

Improving Source Code Readability: Theory and Practice

発表者：ダニエル

Sarah Fakhoury*, Devjeet Roy*, Sk. Adnan Hassan[†], Venera Arnaoudova*

*School of Electrical Engineering and Computer Science, Washington State University, Pullman (WA), USA

[†]Infosapex Limited, Dhaka, Bangladesh

{sfakhour, droy, varnaoud}@eecs.wsu.edu, adnan@infosapex.com

背景

-コード品質メトリクス:

- コードの臭いを検出し、改善の機会を見つけるために広く使用されています。

-開発者の認識:

- 最近の調査では、これらのメトリクスと開発者が品質改善として認識するものの間に不一致があることが示唆されています。

-コード改善ツール:

- このメトリクスに依存するため、有効性を得るには開発者の認識との整合が必要です。

目的

目的:

- さまざまなモデルと測定法がコミットレベルでの可読性の向上をどの程度捉えているかを調査します。

視点:

- 日々のソフトウェア保守タスクで可読性を測定することに関心のある研究者および開発者。

範囲:

- GitHub から収集された63のエンジニアリングされたJava プロジェクトからの548のコミットに基づく評価。

調査方法/ 研究デザイン

データ収集:

- **Reaper** を使用して GitHub からコミットを収集し、エンジニアリングされたプロジェクトを識別します。
- **キーワード マッチング** により、読みやすさのコミットを識別します。
- コミットを手動でフィルタリングして整理し、読みやすさの変更を**分離**します。

結果:

- 読みやすさのコミット **548 件**
- データポイント **2,323 件**
- Java プロジェクト **63 件**

調査方法/ 研究デザイン

読みやすさモデルの評価:

- Scalabrino、Dorn、および Combined モデルを使用して、読みやすさのコミットの前後のファイル进行分析します。

読みやすさの改善を捉えるモデルの能力を評価します。

- **Scalabrino's Model:** Measures quality of source code lexicon (comments and identifier consistency, number of meanings, etc.).
- **Dorn's Model:** Assesses visual, spatial, alignment, and linguistic aspects.
- **Combined Model:** Integrates Scalabrino's, Dorn's, Buse & Weimer's, and Posnett's models.
- **Evaluation:** Run all three models on files before and after readability commits to evaluate their effectiveness.

Research Question and 調査結果

RQ1: Are state-of-the-art readability models able to capture readability improvements in practice?

事例:

- 単一の識別子の名前を変更する: 読みやすさスコアは変わらない
- 機能を変更せずにコードをフォーマットして再グループ化する: 読みやすさスコアが低下する
- 条件をブール変数に置き換える: 読みやすさスコアがわずかに低下する

Dorn's Model:

- Mean readability score before changes: 0.793
- Mean readability score after changes: 0.798
- Mean increase: 0.005
- Indicates improvement in 40.7% of files

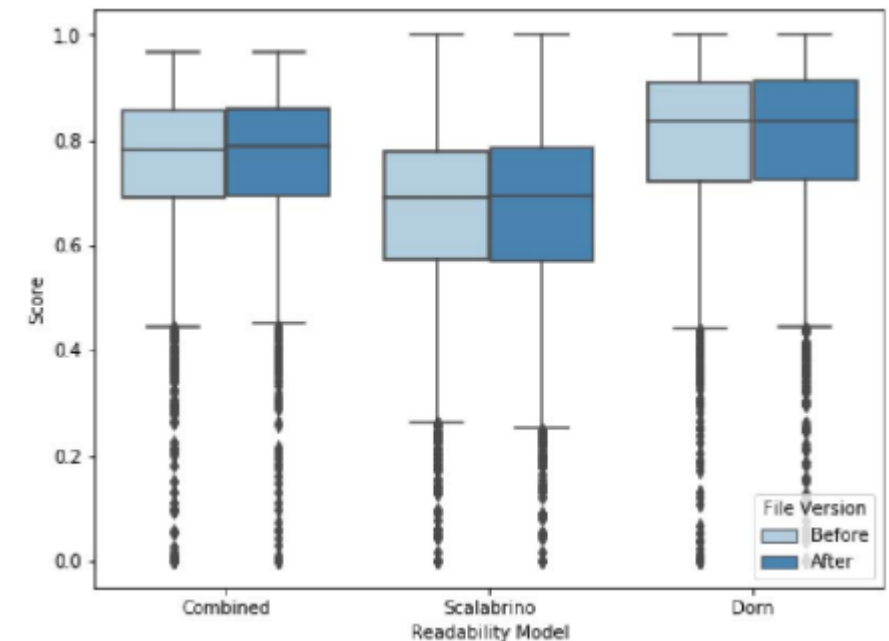


Fig. 1. Readability scores generated by Dorn's, Scalabrino's, and the Combined models for files before and after readability improvements.

Research Question and 調査結果

RQ1: Are state-of-the-art readability models able to capture readability improvements in practice?

読みやすさモデルでは、段階的な読みやすさの改善を効果的に検出できない。

日常的なソフトウェア保守タスクにおいて、読みやすさの小さな実用的な改善を捉えることができるモデルが必要。

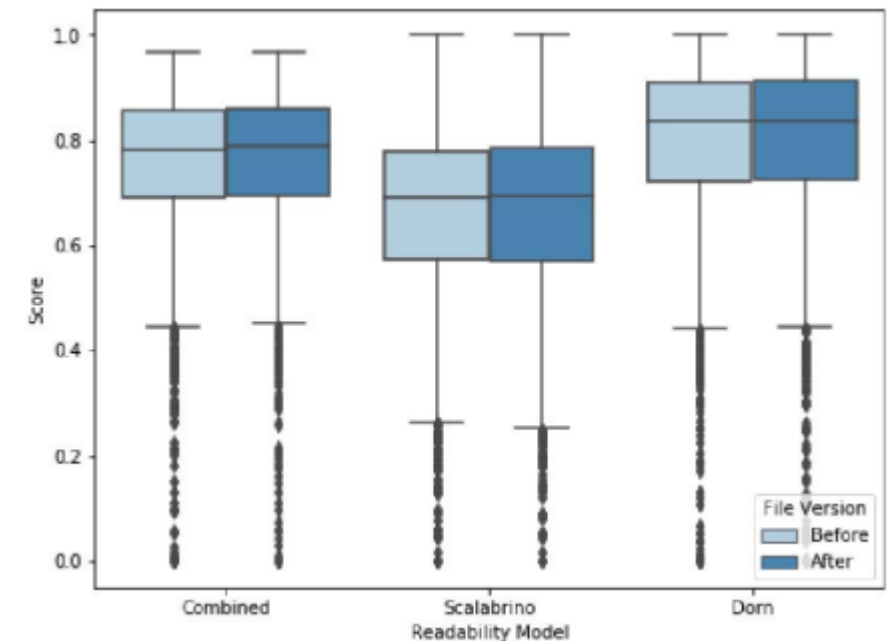


Fig. 1. Readability scores generated by Dorn's, Scalabrino's, and the Combined models for files before and after readability improvements.

Research Question and 調査結果

RQ2: What types of changes do developers perform during readability improvements?

方法: **ChangeDistiller**、CheckStyle、RefactoringMiner を使用して変更を分析しました。

Types of Changes :

Update(全変更の 40%):

- 最も頻繁に発生する変更の種類

Insert(全変更の 17%):

- Statementの挿入、新しいメソッド、コメント、ドキュメントの挿入が含まれます
- 例: TODO コメントの挿入

Delete(全変更の 27%):

- statementの削除とコメントの削除が含まれます。
- 例: TODO コメントの削除、継承されたドキュメントの削除

TABLE III
CHANGES DETECTED BY CHANGEDISTILLER. OVERALL PERCENT OF CHANGES IS CALCULATED ACROSS ALL DETECTED CHANGES; WITHIN GROUP CHANGES ARE CALCULATED PER CHANGE TYPE GROUP.

Change		Non-Readability		Readability	
		Overall	Group	Overall	Group
Delete	REMOVED_OBJECT_STATE	1.19%	4.97%	1.45%	4.90%
	COMMENT_DELETE	1.24%	5.15%	1.27%	4.31%
	ALTERNATIVE_PART_DELETE	1.43%	5.96%	0.73%	2.46%
	REMOVED_FUNCTIONALITY	2.42%	10.07%	2.04%	6.89%
	STATEMENT_DELETE	16.93%	70.42%	22.05%	74.50%
Insert	COMMENT_INSERT	1.93%	4.65%	0.73%	3.84%
	ADDITIONAL_OBJECT_STATE	3.14%	7.55%	1.72%	9.07%
	ADDITIONAL_FUNCTIONALITY	4.68%	11.26%	2.87%	15.12%
	STATEMENT_INSERT	27.07%	65.12%	10.76%	56.74%
	PARAMETER_INSERT	0.95%	2.27%	1.06%	5.60%
Move	STATEMENT_ORDERING_CHANGE	1.68%	19.45%	2.87%	37.42%
	STATEMENT_PARENT_CHANGE	6.53%	75.41%	4.46%	58.06%
Update	METHOD_RENAMING	0.70%	2.70%	0.53%	1.21%
	DOC_UPDATE	0.96%	3.72%	1.27%	2.91%
	ATTRIBUTE_RENAMING	0.54%	2.11%	2.74%	6.26%
	CONDITION_EXPRESSION_CHANGE	3.49%	13.56%	5.87%	13.42%
	STATEMENT_UPDATE	18.67%	72.59%	30.91%	70.61%

Metrics Collection

SourceMeter:

- ソースコードの詳細な静的分析を実行します

CheckStyle:

- コードがルールに準拠しているかどうかを確認するための静的分析ツールです

ChangeDistiller:

- 抽象構文Tree (AST) を使用して、Java ソースコードのステートメントレベルの変更を抽出し、分類します

RefactoringMiner:

- Java プロジェクト履歴のリファクタリングを検出します

Research Question and 調査結果

RQ2: What types of changes do developers perform during readability improvements?

ReadabilityとNon-readabilityコミットの比較:

- 読みやすさコミット: statementの更新と削除が多くなります。
- 読みやすさ以外のコミット: statementの挿入と機能の変更が多くなります。
- statementの順序変更は、読みやすさコミットでより一般的です。

TABLE III
CHANGES DETECTED BY CHANGEDISTILLER. OVERALL PERCENT OF CHANGES IS CALCULATED ACROSS ALL DETECTED CHANGES; WITHIN GROUP CHANGES ARE CALCULATED PER CHANGE TYPE GROUP.

Change		Non-Readability		Readability	
		Overall	Group	Overall	Group
Delete	REMOVED_OBJECT_STATE	1.19%	4.97%	1.45%	4.90%
	COMMENT_DELETE	1.24%	5.15%	1.27%	4.31%
	ALTERNATIVE_PART_DELETE	1.43%	5.96%	0.73%	2.46%
	REMOVED_FUNCTIONALITY	2.42%	10.07%	2.04%	6.89%
	STATEMENT_DELETE	16.93%	70.42%	22.05%	74.50%
Insert	COMMENT_INSERT	1.93%	4.65%	0.73%	3.84%
	ADDITIONAL_OBJECT_STATE	3.14%	7.55%	1.72%	9.07%
	ADDITIONAL_FUNCTIONALITY	4.68%	11.26%	2.87%	15.12%
	STATEMENT_INSERT	27.07%	65.12%	10.76%	56.74%
	PARAMETER_INSERT	0.95%	2.27%	1.06%	5.60%
Move	STATEMENT_ORDERING_CHANGE	1.68%	19.45%	2.87%	37.42%
	STATEMENT_PARENT_CHANGE	6.53%	75.41%	4.46%	58.06%
Update	METHOD_RENAMING	0.70%	2.70%	0.53%	1.21%
	DOC_UPDATE	0.96%	3.72%	1.27%	2.91%
	ATTRIBUTE_RENAMING	0.54%	2.11%	2.74%	6.26%
	CONDITION_EXPRESSION_CHANGE	3.49%	13.56%	5.87%	13.42%
	STATEMENT_UPDATE	18.67%	72.59%	30.91%	70.61%

考察

これらのモデルでは、読みやすさの改善を効果的に捉えることができないことが確認された。

モデルは日常のメンテナンス作業には適していないと結論付けた。

新しい読みやすさモデルを構築する際には、これらの指標と変更を考慮する必要がある。

今後の作業では、名前の変更とドキュメントの変更を分析する。