形式手法による開発において報告された問題のトピックモデルによる分析

井上 蒼士
京都工芸繊維大学
大学院工芸科学研究科 情報工学専攻
s-inoue@se.is.kit.ac.jp

崔 恩瀞， 西浦 生成， 水野 修
京都工芸繊維大学
情報工学・人間科学系
{echoi, k-nishiura, o-mizuno}@kit.ac.jp

要旨

ソフトウェア仕様の品質がソフトウェア品質に寄与することが報告されており，自然言語による仕様の曖昧さが課題として指摘されている．形式手法はこの課題に対処する手法として注目されているが，形式手法はソフトウェア開発に広く普及しているとは言えない．この要因として学習コストの高さや開発時間の増加などの課題が指摘されているが，これらは形式手法の導入の難しさに焦点が当たっており，形式手法の適用によるソフトウェア開発への影響の報告は少ない．本研究では，*GitHub* リポジトリから収集した問題報告（*Issue*）を用いて形式手法を適用したソフトウェア開発に特有の傾向を調査した．具体的には，トピックモデルである *BERTopic* を用いて *Issue* から特徴的な単語および文書を抽出し，これらを用いて *Issue* をラベリングすることでソフトウェア開発の傾向を調査した．調査の結果，形式手法を適用したソフトウェア開発とそうでないソフトウェア開発との間で *Issue* の傾向に有意な差が示された．これらの結果は，形式手法を適用したソフトウェア開発に特有の問題が存在することを示唆している．

1. 緒言

ソフトウェア開発において仕様品質の向上がソフトウェア品質の向上に寄与することが報告されており [9]，ソフトウェア仕様が抱える課題として自然言語による記述に起因する曖昧さが指摘されている [8]．形式手法はシステムの仕様を形式的に記述・検証する手法であり，形式的に記述された仕様は特定の性質を容易に検証できる．このような性質から，高品質なソフトウェアの開発手法として形式手法が注目されている [19]．

一方で現在において形式手法は広く普及しているとは言えず，その適用は形式検証が不可欠な領域に限定されている．この要因として学習コストの高さや開発時間の増加などの課題が報告されており [12, 18]，これらの課題を解消する取り組みが行われている [6, 16]．

しかしこれらの課題は形式手法を導入する段階に焦点を当てており，導入した後の開発過程に焦点を当てた課題の報告は少ない．そこで本研究ではソフトウェア開発に形式手法を導入した後の開発過程に発生した問題を分類することで，形式手法を適用したソフトウェア開発に特有の傾向を調査した．

調査には GitHub リポジトリから取得した Issue をデータセットとして，トピックモデルである BERTopic[4] を手法として，それぞれ用いた．データセットに対してトピックモデルを適用することで Issue から特徴的な単語および文書を抽出し，抽出された特徴によって Issue にラベルを付与した．このラベルにより Issue を分類し，形式手法を適用したソフトウェア開発に特有の傾向を調査した．

本研究では形式手法を適用したソフトウェア開発に特有の傾向が存在することを仮説とした．またこの仮説の下で，ソフトウェアに対して報告される動作不良，バグ，機能の追加や改善，保守の 4 項目に関する報告について形式手法を適用して開発されたソフトウェアに特有の傾向が存在すると仮説を立てた．

調査の結果，形式手法を適用したソフトウェア開発とそうでないソフトウェア開発との間で Issue の傾向に有意な差が認められた．またソフトウェアの動作不良に関する報告について，形式手法を適用して開発されたソフトウェアがそうでないソフトウェアより有意に少ないこ

とが認められた．一方でソフトウェアの機能追加や改善を求める報告について，形式手法を適用して開発されたソフトウェアとそうでないソフトウェアとの間に有意な差は認められなかった．これらは形式手法の適用がソフトウェア開発に特有の影響を与えること，ソフトウェアの動作不良を低減すること，ソフトウェアへの機能の追加や改善の要求に与える影響が小さいことをそれぞれ示唆するものである．

2. 背景

2.1. ソフトウェア仕様

ソフトウェア開発において仕様品質の向上がソフトウェア品質の向上に寄与することが報告されている [9]．ソフトウェア仕様が抱える課題として自然言語による記述に起因する曖昧さが指摘されており [8]，自然言語で記述された仕様の曖昧さを低減する手法が提案されている [10, 13]．一方で Yang らの研究 [13] では自然言語で記述された仕様から曖昧さを予測する手法に改善の余地が報告されており，自然言語で記述された仕様から曖昧さを完全に取り除く手法には課題が残されている．そのためソフトウェア仕様から曖昧さを完全に取り除く手法が求められている．

2.2. 形式手法

形式手法はシステムの仕様を形式的に記述・検証する手法である．形式的とは数学的に厳密であることを意味し，形式的に記述された仕様は特定の性質を満たすことを容易に検証できる．2.1 節で述べた Koerner と Brumm の研究 [10] は自然言語で記述された仕様の曖昧さを低減するものであるが，形式的に記述された仕様は曖昧さを低減する必要がない．また既存のソフトウェア開発に対して形式手法を適用する手法として自然言語で記述された仕様を形式的に記述された仕様に変換する手法が提案されている [7]．形式的に記述された仕様は自然言語で記述された仕様より品質に優れることが報告されており [3]，ソフトウェア仕様の品質を向上させる手法として形式手法が注目されている．ソフトウェア仕様の品質の向上はソフトウェア品質の向上に寄与するため，形式手法はソフトウェア工学における重要な題材の 1 つとなっており，その普及が求められている．

2.3. 形式手法の普及

2.2 節で述べた利点にもかかわらず形式手法は一般的なソフトウェア開発手法として普及しているとは言えない．この要因として学習コストの高さが報告されており [17]，形式手法の学習に関する取り組みが行われている [16, 14, 15]．また開発時間の増加もその要因として報告されている [18] が，これらはいずれも形式手法が持つ複雑さに起因する課題であり，形式手法の適用によるソフトウェア開発への影響についての研究は少ない．本研究では形式手法を適用したソフトウェア開発に特有の傾向を調査することで形式手法の普及に寄与することを目指す．

2.4. GitHub Issues

GitHub Issues[1]（Issue）は GitHub[2] リポジトリに提供される機能であり，リポジトリに対して問題を報告するための機能である．これによりソフトウェア開発者は他の開発者および使用者からバグの報告や改善の要求を受け取ることができる．ソフトウェアに報告された問題の分類によってソフトウェア開発を評価した先行研究が存在し [5]，Issue とソフトウェア開発との間に多くの連関が報告されている [1]．本研究では Issue をデータセットとして形式手法を適用したソフトウェア開発に特有の傾向を調査する．

2.5. BERTopic

BERTopic は文書分類に用いられるトピックモデル手法の一種である．トピックモデル手法として LDA(Latent Dirichlet Allocation) が広く利用されているが，LDA はハイパーパラメータの調整の難しさが指摘されている [11]．一方で BERTopic はモデルの学習後にトピックの数を調整できるため，LDA より容易に最適な分類が得られる．また BERTopic は LDA を含む従来のトピックモデル手法より性能に優れることが報告されている [2]．本研究では Issue に対して BERTopic を適用し，ソフトウェアに報告される問題をトピックモデルにより分類することで形式手法の適用によるソフトウェア開発に特有の傾向を調査する．

[1] https://docs.github.com/en/issues
[2] https://github.com

表 1. 形式手法を適用して開発されたリポジトリの一覧

リポジトリ	Star 数	Issue 数	ソフトウェアの種類
seL4	4445	383	マイクロカーネル
hacl-star	1558	222	暗号化ライブラリ
cakeml	882	456	プログラミング言語
CompCert	1721	276	コンパイラ

表 2. 形式手法を適用せず開発されたリポジトリの一覧

リポジトリ	Star 数	Issue 数	ソフトウェアの種類
blog_os	13612	450	オペレーティングシステム
cryfs	1915	419	暗号化ライブラリ
shc	1856	130	コンパイラ
hakyll	2633	485	静的サイトジェネレータ

3. 形式手法を適用したソフトウェア開発の調査

3.1. 調査対象

本研究では GitHub で公開されているリポジトリの Issue をソフトウェアに対して報告される問題として調査した．調査対象とするリポジトリとして形式手法を適用して開発されたものとそうでないものとの双方についてそれぞれ複数を選定し，選定したリポジトリの Issue を取得した．Issue の取得にあたっては次の条件を満たすリポジトリを選定した．ただし形式手法を適用して開発されたものについては，ドキュメントに”formal method”および”formal verification”のような語が含まれており，形式手法を適用した旨が判断できるものを目視により選定した．

1. Star 数が 100 以上である．
2. Issue 数が 100 以上である．
3. ドキュメントが英語で記述されている．

これにより選定した各データセットに属するリポジトリの一覧を表 1，表 2 に，各データセットに属するリポジトリの Issue 数および Star 数の分布を図 1 にそれぞれ示す．

ただしリポジトリの選定にあたっては，形式手法を適用して開発されたものを選定した後に，そうでないものとして形式手法を適用して開発されたものと規模が同程度でありソフトウェアの種類が類似するものを選定した．これは形式手法を適用して開発されたソフトウェアのリポジトリで条件を満たすものとして発見したものが表 1 に示す 4 例のみであったためである．Issue の傾向はソフトウェアの種類に依存する可能性があり，その影響を排除するには様々な種類のソフトウェアについてのリポジトリを選定することが望ましい．しかし形式手法を適用して開発されたソフトウェアのリポジトリで条件を満たすものが少なかったため，傾向の比較からソフトウェアの種類による影響を排除するために種類が類似するものを選定した．ここで選定した 8 例のリポジトリのソフトウェアの種類は表 1，表 2 に示す通りである．

ここでウェルチの検定（両側検定）により，Issue 数および Star 数のそれぞれについて形式手法を適用して開発されたソフトウェアとそうでないソフトウェアとの間に有意な差は認められなかった．したがって形式手法を適用して開発されたソフトウェアおよびそうでないソフトウェアについて，同等の規模のリポジトリをデータセットとして選定できたと考えられる．

3.2. 仮説

本研究では各リポジトリの Issue を 4.3 節で述べる error，bug，requirement，improvement，question，maintenance，other の 8 種類のラベルによって分類した．ここで形式手法を適用して開発されたソフトウェアに特有の傾向があることを仮説とし，加えて次の仮説を立てた．

仮説 1. 形式手法を適用して開発されたソフトウェアはそうでないソフトウェアより error の出現割合が低い．

仮説 2. 形式手法を適用して開発されたソフトウェアはそうでないソフトウェアより bug の出現割合が高い．

仮説 3. 形式手法を適用して開発されたソフトウェアは

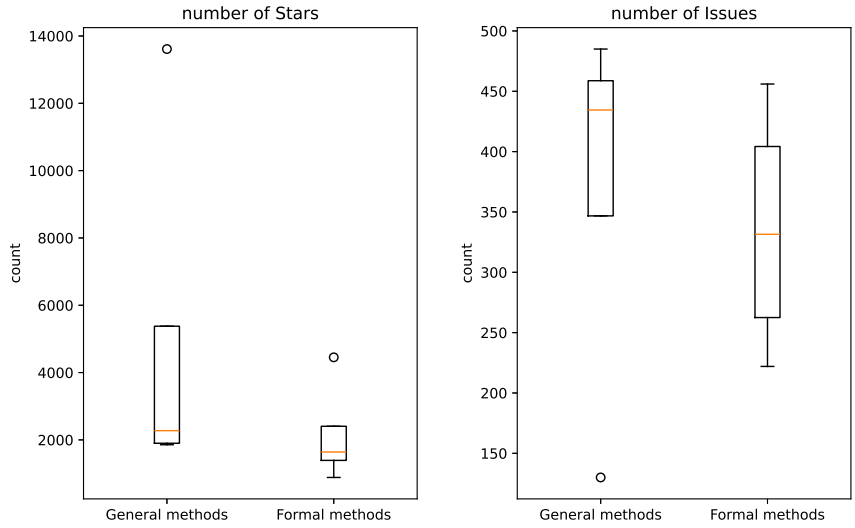

図 1. 形式手法を適用して開発されたリポジトリのデータセットとそうでないリポジトリのデータセットとの Issue 数および Star 数の分布

そうでないソフトウェアより improvement の出現割合が低い．

仮説 4. 形式手法を適用して開発されたソフトウェアはそうでないソフトウェアより maintenance の出現割合が高い．

仮説 1. は error の出現割合についての仮説である．error の出現割合が低いことは形式手法を適用して開発されたソフトウェアに動作不良の報告が少ないことを示唆する．したがって仮説 1. は形式手法を適用して開発されたソフトウェアに動作不良が少ないという仮説である．これは形式手法を適用して開発されたソフトウェアは仕様が形式的に記述・検証されることで動作不良が少ないと考えられるために立てたものである．

仮説 2. は bug の出現割合についての仮説である．bug の出現割合が高いことは形式手法を適用して開発されたソフトウェアにバグの報告が多いことを示唆する．したがって仮説 2. は形式手法を適用して開発されたソフトウェアにバグが多いという仮説である．これは形式手法を適用して開発されたソフトウェアは仕様が形式的に記述・検証されることでバグが発見されやすいと考えられるために立てたものである．

仮説 3. は improvement の出現割合についての仮説である．improvement の出現割合が低いことは形式手法を適用して開発されたソフトウェアに機能の追加や改善を求める報告が少ないことを示唆する．したがって仮説 3. は形式手法を適用して開発されたソフトウェアに機能の追加や改善の要求が少ないという仮説である．これは形式手法を適用して開発されたソフトウェアは仕様が形式的に記述されることで仕様の変更が煩雑であると考えられるために立てたものである．

仮説 4. は maintenance の出現割合についての仮説である．maintenance の出現割合が高いことは形式手法を適用して開発されたソフトウェアに保守に関する報告が多いことを示唆する．したがって仮説 4. は形式手法を適用して開発されたソフトウェアに保守に関する報告が多いという仮説である．これは形式手法を適用して開発されたソフトウェアは仕様が形式的に検証されることで保守が煩雑であると考えられるために立てたものである．

4. 調査手法

形式手法を適用したソフトウェア開発に特有の問題を調査するため，3.1 節で述べたリポジトリの Issue を各々について分類し，その傾向を比較した．調査は以下の手順で行った．

手順 1. 各リポジトリの Issue を取得し，前処理を行う．

手順 2. 前処理が施された Issue を BERTopic により分類する．

手順 3. 手順 2. で得られた分類に対してラベルを付与する．

手順 4. 手順 3. で得られたラベルの出現頻度を，形式手法を適用して開発されたソフトウェアとそうでないソフトウェアとのそれぞれについて集計する．

手順 5. 手順 4. で得られた集計結果をピアソンの χ^2 検定およびウェルチの検定により比較する．

4.1. 手順 1. Issue の取得と前処理

本手順ではリポジトリから取得した Issue を BERTopic により分類するための前処理を行う．前処理の内容は次の通りである．ただしそれぞれの Issue について，最初に投稿された内容のみを調査対象としており，その内容に連なって投稿されたコメントは調査対象外とした．

1. Issue の title 要素および body 要素から URL，メールアドレス，コードブロック，記号を除去する．
2. Issue の title 要素および body 要素を結合し，単一のテキストとする．

BERTopic は一般的な文章分類においては前処理を必要としない．しかし本研究で取得した Issue は Markdown[3]形式で記述されており，またソフトウェアに対する報告であることからエラーメッセージやコードスニペットを含むことがある．これらの情報は文章の意味に影響を与えないが，BERTopic による分類に影響を与える可能性があるため，前処理によって除去した．

4.2. 手順 2. BERTopic による分類

本手順では各リポジトリについて前処理を行った Issue を BERTopic により分類する．この結果として，分類およびその分類に属する Issue 数，また各分類を代表する単語および文書が得られる．ここで分類を代表する単語とは各分類に含まれる文書の中で最もその分類を特徴づけると考えられる単語であり，本手順においてはそれぞれの分類に対して 10 単語を抽出した．また分類を代表する文書とは各分類に含まれる文書の中で最もその分類を特徴づけると考えられる文書であり，本手順においてはそれぞれの分類に対して 3 文書を抽出した．

[3] https://daringfireball.net/projects/markdown

表 3. Issue の分類に付与したラベルとその意味

ラベル	意味
error	動作不良
bug	明確にバグと判断される動作不良
requirement	ソフトウェアに対する要求
improvement	特に機能追加・改善の要求
question	ソフトウェアに関する質問
maintenance	保守に関する報告
other	いずれにも分類されないもの

4.3. 手順 3. ラベルの付与

本手順では手順 2. で得られた分類に対してラベルを付与する．付与するラベルは Hall らの研究 [5] および Bissyandé らの研究 [1] において用いられたものを参考に，本研究で取得した Issue の分類に適切であると考えられるものを設定した．付与したラベルとその意味を表 3 に示す．

各ラベルは手順 2. で得られた分類において，その分類を代表する文書を参考にして付与した．ただし BERTopic による分類では外れ値が発生するため，外れ値に対しては一律に other を付与した．

4.4. 手順 4. ラベルの集計

本手順では手順 3. で得られたラベルの出現回数を，形式手法を適用して開発されたソフトウェアとそうでないソフトウェアとについて集計する．まず各リポジトリにおいて Issue の分類に付与されたラベルから各ラベルに属する Issue の出現回数を算出し，これを各リポジトリにおけるラベルの出現回数とした．次に形式手法を適用したソフトウェアとそうでないソフトウェアとのそれぞれのデータセットについて，各リポジトリにおけるラベルの出現回数を合算し，これから各ラベルの出現割合を算出した．本研究では各データセットについて算出したラベルの出現回数および出現割合を Issue の傾向として扱う．

4.5. 手順 5. 統計分析

本手順では手順 4. で得られた Issue をピアソンの χ^2 検定により，各ラベルの出現割合をウェルチの検定により，それぞれ比較する．ここでピアソンの χ^2 検定は分類されたデータの連関を検定するノンパラメトリック検定であり，ウェルチの検定は平均値の差を検定するパラメトリック検定である．本研究においては手順 4. で得られたラベルの出現割合について，形式手法を適用して開発されたソフトウェアとそうでないソフトウェアとの間でピアソンの χ^2 検定を行う．また各ラベルの出現割合について，形式手法を適用して開発されたソフトウェアとそうでないソフトウェアとの間でウェルチの検定（片側検定）を行う．

ただし検定は other のラベルについての情報を除いて行った．これは BERTopic による分類における外れ値に other を付与しており，どの分類にも属さない Issue のみが other のラベルを持つとは限らないためである．

5. 調査結果

形式手法を適用して開発されたソフトウェアとそうでないソフトウェアとにおけるラベルの出現割合の比をピアソンの χ^2 検定により有意水準 5%で比較した結果，ラベルの出現割合に有意な差が認められた．ここで 4.3 節，4.4 節の手順で得た各リポジトリにおけるラベルの出現回数を表 4，表 5 に示す．また表 4，表 5 のそれぞれについて合計の列から算出したラベルの出現割合を表 6 に示す．これが 4.5 節の手順で検定により比較した対象である．

また 3.2 節で立てた仮説に用いる error，bug，improvement，maintenance のラベルについて，ウェルチの検定（片側検定）により有意水準 5%で出現割合の大小関係をそれぞれ比較した．その結果を次に示す．

1. error の出現頻度は形式手法を適用して開発されたソフトウェアがそうでないソフトウェアより有意に低いことが認められた．
2. bug の出現頻度は形式手法を適用して開発されたソフトウェアがそうでないソフトウェアより有意に高いことが認められなかった．
3. improvement の出現頻度は形式手法を適用して開発されたソフトウェアがそうでないソフトウェアより有意に低いことが認められなかった．
4. maintenance の出現頻度は形式手法を適用して開発されたソフトウェアがそうでないソフトウェアより有意に高いことが認められなかった．

表 4. 形式手法を適用して開発されたソフトウェアの各リポジトリにおけるラベルの出現回数

ラベル	seL4 (回)	hacl-star (回)	CakeML (回)	CompCert (回)	合計 (回)
error	108	70	0	18	196
bug	37	0	0	49	86
requirement	0	0	96	0	96
improvement	30	21	173	32	256
question	0	0	0	0	0
maintenance	15	41	0	74	130
other	193	90	186	103	572

表 5. 形式手法を適用せず開発されたソフトウェアの各リポジトリにおけるラベルの出現回数

ラベル	blog_os(回)	cryfs (回)	shc (回)	hakyll (回)	合計 (回)
error	223	154	58	105	540
bug	0	15	0	0	15
requirement	0	0	0	0	0
improvement	66	68	0	83	217
question	0	0	0	50	50
maintenance	0	0	0	0	0
other	161	182	72	247	662

また有意な差が認められなかった仮説 2.，仮説 3.，仮説 4. について，それぞれと逆の事象を示す次の仮説を立てた．

仮説 2′. 形式手法を適用して開発されたソフトウェアはそうでないソフトウェアより bug の出現割合が低い．

仮説 3′. 形式手法を適用して開発されたソフトウェアはそうでないソフトウェアより improvement の出現割合が高い．

表 6. 形式手法を適用して開発されたソフトウェアとそうでないソフトウェアにおけるラベルの出現割合

ラベル	形式手法を適用 (%)	形式手法を適用せず (%)
error	14.67	36.39
bug	6.44	1.01
requirement	7.19	0.00
improvement	19.16	14.62
question	0.00	3.37
maintenance	9.73	0.00
other	42.81	44.61

仮説 4′. 形式手法を適用して開発されたソフトウェアはそうでないソフトウェアより maintenance の出現割合が低い.

これらの仮説についてウェルチの検定（片側検定）により有意水準 5%で出現割合の大小関係をそれぞれ比較した．その結果を次に示す.

1. bug の出現頻度は形式手法を適用して開発されたソフトウェアがそうでないソフトウェアより有意に低いことが認められなかった.
2. improvement の出現頻度は形式手法を適用して開発されたソフトウェアがそうでないソフトウェアより有意に高いことが認められなかった.
3. maintenance の出現頻度は形式手法を適用して開発されたソフトウェアがそうでないソフトウェアより有意に低いことが認められなかった.

ここでウェルチの検定による比較に用いた各データセットにおけるラベルの出現割合を表 7，表 8 にそれぞれ示す.

6. 議論

本研究では形式手法を適用したソフトウェア開発に特有の傾向を，ソフトウェアのリポジトリに対して報告された Issue に着目して調査した．その結果，形式手法を適用されたソフトウェアとそうでないソフトウェアとの

表 7. 形式手法を適用して開発されたソフトウェアにおけるラベルの出現割合

ラベル	seL4 (%)	hacl-star (%)	CakeML (%)	CompCert (%)
error	28.20	31.53	0.00	6.52
bug	9.66	0.00	0.00	17.75
requirement	0.00	0.00	21.10	0.00
improvement	7.83	9.46	38.02	11.59
question	0.00	0.00	0.00	0.00
maintenance	3.92	18.47	0.00	26.81
other	50.39	40.54	40.88	37.32

表 8. 形式手法を適用せず開発されたソフトウェアにおけるラベルの出現割合

ラベル	blog_os(%)	cryfs (%)	shc (%)	hakyll (%)
error	41.30	36.80	44.62	21.65
bug	0.00	3.58	0.00	0.00
requirement	0.00	0.00	0.00	0.00
improvement	14.67	16.23	0.00	17.11
question	0.00	0.00	0.00	10.31
maintenance	0.00	0.00	0.00	0.00
other	35.78	43.44	55.38	50.93

間で Issue の傾向に有意な差が認められた．このことは形式手法を適用したソフトウェア開発に特有の傾向が存在することを示唆している．また 3.2 節で立てた仮説について，仮説 1. は支持されたが仮説 2.，仮説 3.，仮説 4. は支持されなかった.

以下でそれぞれの仮説について考察する.

仮説 1. が支持されたことは，形式手法を適用して開発されたソフトウェアはそうでないソフトウェアより動作不良の報告が少ないことを示唆する．したがって形式手法の適用がソフトウェアの動作不良を低減すると考えられる.

仮説 2. が支持されなかったことは，形式手法を適用して開発されたソフトウェアはそうでないソフトウェアよりバグの報告が多いとは言えないことを示唆する．各

データセットにおけるラベルの出現割合に着目すると，双方ともに半数以上のリポジトリにおいて bug の出現割合が 0.0%であった．一般にソフトウェアがバグを含まないことは考え難いため，bug が付与されるべき Issue に他のラベルが付与されていると考えられる．ここで bug が付与されるべき Issue に error が付与されていることが考えられる．Issue における動作不良の報告はその後の議論によって bug と判断されることがあるが，本研究では Issue を title 要素および body 要素に着目して分類しており，そのような議論を反映していない．したがって形式手法の適用がソフトウェアにおけるバグの発見を容易にするとは言えないが，Issue に付随する議論を含めた今後の調査が必要であると考えられる．

仮説 3. が支持されなかったことは，形式手法を適用して開発されたソフトウェアはそうでないソフトウェアより機能の追加や改善の要求が少ないとは言えないことを示唆する．したがって形式手法の適用がソフトウェアへの機能の追加や改善の要求に与える影響は小さいと考えられる．

仮説 4. が支持されなかったことは，形式手法を適用して開発されたソフトウェアはそうでないソフトウェアより保守に関する報告が多いとは言えないことを示唆する．ここで形式手法を適用せず開発されたソフトウェアにおけるラベルの出現割合は全てのリポジトリにおいて 0.0%であり，形式手法を適用して開発されたソフトウェアに保守に関する報告が多いように思われる．一方で各データセットにおけるラベルの出現割合についての検定結果は maintenance の出現割合について有意な差を示さなかった．この一因としてデータセットにおけるリポジトリの少なさが考えられる．したがって形式手法の適用がソフトウェアにおける保守に与える影響については今後の調査が必要であると考えられる．

7. 今後の展望

本研究では形式手法を適用したソフトウェア開発に特有の傾向を，リポジトリに対して報告された Issue に着目して調査した．その結果，形式手法を適用したソフトウェア開発とそうでないソフトウェア開発との間で Issue の傾向に有意な差が認められた．この差は形式手法の適用がソフトウェア開発に与える影響を示唆するものである．一方で個々のラベルについては error のみに有意な差が認められた．この差は形式手法の適用がソフトウェアの動作不良を低減することを示唆するものである．

ここで本研究が抱える課題として次が挙げられる．

課題 1. BERTopic による分類に多くの外れ値が含まれている．

課題 2. OSS と産業用ソフトウェアとの差異を考慮していない．

課題 3. リポジトリの開発段階を考慮していない．

課題 1. は BERTopic による分類において，多くの Issue が外れ値と判定され，分類が推定されなかったことを指す．本研究では Issue の title 要素および body 要素をトピックモデルにより分類するために BERTopic を用いたが，その結果として多くの Issue が外れ値と判定され，分類が推定されなかった．本研究ではこの外れ値に other のラベルを付与したため，一部の Issue について適切なラベルが付与できていないと考えられる．BERTopic は外れ値を削減する手法を複数持つため，今後はそれらの手法を用いて外れ値を削減することが望ましいと考えられる．

課題 2. は本研究で対象を OSS(Open Source Software) に限定しているために産業用ソフトウェアについての調査が行われていないことを指す．産業用ソフトウェアの開発には OSS の開発とは異なる事情が存在する．例えば OSS においては開発者は形式手法を独学するが，産業用ソフトウェアにおいては開発者は形式手法について教育を受ける，などである．そのため OSS と産業用ソフトウェアとでは形式手法の適用による影響が異なると考えられる．

課題 3. はリポジトリの開発段階を考慮していないことを指す．本研究で調査対象としたリポジトリは Star 数および Issue 数のみを選定条件としたが，Issue はソフトウェアの開発段階によって傾向が変化すると報告されている [1]．そのためバージョンや開発期間など，リポジトリの開発段階を条件に加えてリポジトリを選定することが望ましいと考えられる．ただし GitHub において公開されているリポジトリで形式手法を適用して開発されたものは極めて少なく，この条件を加えてリポジトリを選定することは難しい．

今後は以上の課題を解消することにより，形式手法を適用したソフトウェア開発に特有の傾向をより詳細に調査することが望ましいと考えられる．

8. 結言

本研究ではソフトウェアに対して報告された問題としてソフトウェアのリポジトリに対して報告された Issue に着目し，形式手法を適用したソフトウェア開発に特有の傾向を調査した．その結果，形式手法を適用したソフトウェア開発とそうでないソフトウェア開発との間で Issue の傾向に有意な差が認められた．このことは形式手法を適用したソフトウェア開発に特有の傾向が存在することを示唆するものである．

またソフトウェアの動作不良に関する報告について，形式手法を適用して開発されたソフトウェアがそうでないソフトウェアより有意に少ないことが認められた．このことは形式手法の適用がソフトウェアの動作不良を低減することを示唆するものである．

一方でソフトウェアの機能追加や改善を求める報告について，形式手法を適用して開発されたソフトウェアとそうでないソフトウェアとの間に有意な差は認められなかった．このことは形式手法の適用がソフトウェアへの機能の追加や改善の要求に与える影響が小さいことを示唆するものである．

この他にバグ，保守に関する報告についても形式手法を適用して開発されたソフトウェアとそうでないソフトウェアとの間に有意な差は認められなかった．これらは Issue に付随する議論の内容を考慮していないことやサンプル数の少なさ，トピック分析による外れ値に起因している可能性が考えられるため，今後の調査が求められる．

参考文献

[1] T. F. Bissyandé, D. Lo, L. Jiang, L. Réveillere, J. Klein, and Y. Le Traon. Got issues? Who cares about it? A large scale investigation of issue trackers from GitHub. In *In Proceedings of the 24th IEEE International Symposium on Software Reliability Engineering (ISSRE)*, pp. 188–197, 2013.

[2] R. Egger and J. Yu. A topic modeling comparison between LDA, NMF, Top2Vec, and BERTopic to demystify twitter posts. *Frontiers in sociology*, 7:1–16, 2022.

[3] F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami. An automatic quality evaluation for natural language requirements. In *In Proceedings of the 7th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ)*, Vol. 1, pp. 4–5, 2001.

[4] M. Grootendorst. Bertopic: Neural topic modeling with a class-based TF-IDF procedure. *arXiv preprint arXiv:2203.05794*, 2022.

[5] T. Hall, A. Rainer, N. Baddoo, and S. Beecham. An empirical study of maintenance issues within process improvement programmes in the software industry. In *In Proceedings of 17th IEEE International Conference on Software Maintenance (ICSM)*, pp. 422–430, 2001.

[6] M. Huisman, D. Gurov, and A. Malkis. Formal methods: From academia to industrial practice: A travel guide. Workingpaper, Cornell University, 2020.

[7] M. Ilieva and O. Ormandjieva. Automatic transition of natural language software requirements specification into formal presentation. In *In Proceedings of the 10th International Conference on Application of Natural Language to Information Systems (NLDB)*, pp. 392–397, 2005.

[8] E. Kamsties. *Understanding Ambiguity in Requirements Engineering*, pp. 245–266. Springer Berlin Heidelberg, 2005.

[9] E. Knauss, C. Boustani, and T. Flohr. Investigating the impact of software requirements specification quality on project success. In *In Proceedings of the 10th International Conference on Product Focused Software Process Improvement*, Vol. 32, pp. 28–42, 2009.

[10] S. J. Körner and T. Brumm. Natural language specification improvement with ontologies. *International Journal of Semantic Computing*, 3(4):445–470, 2009.

[11] A. Panichella. A systematic comparison of search-based approaches for LDA hyperparameter tuning. *Information and Software Technology*, 130:1–20, 2021.

[12] A. Reid, L. Church, S. Flur, S. de Haas, M. Johnson, and B. Laurie. Towards making formal methods normal: meeting developers where they are. *arXiv preprint arXiv:2010.16345*, 2020.

[13] H. Yang, A. De Roeck, V. Gervasi, A. Willis, and B. Nuseibeh. Analysing anaphoric ambiguity in natural language requirements. *Requirements engineering*, 16:163–189, 2011.

[14] 荒木. ソフトウェア開発現場への形式手法導入形式手法適用の実経験から得られた知見. *SEC journal*, 6(2):104–107, 2010.

[15] 荒木. 形式手法導入のための産学連携 PBL の活用. *SEC journal*, 7(4):177–182, 2011.

[16] 大西, 吉村, 阿部, 稲川, 山本, 堀. 工業高等専門学校の学生に対する形式手法 b-method の学生実験の実践. 情報処理学会論文誌, 61(4):863–883, 2020.

[17] 栗田. 形式手法の実践に対してよく尋ねられる質問とその回答モバイル FeliCa の開発における形式仕様記述を通して. *SEC journal*, 7(1):34–39, 2011.

[18] 喜多村, 上原他. F*言語による MQTT パケットパーサの開発と安全性評価. 研究報告セキュリティ心理学とトラスト (SPT), 2021(8):1–7, 2021.

[19] 青木. 車載システム開発における形式手法実践の現状と課題. システム／制御／情報, 62(4):134–140, 2018.