, 継続的インテグレーション, ユニットテスト, コミュニティ, ドキュメント, 履歴, 課題, ライセンス)に基づいて, 品質特性を提供するツール

- RepoReapersの8つの次元のうち少なくとも7つのスコアが0より大きいJavaまたはC#のコードを含むリポジトリ
 - 星の数が10個以上
 - コード行数が1000行以上
- 以上を満たす750のjavaと361のC#のリポジトリを取得

アーキテクチャの臭い

- DesigniteJava³とDesignite⁴というツールを用いて、以下のようなアーキテクチャの臭いを検出

Architecture smell	Description
Cyclic dependency	Two or more architecture-level components depend on each other directly or indirectly [4], [27].
Unstable dependency	A component depends on other less stable components [28].
Ambiguous interface	A component offers only a single, general entry-point into the component [29].
God component	A component is excessively large either in terms of Lines Of Code (LOC) or number of classes [4].
Feature concentration	A component realizes more than one architectural concerns or features [3].
Scattered functionality	An architectural concern is realized by multiple components [29].
Dense structure	Excessive and dense dependencies among components lead a project to suffer from this smell [30].

3: <http://www.designite-tools.com/designitejava>

4: <http://www.designite-tools.com/>

結果 -RQ1-

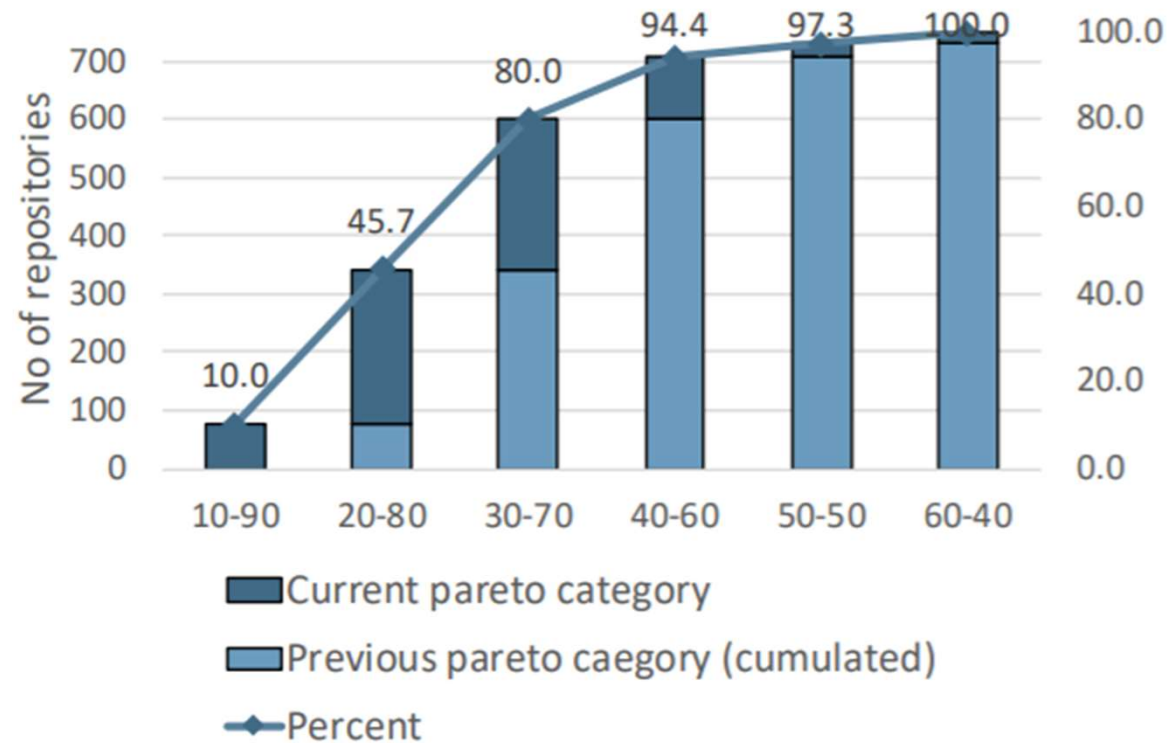
RQ1：アーキテクチャの臭いはパレートの法則に従っているか？

□アプローチ

- 全てのJavaで記述されたりポジトリのコンポーネントで発見された全てのアーキテクチャの臭いを記録. その後, パレート分析を行う.
- アーキテクチャの臭いが最も多いコンポーネントの20%が, アーキテクチャの臭い全体の80%を占めているかどうかを調べる.

結果 -RQ1-

- 45.7%がパレートの法則に従っている。



結果 -RQ2-

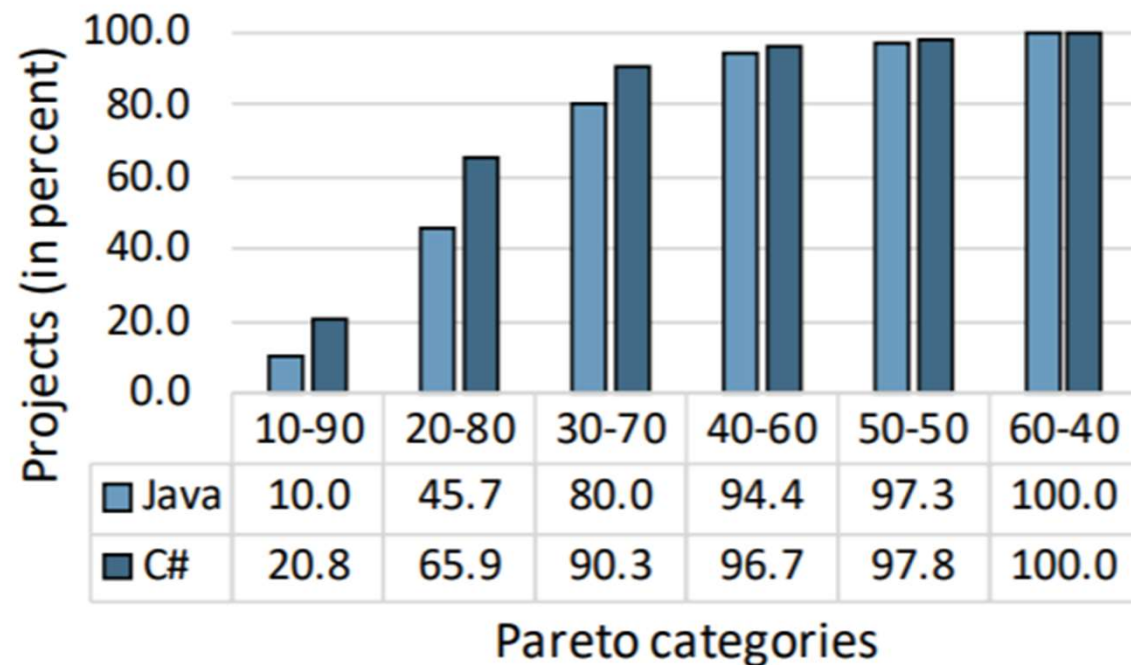
RQ2. プログラミング言語の違いによって、現象の現れ方は異なるのか？

□アプローチ

- RQ1のアプローチをC#のリポジトリで行う.

結果 -RQ2-

- Javaは45.7%であるのに対し， C#は65.9%がパレートの法則に従っている．



結果 -RQ3-

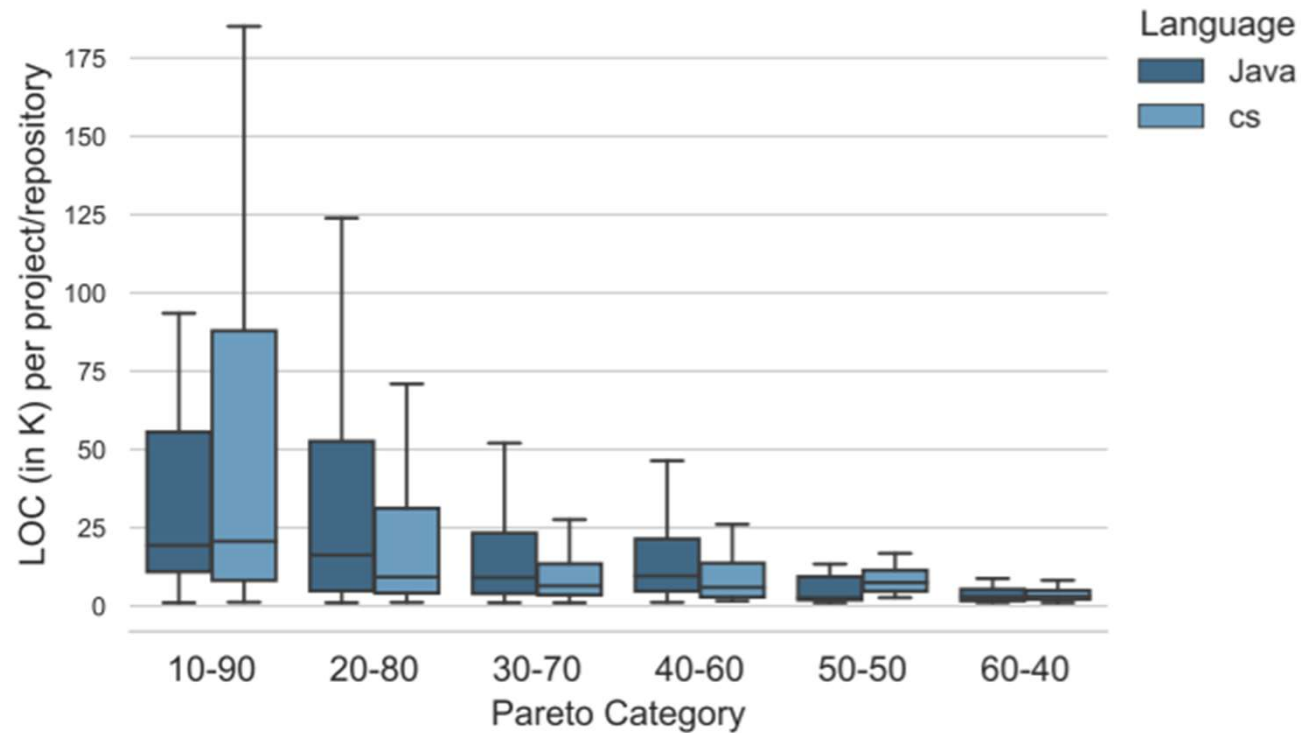
RQ3: リポジトリのサイズはパレート原理の遵守に影響するか？

□ アプローチ

- 分析対象のシステムのLOCと，割り当てられたパレート区分との間でスピアマンの順位相関分析を行う．

結果 -RQ3-

- 10-90, 20-80のカテゴリは他のカテゴリよりもリポジトリのサイズが大きい傾向にある.



結果 -RQ3-

- パレート区分と全リポジトリに属するLOCの間で得られたスピアマン相関係数は-0.29であった.
- C#リポジトリは-0.29
- Javaリポジトリは-0.26
- LOCがパレート区分と低い負の相関を持つことを示している.

まとめ

- パレートの法則を用いたアーキテクチャの臭いの特性の調査
- JavaやC#で見つかったアーキテクチャの臭いに対しパレート分析を行う
- Javaは45.7%, C#は65.9%がパレートの法則に従っている
- LOCとパレート区分の間には、低い負の相関がある

所感

- 他のプログラミング言語では，結果がどうなるか気になった.
- なぜJavaとC#でパレートの法則に従っている割合が大きく違うのかが気になった.