版管理リポジトリに含まれる設定ファイルに対する命名規則の調査 ー命名規則の共通認識形成に向けてー

忠谷 晃佑[1]　柗本 真佑[1]　楠本 真二[1]

概要：近年のソフトウェア開発では版管理リポジトリを中心とした開発形態が広く採用されている．この開発形態ではソースコードやドキュメントだけでなく機械が読み込むための設定ファイル群が版管理リポジトリで管理される．しかし版管理リポジトリ中心の開発形態ではリポジトリ内の各種ファイルの役割の判別のしやすさが損なわれる．その理由として多種多様な設定ファイルがリポジトリルート直下に多数配置される点や，設定ファイルの命名に共通のルールが存在していない点が挙げられる．本研究の目的は版管理リポジトリ上の設定ファイル名に対する共通認識の形成にある．共通認識の形成には現状の設定ファイルに対する命名に関して知見を得る必要がある．調査においては設定ファイルに着目したリポジトリマイニングを実施し，そのファイル名を収集する．調査の結果，版管理リポジトリでは現状，設定ファイルに対応するツール名を含んだ名前を付ける以外に共通の認識が広く浸透していないとわかった．

1. はじめに

近年のソフトウェア開発では，版管理リポジトリを中心とした開発形態が広く採用されている [1]．この開発形態では，開発プロセス中に得られるあらゆる成果物を版管理リポジトリで管理する．リポジトリにはソースコードだけでなく，その検証のためのテストコードや，利用者のためのドキュメント類も存在する [2]．さらには，ビルド方法を記載したビルドスクリプトや，CI/CD（Coninuous Integration/Continuous Deployment）環境での処理を記載したワークフローファイルなど，機械が読み込むための各種ファイル群も同様に管理される．リポジトリ内で開発に係る様々な役割を持つデータを管理することで，プロダクトを中心としたコミュニティに属する様々なステークホルダーに対して一元的なデータアクセスを実現する．

リポジトリを中心とした開発形態では，リポジトリ内のファイルの識別性が低下すると考えられる．本稿では，識別性という言葉を各種ファイルの役割の判別のしやすさという意味で用いる．例えば，ある開発者が CI の設定ミスを修正する際には，どのファイルがビルドスクリプトなのか，どのファイルがワークフローファイルなのか等を識別する必要がある．本稿ではリポジトリ内に含まれるファイルのうち，ソースコードやドキュメントを除いた設定ファイル群に着目する．様々な開発支援ツールが存在する現在では，多種多様な設定ファイルがリポジトリ内で管理される．また設定ファイルはリポジトリのルート直下に設置するという暗黙のルールが存在するため，特に識別性が低下する傾向にある．我々の事前調査によると，リポジトリの直下には平均 34.3 個のファイルとフォルダが存在しており，この存在が識別性低下の要因となりうる．

リポジトリ内の多数のファイルを許容しつつ，識別性を確保するためには，ファイルの命名規則が最も重要な要素であると考える．ファイル名はリポジトリを俯瞰する際に最初に目にする要素であり，適切なファイル名の付与は，ファイルが担う役割の判別を容易にする．例えば，拡張子はファイルの書式を端的に表す一種の識別子であり，共通認識のある命名規則であるともいえる．不適切な拡張子はファイル識別性の低下に繋がるだけでなく，テキストエディタ上でのシンタックスハイライトの混乱にも繋がる．多種多様な設定ファイルの役割を識別するためには，拡張子以外にも設定ファイルに対応するツール名やその役割などを記載するべきだと考える．例えば，ビルドツール Gradle 専用のビルド設定ファイルである build.gradle の場合，build が設定を行うドメイン，gradle はツール名であり，自己完結な名前ではあるものの，拡張子は不適切とも考えられる．`build.gradle` は Groovy で記述されるファイルであるため，その書式を拡張子で表現するには `build.gradle.groovy` がより適切であると考えられる．

UNIX 系 OS における一つの有名な命名ルールとしてホーム直下の設定ファイルを名前の先頭にドットを持つドットファイルや名前の先頭にドットを持つドットディレ

[1] 大阪大学大学院情報科学研究科 大阪府吹田市

クトリ内のファイルとして管理する慣習があるが，これが版管理リポジトリにおいても共通の認識として存在しているかも不明である．版管理リポジトリ上でもその考えが採用されているファイルが存在し，Git の動作を設定する `.gitignore` や `.gitattribute` などがリポジトリルートにおいてドットファイルとして管理されている．一方でビルド設定を記載した `build.gradle` はドットファイルではない，といったように版管理リポジトリ上では設定ファイルの命名ルールは異なっている．CI 環境設定など版管理リポジトリに特有のファイルが存在することからも，両者の間で命名の認識は異なる場合が多いとも考えられる．

本研究の目的は，リポジトリに含まれる設定ファイルの命名規則に対する共通認識の形成である．そのために設定ファイルに着目してマイニング調査を実施する．調査にあたっては版管理リポジトリ上でどれほどの共通認識が形成されているかを分析するため，以下 3 つの Research Questions（RQ）を設定する．

RQ1：設定ファイルはドットファイルとして管理されているか
RQ2：設定ファイルにはどのような名前がつけられているか
RQ3：設定ファイルの記述形式は拡張子と一致するか

調査においては 140 個のリポジトリから 618 個の設定ファイル名を収集し，これらを分析する．

本研究の貢献は以下の通りである．

1．初めて版管理リポジトリ上の設定ファイルに注目したマイニング調査を実施した．版管理リポジトリ上の設定ファイルに対して，設定ファイルに対応するツールや設定ファイルが利用されるソフトウェア開発プロセスを限定せずに複数リポジトリにわたって実施されたマイニング調査は我々の知る限り存在せず，本研究が初の調査である．

2．ドットファイルと設定ファイルの関係を洗い出し知見を得た．調査の結果，設定ファイル 37%のみがドットファイルとして管理されていると判明した．さらにこの原因を考察し，リポジトリでは設定ファイルの中でもビルドに関連するファイルが非ドットファイルとして管理されていること，また版管理リポジトリにおいてもドットファイルは UNIX 系 OS 上と同様，隠しファイルとして扱い，一部ステークホルダーの関心から除外することが共通認識があると考えられた．この知見は，ファイル識別性の向上に向けた共通認識の形成に貢献しうる．

3． 版管理リポジトリに含まれる設定ファイルの命名に対する共通認識について知見を得た．本研究における調査の結果，版管理リポジトリにおいて設定ファイルには対応するツール名を含む名前を付けるという共通認識のみが存在するという知見が得られた．得られた知見は，ツールの開発者やリポジトリの管理者に対して設定ファイルの命名に対する提案に繋がり，ファイル識別性の向上に貢献する．

2. 準備

2.1 版管理リポジトリ中心の開発

近年のソフトウェア開発プロジェクトは版管理リポジトリを中心に行われる [1]．プロジェクトの成果物は，全て版管理リポジトリ上で管理され，プロジェクトのステークホルダーはそれぞれの目的を達成するために，版管理リポジトリ上のそれぞれ異なるファイルにアクセスする．

例えば，プロダクトに新たな機能を追加したいと考える開発者は，`src/`といったディレクトリにアクセスしてソースコードを閲覧または編集したり，`contribute.md` を読んで開発環境の構築方法や Pull Request の作成ルールなどを把握すると予想できる [3]．他にも，プロダクトを利用したいと考える利用者は，`readme.md` を読んでインストールやその後の利用方法を把握すると予想できる．

このように，ステークホルダーによって目的達成のためにアクセスしたいリポジトリ上のファイルは様々だと考えられる．したがって，リポジトリ上でファイルが果たす役割の把握しやすさは重要である．

2.2 ファイル識別性

本稿ではファイル識別性という概念を導入する．これは，版管理リポジトリが持つ性質の一つで，リポジトリ上のファイルの役割の把握しやすさを指す．ファイル識別性の高いリポジトリは 2.1 節で述べたようにステークホルダーにとって便利なだけでなく，新規の開発者にとって参入しやすい環境であるとも考えられる [4]．

リポジトリ識別性を向上する工夫の一つとしてプロジェクトに導入する様々なツールの標準ディレクトリ構成の利用があげられる．Java のビルドツールとして広く用いられる Maven は，それを用いるプロジェクトに対して標準ディレクトリ構成[*1] を推奨している．標準ディレクトリ構成の利用により，Maven を利用するプロジェクトにリポジトリ上のファイルと名前，またディレクトリ構造に共通の認識を与える効果がある．

また，そのファイルが持つ役割を表現する要素を十分に含むファイル名を設定するのも重要である．2.1 節で提示した具体例から予測できるように，ステークホルダーは目的を達成するために閲覧すべきファイルを名前で判断していると考えられる．版管理リポジトリで管理されるファイルの一つであるソースコードについては，一定の命名規則がある．Anquetil と Letbridge はソースコードの名前はいくつかの要素に分解できそれぞれの要素はそのファイルに記述されたソースコードが実現する機能や利用するデータといった属性を表現するという知見を述べている [5]．ま

[*1] https://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html

た，開発組織が発達しているほど，命名規則や含まれる要素がよく定まっているとも述べている [5]．加えて，ファイル名はそのファイル内容を端的に表しているべきという共通認識も存在するため [6]，適切なファイルの命名はファイル識別性の向上に非常に重要な要素である．

2.3 リポジトリ上の設定ファイルとその問題点

近年の版管理リポジトリにおいてはプロダクトのソースコードやドキュメントに加えて [2]，様々なアプリケーションの設定ファイルも管理されている．以降，本稿では設定ファイルをソフトウェアの動作を変更するためのパラメータを記述したテキストファイルと定義する．例えばソースコードの静的解析ツール [7] や CI/CD 環境の導入が開発プロジェクトに進んでおり [8]，これらに伴いその設定ファイルがリポジトリに増加していると考えられる．

しかし，この設定ファイルの増加が，リポジトリのファイル識別性の低下につながっていると考える．設定ファイルの名前は対応するツールによってデフォルト名が定められている場合が多く，プロジェクトごとの命名による独自なファイル識別性改善が難しいためである．さらに，設定ファイルの多くがツールの都合でリポジトリルートに設置され，ルートのファイル数が肥大していることもファイル識別性の低下につながると考える．

2.4 設定ファイルの命名に対する共通認識の形成

我々はファイル識別性向上の観点から，版管理リポジトリ上の設定ファイルの命名に一定の共通認識を与えるべきだと考える．既に存在する共通認識として UNIX 系 OS の設定ファイルの命名に対するものがある．この共通認識と同様に，版管理リポジトリに含まれる設定ファイルの自己完結な名前を目指した共通認識を考察する．

ドットファイルとしての管理

UNIX 系 OS における一つの共通認識としてアプリケーションの設定ファイルをドットファイルとしたり [9], [10]，ドットディレクトリ内に配置したり [11] する文化が存在する．

ドットファイルおよびドットディレクトリとはファイル名の先頭が「.」であるようなファイルやディレクトリであり，隠しファイルとして扱われ，ユーザーのアクセス機会が少ないファイルを ls コマンド等において非表示にできる．

我々は設定ファイルはドットファイルであるというプラクティスを版管理リポジトリ上でも適用すべきであると考える．設定ファイルはユーザーが明示的にアクセスする機会は少ないが [12]，アプリケーションに設定を適用するために必要な点が，隠しファイルの目的と一致するためである．プロジェクトの開発者やプロダクトの利用者からプロダクトの開発に用いるツールの設定ファイルを隠すと編集や閲覧すべきファイルの相対的な強調となりファイル識別性向上につながると考えられる．

対応するツール名を含む命名

設定ファイルの命名に対しては設定ファイルが定義するパラメータを読み込むツール名を含ませるべきであると考える．ツール名が名前に含まれていれば，ファイルの使用目的が明確になりファイル識別性の向上につながる．さらに，ファイルの記述時に参照すべきドキュメントも明らかになる．

設定ファイルであると示す命名

そのファイルが設定ファイルであると表現する場合，その旨を表現するファイル名も求められると考える．ファイル識別性の向上にはそのファイルが担う役割の記述が不可欠なためである．

フォーマットに一致する拡張子の付加

設定ファイルのフォーマットに一致する拡張子を付加すべきであると考える．つまり，ファイル名の末尾をピリオドで区切り，ファイルフォーマットを表現する要素を持たせるべきであると考える．設定ファイルはその多くがパラメータを構造化して記述したテキストファイルである．また，多くの OS では拡張子によってファイルの形式を判別する．そのため記述形式に対応する拡張子を付加すると，多くの環境でシンタックスハイライトなどのファイル内容の把握に役立つ支援を受けられるため，ファイル識別性の向上に繋がるといえる．

3. Reserach Questions

本研究の目的は版管理リポジトリのファイル識別性向上にある．この目的の達成には設定ファイルの命名における共通認識の存在が望ましい．そのために版管理リポジトリで管理されている設定ファイルの現状について調査する．調査のために以下の Research Questions を定める．

RQ1：設定ファイルはドットファイルとして管理されているか

RQ1 では版管理リポジトリ上のドットファイルは設定ファイルであるか、また設定ファイルはドットファイルであるかを調査する．2.4 節で述べた通り，設定ファイルのドットファイルとしての管理はファイル識別性向上に寄与すると考えられる．そこで設定ファイルとドットファイルの集合はどれほど一致しているか現状を調査する．

RQ2：設定ファイルにはどのような名前がつけられているか

RQ2 では設定ファイルがどのようなファイル名で管理されているか，調査する．以降本稿におけるファイル名は当該ファイルのフルパスをパス区切り文字に従って分割した時に末尾に得られる部分文字列とする．適切なファイル名の設定はファイル識別性の改善につながる．我々はソースコードなどと同様，設定ファイルについてもファイルが持つ特性を名前に表現すべきであると考える．そこで現状，版管理リポジトリで管理されている設定ファイル名にファイルの書式を適切に表すフォーマット名やファイルが設定するツールの名前，また記述内容がツールの設定であると推測できる語が含まれているか調査する．

RQ3：設定ファイルの記述形式は拡張子と一致するか

RQ3 では設定ファイル名と内容の一致を調査する．ファイル名とファイル内容の一致はファイル識別性の一つである．本研究では特にファイルのフォーマットとファイル拡張子の一致に着目し，現状どれほど一致しているかを調査する．

4. 調査手法

4.1 RQ1 の調査

ドットファイルと設定ファイルの現状調査

本調査の目的は現状の版管理リポジトリにおいてドットファイルの集合と設定ファイルの集合がどの程度一致しているかを確かめる点にある．調査においてはドットファイルと設定ファイルを版管理リポジトリから抽出しそれぞれのユニークなファイル名集合を作成する．本調査においてファイル名の集合を用いる理由は注目する設定ファイルの名前は，対応するツールによって定められている場合が多く同じ名前のファイルが多数存在すると考えられるためである．

はじめにファイルを抽出するリポジトリを選定する．対象リポジトリの選定においては 14 種類の異なる主開発言語のリポジトリをそれぞれ 10 個ずつ用いる．それぞれ異なる言語を選択した理由は多様な設定ファイルを収集するためである．開発に用いられる言語によって異なるエコシステムが形成されており，それらを構成するツールがそれぞれ異なる．したがって主たる開発言語が異なるリポジトリを対象とし多様なファイルを収集する．言語ごとのリポジトリはそれぞれ 10 個ずつスター数の多いリポジトリを選定する．スターの多いリポジトリは多くのコミュニティから支持を得ており，評価が高いと考えられるためである．なお，リポジトリの選定においてはサンプルソースコードの集積リポジトリや教育もしくは実験目的のリポジトリを目視にて除外している．

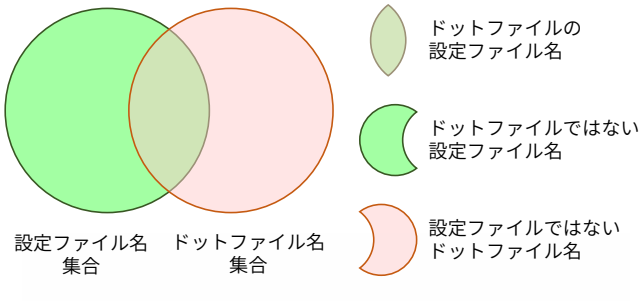

図 1: 分析するファイル名の集合

次に対象リポジトリのルートディレクトリからファイルを抽出する．ファイル抽出においては，はじめに全ファイルの名前を収集し，それをユニークな名前のみの集合にする．収集したファイル名の分類においては.c や.rs といったコンパイルが必要なファイルや.md などのドキュメント，またテキストファイルではないファイルなど，明らかに本稿における設定ファイルの定義に一致しないファイルをあらかじめ除外し，目視でファイルを調査する．調査においてはそれぞれのファイルについてファイル名やコミットメッセージを参照し設定ファイルであるかを判定する．ディレクトリについても内部に配置されたファイルやリポジトリ管理者が提供するドキュメントによって設定ファイルであるかを判定する．また，先頭がドットであるようなファイル名も同様に収集する．

以上の調査手順により収集したファイル名は 図 1 に示す設定ファイル名の集合 A とドットファイル名の集合 B にそれぞれファイル名を分類できる．その後，それぞれのファイル名集合について数を分析する．ドットファイルとして管理される設定ファイル名 $A \cap B$ の数とドットファイルではない設定ファイル名 $A \cap \bar{B}$ および設定ファイルではないドットファイル名 $\bar{A} \cap B$ の数を調査して UNIX 系 OS における設定ファイルと版管理リポジトリ上の設定ファイルとの命名に対する共通認識の差異を明らかにする．

4.2 RQ2 の調査

ファイル名に含まれる語の調査

RQ2 に対する調査の目的はツール名やフォーマットを表す語，また設定ファイルであると表す語が含まれている設定ファイル名がどれほど存在するかを確かめることにある．本調査では設定ファイルにつけられた名前に含まれる要素を分析する． 図 1 に緑で示した設定ファイル名の集合 A を分析対象として利用する．

分析においてはそれぞれのファイル名についてそれらを構成する要素を抽出し，それらがどのような語であるかを分類する．要素の抽出においてはデリミタのない文字列を辞書に基づいて単語に分割するライブラリのひとつである Word Ninja [*2] を利用する．本調査で用いる辞書には一般

[*2] https://github.com/keredson/wordninja

表 1: ファイル名について調査する要素

ID	名前	説明
T	ツール名	パラメータを適用するツールの名前
S	役割	ファイルの利用目的が設定であると示す語
F	フォーマット	ファイルの記述形式
O	その他	その他の語

表 2: ファイル名に含まれる要素

要素									合計
ツール名 (T)	-	-	-	-	o	o	o	o	
役割 (S)	-	-	o	o	-	-	o	o	
フォーマット (F)	-	o	-	o	-	o	-	o	
#ファイル名	101	46	8	4	191	164	43	61	618

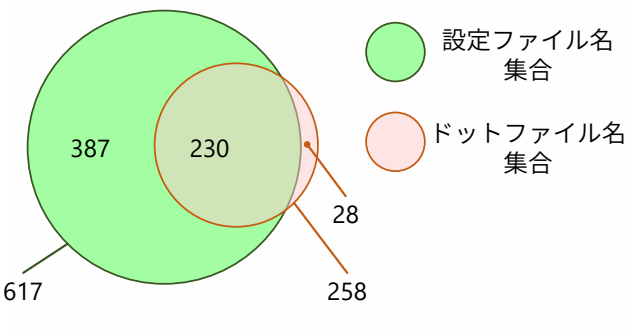

図 2: ドットファイル名と設定ファイル名の数

的な英単語に加え，RQ1 に対する調査で得た各設定ファイルに対応するツール名や調査対象とするリポジトリ名，またソースコード記述時に特によく利用される略語[*3] を用いる．抽出した要素についてそれぞれどのような要素であるかを確認し 表 1 にしたがって分類する．その後，ファイル名それぞれにツール名とフォーマット，または役割を表す語が含まれているかを調査する．

4.3 RQ3 の調査

拡張子とファイルフォーマットの調査

この調査では設定ファイルの拡張子とファイルの記述内容のフォーマットに着目し，これらが一致するかを調査する．調査においては RQ1 の調査において作成した設定ファイル名の集合 A からディレクトリ名を除いた集合を用いる．拡張子はファイル名から取得する．本調査における拡張子は先頭につけられたものを除いた「.」でファイル名を区切るときに得られる末尾の文字列とする．設定ファイルのフォーマットは各ツールのドキュメントを参照し取得する．その後，各設定ファイル名について拡張子が付加されているかや，拡張子がそのファイルのフォーマットを表しているかを調査する．

5. 調査結果

5.1 RQ1：設定ファイルはドットファイルとして管理されているか

RQ1 の調査結果を 図 2 に示す．調査対象のリポジトリからは全部で 2,417 個のユニークなファイル名を抽出した．分類の結果そのうち 618 個が設定ファイルの名前であった．また，このうち 402 個（65%）の名前が設定ファイルのデフォルトネームであった．ここでのデフォルトネームはそれぞれの設定ファイルに対応するツールのドキュメントなどで指定されている標準のファイル名を指す．現状，OSS の版管理リポジトリでは 618 個の設定ファイル名のうち 230 個（37%）がドットファイルであり 387 個（63%）が非ドットファイルであった．したがって，アプリケーションの設定ファイルをドットファイルとして管理するプラクティスが，版管理リポジトリにおいて共通認識として必ずしも存在するわけではないと考えられる．一方で図の赤色で示したドットファイルの集合に注目すると，258 個のうち 230 個（89%）のファイルが設定ファイルの名前であった．これより，ドットファイルとしてリポジトリ上で管理するファイルには一定の共通認識があると推測される．現状のリポジトリにおいて見られるドットファイルではない設定ファイルや，ドットファイルに対する認識については それぞれ 6.1.1 節と 6.1.2 節にて述べる．

> **RQ1 への回答**：現状，設定ファイルの多くは非ドットファイルとして管理されている．

5.2 RQ2：設定ファイルにはどのような名前が付けられているか

各要素が含まれていたファイル名の数を 表 2 に示す．

ツール名（T）については 618 個のファイル名のうち，459 個（74%）に含まれていた．このことから，設定ファイルには対応するツール名をファイル名に表す一定の共通認識があると考えられる．

つぎに，ファイルの役割が設定ファイルであると表す語（S）に着目する．この語に分類される語として，“config” や “cfg”，また “rc” や “opts” があった．これらの語が含まれるファイルは 618 個のうち 116 個（18%）であった．設定ファイルの命名において，そのファイルが設定ファイルであると明言する要素はそれほど用いられないといえる．現状ではこれらの語ではなく，対応するツール名を名前に含むファイルこそが設定ファイルとして用いられるという共通認識があると考えられる．

またファイルの記述形式（F）については，ディレクトリではないファイルの名前 534 個のうち，275 個（51%）に含まれていた．また，ドットファイル名とそうでないファイル名それぞれについてフォーマットを表す語がどれほど

[*3] https://github.com/abbrcode/abbreviations-in-code

含まれていたかを調査したところ，ドットファイルは 191 個のうち 96 個（50%），そうでないファイルは 343 個のうち 179 個（52%）であった．したがって，設定ファイル名にフォーマットを表す語を挿入するかはドットファイルであるかにかかわらず，各ツールやリポジトリの規則によってさまざまであると推測できる．

RQ2 への回答：設定ファイルには対応するツール名を含む名前をつけられる場合が多い．

5.3 RQ3: 設定ファイルの記述形式は拡張子から特定できるか

ファイルのフォーマットごとにファイル名に付加された拡張子を調査した結果を 表 3 に示す． 表 3 には汎用的なデータ記述言語やプログラミング言語で記述されたファイルのみを取り上げる．拡張子によってファイルのフォーマットを特定できるかは，そのフォーマットによって大きく異なるとわかる．したがって，拡張子を付加するかどうかは当該ファイルが設定ファイルであるかやドットファイルであるかに関わらずそのフォーマットによってある程度の共通認識があると予想できる．

一方でツールごとに固有な言語や構造，または一般的な構造化がされていないファイルも存在した．設定ファイルに特徴的な形式のファイルについての調査結果を 表 4 に示す．List に分類されるファイルには一行につき一つのパラメータもしくは Key-Value の対応を記述したファイルである．この形式に対応したファイルには一般的に用いられる特定の拡張子が存在しない．List ファイルに関するファイル識別性の考察は 6.2.1 節で述べる．DSL や Uniq に分類されたファイルは，対応するツールに特有の言語や構造で記述されたファイルである．特にツールに固有の言語はドメイン固有言語と呼ばれることが多かった．これら 2 つに分類されるフォーマットについてもそれぞれに対応する拡張子は存在しなかった．DSL ファイルに関するファイル識別性の考察は 6.2.2 節で述べる．

表 3 と 表 4 より，拡張子から実際にフォーマットを特定できるファイルは 256/534（47%）であった．5.2 節において述べたフォーマットを表す語が含まれるファイル名と比較して少ない理由は，フォーマットが拡張子以外で表現されているファイル名が存在するためである．以上より，フォーマットによっては拡張子をつける慣習が存在しているが，拡張子による記述形式の表現が難しい場合や，独自拡張子を利用する場合があり，フォーマットを拡張子から特定できるファイル名は半数以下に留まっているとわかる．

RQ3 への回答：記述形式と一致する拡張子が付加されたファイル名は全体の半数程度である．

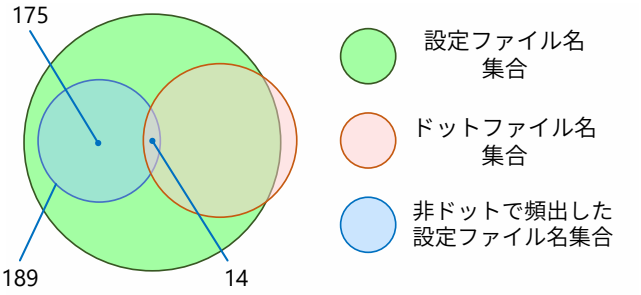

図 3: 非ドットファイルに特徴的な設定ファイルの数

6. 考察

6.1 ドットファイル名集合と設定ファイル名集合の不一致に対する分析

6.1.1 非ドットファイルである設定ファイル名の分析

RQ1 で得られたファイル集合をもとに，UNIX 系 OS 上の設定ファイル命名にある共通認識と版管理リポジトリで管理される設定ファイルの現状の命名との間にある違いの原因を考察する．現状では設定ファイルのほとんどがドットファイルではなかった．そこで収集した設定ファイルを詳細に調査し，どのような設定ファイルが存在するかを確認する．その結果，非ドットファイルに頻出な設定ファイルが見られた． 図 2 に対してそのようなファイル名集合を青色の集合として追加した図を 図 3 に，また追加で図示した集合に属するファイルの分類を 表 5 を示す．この集合に分類される設定ファイル名を持つファイルの多くがなぜ非ドットファイルとして管理されているか理由を考察する．

結果として，ビルド再現に必要な設定ファイルは非ドットファイルとして管理する認識があると考えられた．Scripts に分類されるファイルはツールに対してパラメータだけではなく，その出力もしくはプロセスの手順を定義している．具体的には `Makefile` や `build.gradle` といった出力を宣言する形式でプロセスを定義したファイルに加え，`Dockerfile` や CI のパイプラインを記述したファイルなどが挙げられる．本調査で得られた Scripts ファイルは多くがビルドツールのプロセスを記述したファイルであった．Manifest に分類されるファイルはビルド設定に加え，開発対象のプロジェクトの名前や開発者といったメタデータが記述されている．また Dependencies に分類されるファイルにはプロダクトの依存関係や，依存するパッケージの詳細なバージョンを記述している．これら 3 分類のファイルの多くはビルドの再現に必要な点に共通している．ビルドはプロダクトの開発と利用いずれにも必須のプロセスであり，ほぼすべてのステークホルダーに重要である．したがって，ビルドに必要なファイルは隠しファイルであるドットファイルにせず管理する認識が形成されたのではないかと考えられる．

表 3: フォーマットごとの拡張子の関係

	拡張子あり			拡張子なしまたは独自拡張子			
Format	dotfile	non-dotfile	小計	dotfile	non-dotfile	小計	合計
JSON	21	52	73 (78%)	12	8	20(22%)	93
YAML	40	41	81 (90%)	7	2	9(10%)	90
JS	19	27	46 (100%)	0	0	0(0%)	46
XML	0	9	9 (25%)	1	26	27(75%)	36
TOML	6	14	20 (95%)	0	1	1(5%)	21
INI	2	4	6 (33%)	9	3	12(67%)	18
その他	9	12	21 (47%)	6	17	23(53%)	44

表 4: 設定ファイルに特徴的な形式

名前	説明	#
List	一行に 1 つの値もしくは Key-Value を記述	80
DSL	その設定ファイルに固有の言語で記述	70
Uniq	その設定ファイルに固有の構造で記述	34
Other	その他のファイル	2
合計		186

表 5: 図 3 において新たに図示したファイルの分類

分類名	説明	#
Scripts	ツールのプロセス手順を宣言	100
Manifest	プロジェクト情報を記述	56
Dependencies	ビルドに必要なパッケージを記述	33
合計		189

6.1.2 設定ファイル名ではないドットファイルの分析

RQ1 の調査ではファイル名がドットファイルであるが設定ファイルではないファイルも 28 個存在した．これらのファイル内容を分類した結果を 表 6 に示す．これらのファイルがドットファイルとして扱われている意図は大きく分けて二つあると考えられる．

一つ目は慣例や命名規則に従うためである．これは Example に分類されるファイルに見られ，具体的には `.env.example` や `.ruby-gemset.sample` などがあった．それぞれ，環境変数を記述するときによく用いられる `.env` と Ruby を用いたプロジェクトの設定を行うファイルの一つである `.ruby-gemset` の具体的な例であり，慣習や命名規則に沿ったファイル名であるといえる．その一方でこれらのファイルは設定ファイルではなくプロダクトの利用者に向けた一種のドキュメントであるととらえられるため，共通認識の形成に向けては，ドキュメントを保管するサブディレクトリでの管理などが適していると考えられる．

表 6: 設定ファイルではないドットファイルの詳細

分類名	説明	#
Example	プロダクト設定ファイルの例示	9
Helper	開発者が用いるヘルパースクリプト	8
Document	開発者やプロダクト利用者に対する文書	6
Artifact	プロダクトの一部	4
その他		1
合計		28

二つ目はファイルをプロジェクトのステークホルダーの一部に対して隠す意図である．Helper に分類されるファイルはプロジェクト参加者が利用するファイルであるが，開発対象プロダクトの利用者が関心をもつ可能性は低い．同様にドットファイルとなっていた Document や Artifact はいずれも利用者に向けたファイルではない場合や，画像等の変更される機会が少ないと考えられるファイルであった．これらのファイルがドットファイルである理由はファイルの存在を隠す意図だと考えられる．したがって，これらのファイルをドットファイルとして管理するプラクティスはファイル識別性向上に役立つといえる．

6.2 フォーマットに着目したファイル名の再分析

RQ2 における調査では，ファイル名にフォーマットを表す要素（`F`）が含まれているのはドットファイル名とそうでないファイル名いずれも全体の 50%程度にとどまっていた．この理由について RQ3 に対する調査により，拡張子について共通の認識がないフォーマットであるため，または設定ファイルについて独自の言語やフォーマットを利用しているためであると二つの理由が考えられた．そこで，フォーマットが明らかになった状態で，拡張子を含めたファイル名全体とそのフォーマットの関係を改めて調査する．以下に分析の結果を示す．

6.2.1 各種ツールの ignore ファイル

拡張子を持たないファイルの例として各種ツールの ignore ファイルがあった．ここでの ignore ファイルは対応するツールの監視や影響下から特定のファイルまたはサブディレクトリを除外する役割を持つファイルを指す． 表 4 において List に分類した設定ファイルの一部は，ignore ファイルである．ignore ファイルは一行に一つの glob パターンに否定などの拡張を与えた共通の書式で記述され，この書式は共通認識として `.gitignore` と同様の書式と表現される．本調査で得た ignore ファイル名は全て，「ツール名+`ignore`」の名前をもつドットファイルであった．し

たがって，ignore ファイルは名前に一定の共通認識を持っており，書式を表現する拡張子は存在しないものの共通認識に従った命名はファイル識別性向上に役立ちうるといえる．

6.2.2 ドメイン固有言語で記述されたファイル

広く知られた拡張子を持たないファイルの一種としてドメイン固有言語で記述されたファイルも見られた． 表 4 において DSL に分類した設定ファイルはそれぞれの対応するツールのドキュメントにて，ドメイン固有言語で記述されたファイルと表現されている．これらのファイルの記述で利用される言語は多くの場合，そのファイルを書くための専用言語ともとらえられる．DSL に分類されたファイルには具体的に `Dockerfile` や `BUILD.bazel` などがあった．前者はコンテナ仮想化のためのプラットフォームである Docker でコンテナをビルドする手順を定めるファイルである．後者はビルドツールの一つである Bazel が実行時にビルド手順を読み込むファイルであり，Starlark と呼ばれる専用言語で記述されている．これらにはそれぞれ独自の言語が用いられている．このことから，それぞれの独自言語がそれぞれのツール名や設定ファイル名と強く結びついているため拡張子がなくともフォーマットについて一定の共通認識があると推測される．

7. 関連研究

設定ファイルを対象としてリポジトリマイニングを実施した研究は数多く存在する [13], [14]．Bessghaier ら [13] はシステムで用いられる設定ファイルの定性的な定義を目的として，クラウドコンピューティングプラットフォームである OpenStack のリポジトリから設定ファイルを抽出し分類を行った．Zampetti ら [14] は CI 環境で用いられるソースコード静的解析ツールについて，開発者らの利用方法や，静的解析ツールが CI プロセス上で及ぼす問題を調査するために CI の動作を定義する設定ファイルの調査を実施している．

本研究も同様に設定ファイルを対象としてリポジトリマイニングを実施し，版管理リポジトリ上の設定ファイルの理解を目的とした研究である．先行研究では特定のリポジトリ上の設定ファイルを対象にした調査や，特定のツールの設定ファイルに絞った調査を行っている．本研究では複数のリポジトリを対象としてさまざまな設定ファイルを調査している．一方で本研究では先行研究と異なり，設定ファイルの記述内容は詳細を調査せずファイル名のみを調査している．

8. 妥当性への脅威

8.1 調査対象のリポジトリ

本研究では調査対象のリポジトリとしてスター数の多いリポジトリを選定した．リポジトリの選定基準として他のメトリクスを採用した場合，調査結果が異なる場合が予想される．また，本研究で採用しなかった主開発言語のリポジトリを採用した場合にも調査結果が変わる可能性がある．

8.2 設定ファイルの定義

本研究において，設定ファイルの定義をソフトウェアの動作を変更するためのパラメータを定義するテキストファイルとした．しかし，設定や設定ファイルという言葉がさすファイルのスコープは様々であり [13], [15]，どのファイルを設定ファイルとするかに関しては議論の余地がある．

9. おわりに

本研究では，版管理リポジトリのファイル識別性向上を目的として設定ファイルの命名に対するプラクティスの提案とリポジトリマイニングによる設定ファイル名の現状調査を実施した．RQ1 では設定ファイルをドットファイルで管理するプラクティスは版管理リポジトリ上では一般的でないと判明した．RQ2 の調査では設定ファイルの命名においては，現状ツール名をファイル名に持たせる共通認識があると考えられた．RQ3 では，設定ファイルの記述形式などの影響から拡張子とフォーマットが一致するファイル名は全体の半分以下にとどまると判明した．以上から現状の設定ファイルの命名に広く存在する共通認識は対応するツール名の挿入に限られるという知見を得られた．

今後の課題として，ファイル名に含まれる要素のさらなる詳細な分析が考えられる．本稿では設定ファイルの役割として，“config” など設定ファイルに対応するツールが用いられるプロセスや設定が担うドメインなどを考慮しない大局的な語に注目した．しかし，設定ファイルに対応するツールはそれぞれ利用されるソフトウェア開発プロセスにおける場面はさまざまである．すると，ステークホルダーによってもそれぞれ関心のあるプロセスおよびそれに用いられるツールの設定ファイルも異なる．さらなるファイル識別性の向上にはファイルが果たす役割のさらに詳細なドメインの分析が必要になる．

謝辞 本研究の一部は，JSPS 科研費（JP24H00692, JP21H04877，JP21K18302）による助成を受けた．

参考文献

[1] Gousios, G., Kalliamvakou, E. and Spinellis, D.: Measuring developer contribution from software repository data, *Proc. International Conference on Mining Software Repositories (MSR)*, pp. 129–132 (2008).

[2] Ma, Y., Fakhoury, S., Christensen, M., Arnaoudova, V., Zogaan, W. and Mirakhorli, M.: Automatic classification of software artifacts in open-source applications, *Proc. International Conference on Mining Software Repositories (MSR)*, pp. 414–425 (2018).

[3] Omar, E., Margaret-Anne, S., Neil, E. and Andy, Z.:

Do as I do, not as I say: Do contribution guidelines match the github contribution process?, *Proc. International Conference on Software Maintenance and Evolution (ICSME)*, pp. 286–290 (2019).

[4] Steinmacher, I., Conte, T., Gerosa, M. A. and Redmiles, D.: Social Barriers Faced by Newcomers Placing Their First Contribution in Open Source Software Projects, *Proc. Conference on Computer Supported Cooperative Work & Social Computing (CSCW)*, pp. 1379–1392 (2015).

[5] Anquetil, N. and Lethbridge, T.: Recovering software architecture from the names of source files, *J. Software Maintenance: Research and Practice*, Vol. 11, No. 3, pp. 201–221 (1999).

[6] Parker-Wood, A., Long, D. D. E., Miller, E., Rigaux, P. and Isaacson, A.: A File By Any Other Name: Managing File Names with Metadata, *Proc. International Systems and Storage Conference (SYSTOR)*, pp. 1–11 (2014).

[7] Beller, M., Bholanath, R., McIntosh, S. and Zaidman, A.: Analyzing the State of Static Analysis: A Large-Scale Evaluation in Open Source Software, *Proc. International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pp. 470–481 (2016).

[8] Rostami Mazrae, P., Mens, T., Golzadeh, M. and Decan, A.: On the usage, co-usage and migration of CI/CD tools: A qualitative analysis, *J. Empirical Software Engineering*, Vol. 28, No. 2, p. 52 (2023).

[9] Craig, W., Kirstin, C. and William, M.: Customization in a UNIX Computing Environment, *Proc. Large Installation System Administration Conference (LISA)*, pp. 43–50 (1993).

[10] Wenhan, Z.: Using Crowd-Based Software Repositories to Better Understand Developer-User Interactions, PhD Thesis, University of Waterloo, Waterloo, Ontario, Canada (2023).

[11] Bastian, W., Karlitskaya, A., Poettering, L. and Löthberg, J.: XDG Base Directory specifications, `https://specifications.freedesktop.org/basedir-spec/latest/` (accessed 2025-07-22).

[12] Sayagh, M., Kerzazi, N., Adams, B. and Petrillo, F.: Software Configuration Engineering in Practice Interviews, Survey, and Systematic Literature Review, *Trans. Software Engineering*, Vol. 46, No. 6, pp. 646–673 (2020).

[13] Bessghaier, N., Sayagh, M., Ouni, A. and Mkaouer, M. W.: What constitutes the deployment and runtime configuration system? An empirical study on openstack projects, *Trans. Software Engineering and Methodology*, Vol. 33, No. 1, pp. 1–37 (2023).

[14] Zampetti, F., Scalabrino, S., Oliveto, R., Canfora, G. and Di Penta, M.: How Open Source Projects Use Static Code Analysis Tools in Continuous Integration Pipelines, *Proc. International Conference on Mining Software Repositories (MSR)*, pp. 334–344 (2017).

[15] Siegmund, N., Ruckel, N. and Siegmund, J.: Dimensions of software configuration: on the configuration context in modern software development, *Proc. Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, pp. 338–349 (2020).