社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

オープンソースAndroidアプリケーションのビルド可能性に関する調査

小池　耀†　眞鍋　雄貴††　松下　誠†　井上　克郎†††

† 大阪大学大学院情報科学研究科　〒565–0871 大阪府吹田市山田丘 1-5
†† 福知山公立大学情報学部情報学科　〒620–0886 京都府福知山市字堀 3370
††† 南山大学理工学部ソフトウェア工学科　〒466–8673 愛知県名古屋市昭和区山里町 18

あらまし　オープンソースソフトウェアをビルドする際，失敗し，修正を必要とされることがよくある．先行研究では，半数以上の Java オープンソースソフトウェアで自動的なビルドに失敗し，手動で修正を加えなければビルドが成功しない状況にあるとされている [1]．しかし，これらの研究は純粋な Java プロジェクトに焦点を当てており，Android アプリケーションのような他のエコシステムが関与しているアプリケーションのビルドに関する調査は行われていない．本論文では，一般的な環境を模した仮想環境で，オープンソースの Android アプリケーションの自動的なビルドが成功するか否かを調査した．そして，得られたログを分析し，Java プロジェクトの比較を通して Android アプリケーションのビルドの特徴について分析した．その結果，Android アプリケーションのビルドはリポジトリ作成日や最終更新日に影響を受けることが分かった．また， Gradle のバージョンが重要であることが分かった．
キーワード　ビルド，Android，オープンソースソフトウェア，リポジトリマイニング，ビルドログ調査

Investigation of Open Source Android Application Buildability

Yo KOIKE†, Yuki MANABE††, Makoto MATSUSHITA†, and Katsuro INOUE†††

† Graduate School of Information Science and Technology, Osaka University
1–5 Yamadaoka, Suita, Osaka, 565–0871 Japan
†† Faculty of Informatics, The University of Fukuchiyama　Hori, Fukuchiyama, Kyoto, 620–0886 Japan
††† Department of Software Engineering, Faculty of Science and Technology, Nanzan University
18 Yamazato, Shouwa, Nagoya, Aichi, 466–8673 Japan

Abstract　Building of Open source software often fail and require modifications. Prior studies have shown that more than half of Java open source software fails to build automatically and requires manual modifications to build successfully. However, these studies focused on pure Java projects and did not investigate the builds of projects involving other ecosystems such as Android applications. In this paper, we attempt to automatically build an open source Android application in a virtual environment that mimics a typical environment. We analyze the obtained logs and investigate the characteristics of building Android Apps through a comparison of Java projects. As a result, we found that Android app builds are affected by the repository creation date and the last update date. We also found that the Gradle version is important.
Key words　Build, Android, Open source project, Repository mining, Build log

1. まえがき

ソフトウェアを開発するときや，Git に公開されているソースコードからソフトウェアを利用する際には，ビルドが必要となる．しかし，完全に手作業でソフトウェアをビルドすることは手間がかかる．そのため，Gradle や Maven のようなビルドツールが一般的に用いられている．こうしたビルドツールは，ビルドに必要な作業をスクリプトファイルにまとめることで，コンパイルや依存関係の解決といった作業を自動的に行う．

ビルドツールにより，ビルドが容易に行えるようになったことで，ソフトウェアの構築以外の目的で利用することも容易になった．一例として，多数のプロジェクトを集め，コンパイル及びコンパイルの逆の処理であるデコンパイルを通して，ソースコードを解析する研究がある [2]．このような研究では，収集した多数のプロジェクトのビルドが正しく行われることが必須である．

しかし，ビルドツールを利用したソフトウェアの多くが，自動的なビルドに失敗し，何らかの手動の操作が必要な状態にあることが先行研究で明らかになっている [1], [3], [4]．ビルドを自動的に行えなければ，そのソフトウェアの利便性が下がるだ

けでなく，ビルドを利用した研究において，十分な量のデータを得ることが困難になる．よって，ビルドが手動の変更作業などを伴うことなく，完全に自動で行えることは，重要である．

ビルドツールを用いたビルドが失敗する原因を調査した先行研究は存在する [1], [4]〜[6]．しかし，これらの研究は特定の環境や言語を対象としており，未だ調査が行われていない環境や言語は多数存在している．

そこで，本研究ではオープンソース Android アプリケーションのビルドに焦点を当てる．Android アプリケーションは，スマートフォンの普及に伴い増加しているが，これらのビルドに関する先行研究は存在していない．本研究では，Java プロジェクトに対する先行研究のスクリプトを用いて GitHub のリポジトリからフォーク数などの一定の条件を満たすオープンソース Android アプリケーションを収集し，ビルドを試みる．その後得られたログから成功率やエラーの原因などをまとめた．

以降，2 章では先行研究の概要について，3 章では Android アプリケーションの収集，ビルドに用いるスクリプトの動作，ビルドエラーの分類について説明し，4 章でビルド結果について述べる．5 章では，ビルド結果に対する考察を行い，6 章で本研究のまとめと今後の課題を述べる．

2. 先行研究の概要

Java プロジェクトに対して，ビルドの成功率の調査した M. Sulír らの先行研究について説明する [1]．この先行研究の目的は，様々なビルドツールが存在するにもかかわらず，頻繁にオープンソースプロジェクトのビルドが失敗しているという経験を定量的に調査することである．GitHub から 7,264 個の Java プロジェクトを収集し，ビルドが成功するか否かを調査した．その結果，38% 以上のプロジェクトがビルドに失敗した．その主要な要因として依存関係に関するエラーがあることを示した．また，大規模なプロジェクト，古いプロジェクト，更新が古いプロジェクトは，ビルドが失敗する可能性が高いことを示している．

上記の研究は 2016 年に行われたが，2020 年にも同様の調査が行われた [4]．その結果，59.4% のプロジェクトがビルドに失敗しており，ビルドがより失敗しやすい状況になっていることが分かった．

3. 方　　法

3.1 調 査 対 象

本研究では，以下の条件を満たすリポジトリを調査対象とした．

（1） Java で記述されているアプリケーションのリポジトリであること

（2） オープンソースソフトウェアのライセンスが記述されたファイルがあること

（3） 1 回以上フォークされていること

（4） JNI や Java Mava の API を用いていないこと

（5） Android の API を用いていること

これらの基準を設けた理由について説明する．

条件（1）については，Java は Gradle, Maven, Ant などの高機能なビルドツールが複数存在し，Android アプリケーションにおいても頻繁に用いられるため，調査対象として選択した．

条件（2）については，オープンソースのソフトウェアを対象にしているため，そのソフトウェアが用いているライセンスについて記述されたファイルが存在していることを条件として設定した．

条件（3）は，個人的なリポジトリを排除し，他者からの協力を受けたリポジトリを対象とするために設定した．

そして，条件（4）と条件（5）は，JNI や Java ME といった技術を排除し，Android API の使用を確認することで，Android アプリケーションに焦点を当て，他の言語に関するデータが入らないようにするために設定した．

本研究では，上記の条件を満たすリポジトリを GitHub からランダムに 10000 個抽出し，ビルドファイルを認識できた 6,995 個のリポジトリに対して，ビルドを実行し成否を調査した．なお，この条件では，Android アプリケーション以外のリポジトリも対象となる可能性があるが，本論文ではこれらの調査対象のリポジトリが Android アプリケーションであると仮定し，以後 Android アプリと呼ぶ．

3.2 比 較 対 象

比較対象として，一般的なオープンソース Java プロジェクトのビルドについても調査した．比較対象とするリポジトリの条件は，条件（5）を Android の API を用いていないことに変更した点以外，上記の条件と同じである．これは先行研究と全く同じ条件だが，先行研究は 2016 年と 2020 年に行われており，それ以後のリポジトリは調査に用いられていない．2020 年以後のプロジェクトも結果に含めるために，先行研究の結果ではなく，著者が調査を行った結果を用いている．本研究では，上記の条件を満たすリポジトリを GitHub からランダムに 10000 個抽出し，ビルドファイルを認識できた 7,196 個のリポジトリに対して，ビルドを実行し成否を調査した．

3.3 ビルドの手順

この章では，上記の条件を満たすリポジトリに対して行うビルドの手順について説明する．

3.3.1 ビルドツール検索

Java の主なビルドツールとして，Gradle, Maven, Ant の 3 つが存在する．ルートディレクトリ内にこれらのシステムが用いるビルドファイルが存在しているか確認した．もし複数のビルドファイルが存在している場合は，Gradle, Maven, Ant の順に優先する．これは，Gradle, Maven, Ant の順に新しいビルドツールであり，新しいビルドツールが主に使用され，古いビルドツールは互換性のために削除していないという状況を想定しているためである．

3.3.2 ビルド環境

本研究では，ビルドを Docker 環境で行った．これは，Android アプリ開発者の環境を高い再現性を持ってシミュレーションするためである．Docker のイメージは先行研究で使用されたものを一部修正して活用した．先行研究で用いられたイメージは以下のようなソフトウェアで構成されている．

（1） RPM 系 Linux ディストリビューション Fedora
（2） JDK 8
（3） Gradle 6.5.1 , Maven, Ant
（4） Apache Ivy
（5） Git
（6） Ruby

上記に加え，Android アプリのビルドのために Android SDK 26.1.1 を追加した．

3.4 ビルドツールの制御

ビルドツールを制御するスクリプトの動作について説明する．制御スクリプトは，各プロジェクトのビルドを実行する．各アプリのビルドファイルに基づいたビルドツールに対応するコマンドを実行する．使用したコマンドを表 1 に示す．

ビルドツールの一般的な目的は，実行可能あるいは配布可能なアーカイブファイルを，複雑な説明やファイルの書き換えなどをすることなく [7]，どのようなアプリに対しても自動的に得ることである．よって，あるアプリがビルドツールに対して正常に動作しているか調査する手法として，これらのコマンドを実行し，その出力を記録することは，有意義であると言える．また，ビルドの際に生成物が正常に動作するかテストする場合があるが，テストに失敗しても，アーカイブファイルが出力されれば，ソフトウェアが正常に実行できる場合が多いため，テストを除外できるビルドツールでは除外している．

各ビルドでは，ビルドツールが正常に動作したかを確認するため，標準出力とエラー出力をログファイルに出力する．各ビルドの実行時間は 1 時間に制限した．これは，ビルドが無限実行されるケースが存在するためである．終了コードやソースのファイル数など，補助的なメトリクスは CSV ファイルに保存する．

仕様ツール	ビルドファイル名	コマンド
Gradle	Build.gradle	gradle clean assemble
Maven	pom.xml	mvn clean package -DskipTests
Ant	build.xml	ant clean; ant jar ‖ ant war ‖ ant dist ‖ ant

表 1 使用したコマンド

3.5 ビルドエラーの分類

この章では，ログからビルドエラーを分類する方法について説明する．ログを検索し，特定の文字列が存在するのか調査することでエラータイプに分類する．ビルドツールはそれぞれ独自のログ形式で出力するため，ビルドツール毎に分類を決定する必要があるが，後述の理由により本研究では Gradle に注目するため， Gradle におけるビルドエラーの分類について説明する．

3.5.1 Gradle

Gradle のビルドエラーでは，一つの例外が複数の例外を原因として発生することで，例外がツリー構造で形成されることがある．例として，失敗したビルドのログから失敗原因を抜き出し簡略化したものを以下に示す.

```
* Exception is:TaskExecutionException:
Caused by: DefaultLenientConfiguration
$ArtifactResolveException:
Cause 1: ModuleVersionResolveException:
Caused by: ModuleVersionResolveException: 中略
Cause 2: ModuleVersionResolveException: 中略
```

上記の例では，「TaskExecutionException」が「DefaultLenient-Configuration $ArtifactResolveException:」によって引き起こされ，さらに，そのエラーが「Cause 1: ModuleVersionResolveException」と「Cause 2: ModuleVersionResolveException」によって引き起こされている．これらのエラーは同じ名前だが，異なった原因によって引き起こされている．「Cause 1: ModuleVersionResolve-Exception」はさらに異なる「ModuleVersionResolveException」によって引き起こされている．このように，一つの例外が複数の例外が原因で引き起こされる場合がある．

一つのビルドに複数のエラータイプを割り当てると，その後の解析が複雑になるため，最も関連性の高い例外を以下の方法で決定する．例外がツリーを形成しない単一の例外であれば，それを選択する．例外がツリー構造になっている場合は，根となる例外を引き起こしている例外を選択する．上記の例の場合は，根となる例外の「TaskExecutionException」を引き起こしている，「DefaultLenientConfiguration $ArtifactResolveException」を選択する．

3.5.2 エラータイプの分類

ログから分類したエラータイプは非常に数が多く，人間には読みにくい名前をしているため，より分かりやすい名前に再分類している．これらの作業は手作業で行うため，頻繁に発生するエラータイプのみを分類している．結果として 84 種類のエラーが分類された．分類が不明なものは Unknown となっている．

4. 調 査 結 果

本章では，Android アプリと Java プロジェクトの比較を通して，Android アプリのビルドの状況について調査した結果をまとめる．

まず，ルートディレクトリ内で発見したビルドファイルの種類の割合について，Java プロジェクトについては図 1 に，Android アプリについては図 2 に示す．Android アプリでは，ほぼすべてのプロジェクトで Gradle が使用されていることが分かる．これは，Android アプリの開発で使用される Android Studio がビルドに Gradle を用いているためである．よって，本研究では，Gradle について，詳しく調査する．また，ビルドファイルを認識できないアプリが 30.05% 存在した．

次に，ビルドの成功率について，図 3 と図 4 に示す．図 3 においてビルドされた Java プロジェクトは 7196 個であり，図 4 においてビルドされた Android アプリは 6995 個であった．図 3 に示されているように，Java プロジェクトは 44.22% 成功している一方で，図 4 では，Android アプリの成功したビルドはわずか 12.44% である．このことから，多くのアプリが，手動で何らかの操作を行わなければ，成功しない状況にあることが

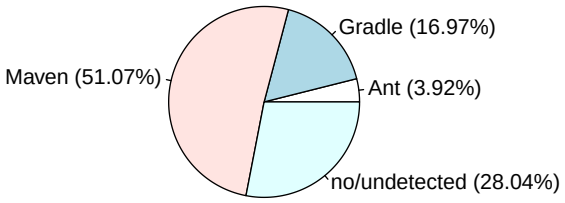

図 1　Java プロジェクトで使用されたビルドツール

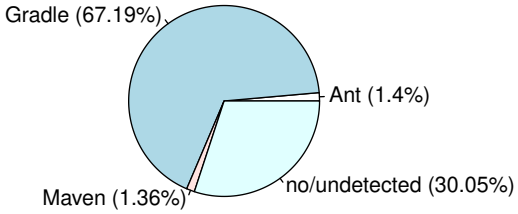

図 2　Android アプリで使用されたビルドツール

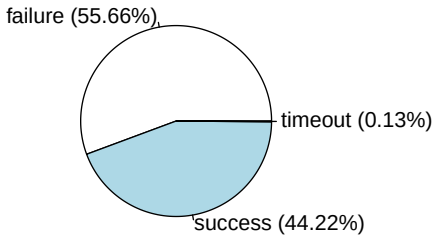

図 3　Java プロジェクトのビルドの成功率

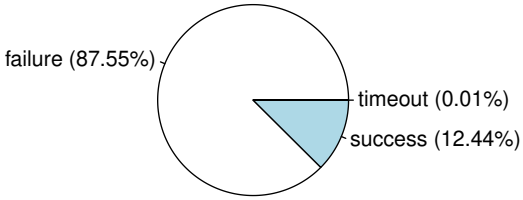

図 4　Android アプリのビルドの成功率

分かる．

続いて，成功したプロジェクトと，失敗したプロジェクトの持つファイル数，スター数，リポジトリ作成日，最終更新日の分布と中央値を表すビーンプロットをそれぞれ図 5，6，7，8 に示す．各図において左図が Java プロジェクトの図で，右図が Android アプリの図である．

まず図 5 では，各プロジェクトが含むファイル数とビルド成功率との関係を示す．縦軸はファイル数、横軸は成功／失敗したプロジェクトの密度を示す。図 5 の左図の通り，Java プロジェクトでは失敗したビルドのほうがファイル数の中央値が大きいため，ファイル数が多いほど，失敗しやすい傾向にあることを示している．一方で，図 5 の右図を見ると，Android アプリは Java プロジェクトと比べ，失敗したビルドのほうが中央値が小さく，逆転していることが分かる．

次に，図 6 のスター数については，Java プロジェクト，Android アプリの中央値に大差なく，ビルドの成功率とスター数には関連がないことが分かる．

図 7，図 8 に示されている，リポジトリ作成日や最終更新日とビルド成功率の関係では，成功したリポジトリのほうが中央値が低く，最近作成され，更新が新しいリポジトリのほうが成功率が高いことが Java プロジェクトでも，Android アプリでも共通していると言える．

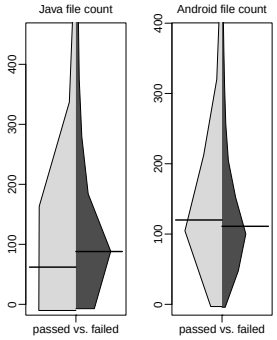

図 5　Java プロジェクトと Android アプリのファイル数とビルド成功率の関係

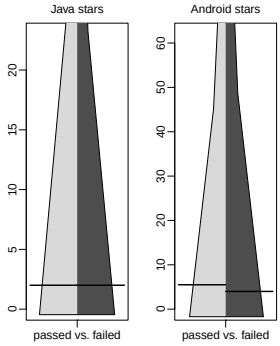

図 6　Java プロジェクトと Android アプリのスター数とビルド成功率の関係

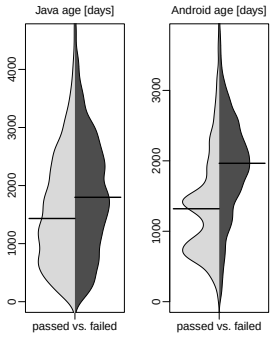

図 7　Java プロジェクトと Android アプリのリポジトリ作成日からの日数とビルド成功率の関係

最後に，Gradle を用いたビルドの失敗原因について，それぞれ図 9 と図 10 に示す．Java プロジェクトでは，様々なエラー

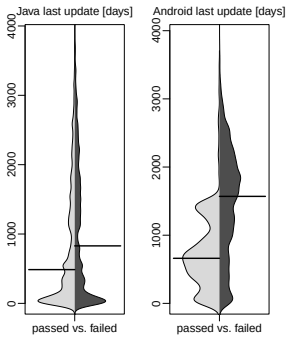

図 8　Java プロジェクトと Android アプリの最終更新日からの日数とビルド成功率の関係

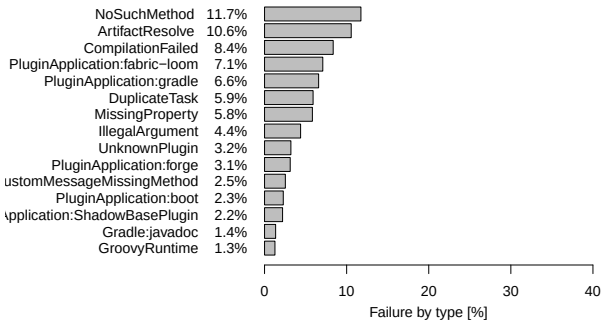

図 9　Gradle を用いた Java プロジェクトのビルドの失敗原因

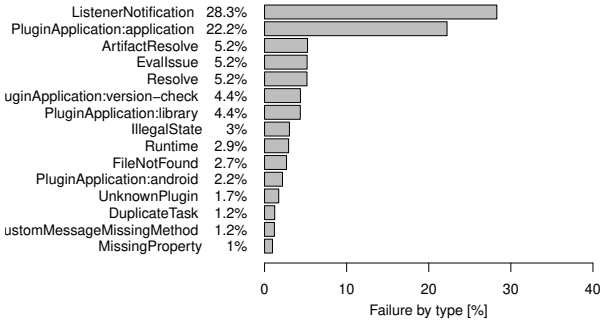

図 10　Gradle を用いた Android アプリのビルドの失敗原因

タイプが存在している一方で，Android アプリでは，「ListenerNotification」と「PluginApplication:application」のエラーが過半数を占めていることが分かる．この原因について調査したところ，「ListenerNotification」については，ビルドに用いる Gradle のバージョンが正しくないことが原因の一つであった．

4.1　Gradle wrapper を用いた調査結果

Gradle にはこうした環境の不整合によるビルドエラーを防ぐために，Gradle wrapper が存在している．Gradle wrapper は Gradle をインストールしていない環境や，異なるバージョンの Gradle を用いた環境でも，開発者が指定した Gradle のバージョンでビルドを行える Gradle の機能である．

Gradle wrapper が使用可能なアプリであれば，使用する設定で，7,021 個のアプリを対象に調査した結果を図 11，12，13 に示す．

図 11 に示されている Grade wrapper を用いたビルドの成功率と，図 4 を比較すると，ビルドの成功率が改善していることが分かる．また，タイムアウトするビルドが増加していることが分かる．

図 12 は，成功したプロジェクトと，失敗したプロジェクトの持つファイル数，スター数，リポジトリ作成日，最終更新日の分布と中央値を表すビーンプロットを示している．

図 12 の左側の 2 つの図はそれぞれファイル数とスター数における Gradle wrapper を用いたビルドの成功率を示している．図 5 と図 6 の右図と比較すると，大差ないことが分かる．

図 12 の右側の 2 つの図はそれぞれリポジトリ作成日と最終更新日における Gradle wrapper を用いたビルドの成功率を示している．図 7 と図 8 の右図と比較すると，成功したビルドにおける，リポジトリ作成日と最終更新日の中央値が増加し，より古く，更新頻度の低い Android アプリを対象としたビルドが成功していることが分かる．

ビルドの失敗原因を示している図 10 と図 13 を比較すると，失敗原因が大きく変化していることが分かる．図 10 では過半数を占めていた「ListenerNotification」と「PluginApplication:application」のエラーは確認できず，「Runtime」や「CommandLineArgument」が最も多い原因となった．

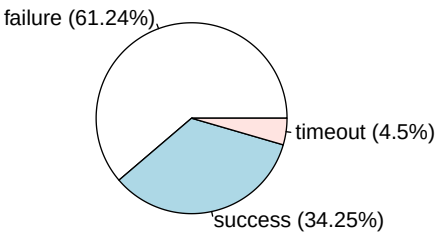

図 11　Gradle wrapper を用いた Android アプリのビルドの成功率

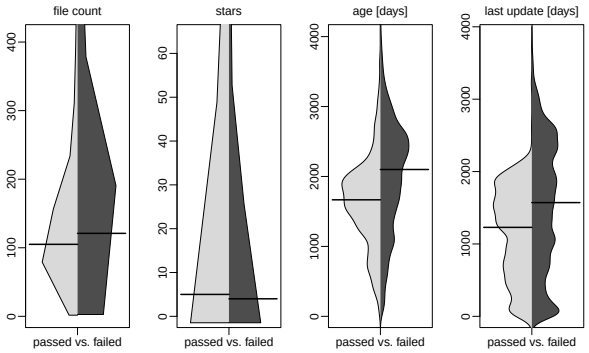

図 12　Gradle wrapper を用いた Android アプリの特徴

5.　考　　察

5.1　ビルドの成功率とリポジトリ作成日と最終更新日

図 7 と図 8 の右図に示されている，Android アプリのビルドの成功率とリポジトリ作成日，最終更新日の関係に注目する．Android アプリのビルドでは，リポジトリ作成日と最終更新日の両方において，ビルドが成功したグループに二つ，失敗したグループに一つ山ができている．このような山ができる原因

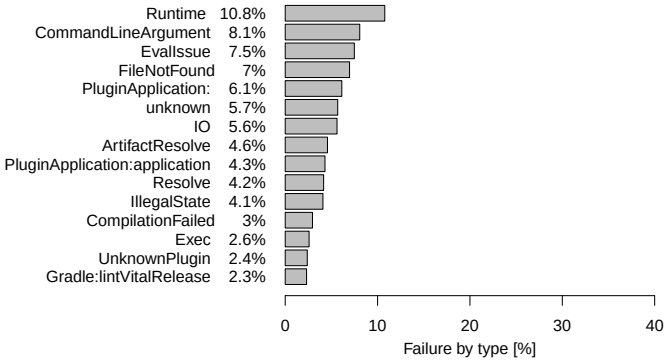

図 13　Gradle wrapper を用いた Android アプリのビルドの失敗原因

として Android SDK の更新が考えられる．例えば，2020 年 7 月から 8 月にかけて，Android SDK コマンドラインツールや Android Gradle プラグイン，SDK Platfrom などが更新されている [8], [9]．2020 年 7 月は約 700 日前であり，ビルドに成功したリポジトリが多く分布している．このような更新があった際に作られたリポジトリや，対応作業を行ったリポジトリは，ビルド成功率が高くなり，Android SDK が更新されても，対応作業を行わなかったリポジトリはビルドが失敗しやすくなった可能性がある．約 1400 日前の 2018 年 8 月から 9 月にも Android Gradle プラグインや SDK Platform が更新されており，ビルドに成功したリポジトリが多く分布しているタイミングである．

一方で，約 2000 日前の 2016 年頃に作成，更新されたリポジトリはビルドが失敗しやすくなっている．この原因として，2016 年ごろの SDK Platform などの更新に合わせ作成，更新されたリポジトリが放置され，その後の非互換の更新で使用不能になった可能性が考えられる．

5.2　Gradle wrapper の効果

Android アプリのビルドが失敗する原因として，Gradle のバージョン違いがあることが分かった．しかし，Docker イメージ内の Android SDK Tools のバージョンや Gradle のバージョンを変えて，同様の調査を行ったが，4 章の調査結果と同様の傾向を示していた．

Gradle wrapper を用いたビルドでは，ビルドの成功率が上昇しており，図 13 では，Gradle のバージョン違いが原因である「ListenerNotification」のエラーが確認できないことから環境の不整合によるエラーを減少させる効果があることが分かる．また，図 7, 図 8 の右図と図 12 の右側の二つの図を比較すると，図 7, 図 8 の右図では約 2000 日前作成されたリポジトリはビルドが失敗しやすい傾向にあったが，図 12 ではその傾向がなくなっていることが分かる．これも Gradle wrapper によって，開発者が指定した環境でビルドを行ったことにより，環境の不整合によるエラーが減少したためと考えられる．

一方で，図 4 と図 11 を比較すると，図 4 では 0.13% であった timeout の割合が図 11 では 4.51% に上昇している．原因として，Java プロジェクトと比較して，Android アプリのビルド時間が長い傾向にある可能性が考えられる．

6.　まとめと今後の課題

本研究では，多数の Android アプリに対して，ビルドの成功率に関する調査を行い，既存研究で示された Java プロジェクトでの結果と比較した．その結果，Android アプリのビルド成功率は，リポジトリが作成された日時や最終更新日と関係があることが分かった．また，ビルドの成功率を下げている要因として，Gradle のバージョンが関係しており，Gradle wrapper を用いることで解決できることが分かった．

今後の研究課題として，リポジトリ作成日や最終更新日とビルド成功率の関連について，詳しく調べることが挙げられる．

また，利用したアプリは全て GitHub から抽出したものであり，偏りが発生している可能性がある [10]．そのため，他の Git ホスティングサービスなどで調査することも考えられる．

他にも，こうしたビルド成功率に関する研究は，先行研究が少なく，C や C++ のような他の言語への拡張や，他のソフトウェアメトリクスと関連性の調査など，様々な研究方針が考えられる．

謝　　辞

本研究は JSPS 科研費 JP18H04094，JP21K02862 の助成を受けたものです．

文　　献

[1] M. Sulír and J. Porubän: “A quantitative study of java software buildability”, Proceedings of the 7th International Workshop on Evaluation and Usability of Programming Languages and Tools, pp. 17–25 (2016).

[2] C. Ragkhitwetsagul and J. Krinke: “Using compilation/decompilation to enhance clone detection”, 2017 IEEE 11th International Workshop on Software Clones (IWSC)IEEE, pp. 1–7 (2017).

[3] N. Kerzazi, F. Khomh and B. Adams: “Why do automated builds break? an empirical study”, 2014 IEEE International Conference on Software Maintenance and Evolution, pp. 41–50 (2014).

[4] M. Sulír, M. Bačíková, M. Madeja, S. Chodarev and J. Juhár: “Large-scale dataset of local java software build results”, Data, **5**, 3, p. 86 (2020).

[5] A. Neitsch, K. Wong and M. W. Godfrey: “Build system issues in multilanguage software”, 2012 28th IEEE International Conference on Software Maintenance (ICSM), pp. 140–149 (2012).

[6] Y. Lou, Z. Chen, Y. Cao, D. Hao and L. Zhang: “Understanding Build Issue Resolution in Practice: Symptoms and Fix Patterns”, p. 617–628, Association for Computing Machinery, New York, NY, USA (2020).

[7] Joel Spolsky, 青木 靖：“Joel on Software”, オーム社 (2005).

[8] “Android Gradle プラグインのリリースノート”, https://developer.android.com/studio/releases/gradle-plugin?hl=ja.

[9] “SDK Platform リリースノート”, https://developer.android.com/studio/releases/platforms?hl=ja.

[10] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German and D. Damian: “The promises and perils of mining github”, Proceedings of the 11th working conference on mining software repositories, pp. 92–101 (2014).