情報処理学会論文誌 Vol.64 No.9 1394–1398 (Sep. 2023)

テクニカルノート

Javaを用いたプロジェクトおよびKotlinを用いたAndroidアプリケーションを対象としたビルド可能性調査

小池 耀[1] 眞鍋 雄貴[2] 神田 哲也[1] 井上 克郎[3,a)] 肥後 芳樹[1]

受付日 2023年3月31日, 採録日 2023年6月5日

概要：ビルドツールは現代のソフトウェア開発に不可欠な存在であるが，自動ビルドはしばしば失敗し，多大なコストをかけて修正する必要がある．このため，ビルド可能性に関する研究が行われているが，Android アプリケーションについての研究はほとんどない．そこで，本研究では，それぞれ GitHub で公開されている 7 千件余りの Java を用いた一般的な開発プロジェクト，および Kotlin を用いた Android アプリケーションの開発プロジェクトに対して自動ビルドを実行し，ビルドの成功率やエラーを調査した．結果，ビルドの成功率は低く，Android アプリケーションのビルドエラーの原因は File not found や依存関係に関するエラーが多いことが分かった．

キーワード：ビルド，Android，オープンソースソフトウェア（OSS），リポジトリマイニング

Buildability Study for Java Projects and Kotlin's Android Applications

Yo Koike[1] Yuki Manabe[2] Tetsuya Kanda[1] Katsuro Inoue[3,a)] Yoshiki Higo[1]

Received: March 31, 2023, Accepted: June 5, 2023

Abstract: Build tools are an integral part of modern software development, but automated builds often fail and must be corrected at a great cost. For this reason, research on buildability has been conducted, but there is little research on Android applications. In this study, we executed automated builds on over 7,000 distinct Java projects and Kotlin's Android applications in GitHub, and conducted an examination of the success rate and errors of these builds. The results showed that the build success rate was low and that the most common causes of build errors in Android applications were 'file not found' errors and dependency errors.

Keywords: build, Android, open source software, repository mining

1. まえがき

ビルドは，ソースコードや各種リソースファイルを入力とし，ライブラリのダウンロード，環境に合わせて修正，コンパイル，リンク，オブジェクトの生成，配置などの一連の作業を経て実行可能なファイルを出力することである．

ソフトウェアを開発する際や，GitHub などで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，大きな労力を要する．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専用のファイルに記述することで，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．これにより，ビルドを 1 つのコマンドのみで実行することができる．

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

[1] 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University, Suita, Osaka 565–0871, Japan
[2] 福知山公立大学情報学部情報学科
Department of Informatics, Faculty of Informatics, The University of Fukuchiyama, Fukuchiyama, Kyoto 620–0886, Japan
[3] 南山大学理工学部ソフトウェア工学科
Department of Software Engineering, Faculty of Science and Technology, Nanzan University, Nagoya, Aichi 466–8673, Japan
[a)] inoue599@nanzan-u.ac.jp

態にあることが先行研究で明らかになっている [1], [2], [3]. ビルドを自動的に行えず，複数の手作業による操作が必要な状況は一般的に望ましくないとされている [4]．また，近年では，ビルドを利用してソースコードの解析を行う研究などが存在している [5]．自動的なビルドが行えなければ，こうした研究において十分な量のデータを得ることも困難になる．よって，ビルドが手動の変更作業などをともなうことなく，完全に自動で行えることは重要である．

そこで本研究では，先行研究 [2], [3] が行った一般的な Java の開発プロジェクトのビルト可能性の現状を調べるとともに，Kotlin を用いた Android アプリケーションの開発プロジェクトの自動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ともない増加しているが，これらのビルドが失敗する原因について調査した先行研究は存在していない．そのため，オープンソース Android アプリケーションのビルドの成功率や失敗する原因などは不明であり，調査が必要である．

本研究では，Java プロジェクトに対する先行研究と同様に 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，自動的なビルドを試みる．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．また，出力されたエラーの原因とその解決方法について調査した．

以降，2 章では，自動的なビルドに関する先行研究について，3 章では調査方法について述べる．4 章では調査結果について，5 章ではビルドエラーの原因の分析について述べる．最後に 6 章でまとめについて述べる．

2. 自動的なビルドに関する先行研究

自動的なビルドの現状について調査した Sulír らの先行研究について説明する [2]．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．ライセンス記述があることやフォークされていることなどいくつかの条件を満たす 7,264 個の Java プロジェクトを GitHub から収集し，自動的な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．

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 [3]．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．

この 2016 年での調査ではビルドの実行環境として Java8 を用いており，2020 年の調査では Java11 を用いていた．そのため，Java8 と Java11 の非互換性がビルドに与える影響が不明となっている．そこで，同一の Java プロジェクト群に対して Java8 を用いるビルド環境と Java11 を用いるビルド環境の 2 つを用意しビルド成功率の調査を行った．

一方，先行研究からビルドツールを用いた自動的なビルドの成功率が示されたが，Java という特定の言語のみを対象としており，いまだに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，Kotlin で記述された Android アプリケーションについても調査対象とした．

3. 調査方法

本研究ではまず，Java プロジェクトに対して先行研究と同様の調査を実行する．その後オープンソース Android アプリケーションのビルドの成功率や失敗原因について調査する．以下の Research Question を設定した．

- RQ1：現在の Java プロジェクトや Android アプリケーションにおけるビルド成功率はどの程度か
- RQ2：Android アプリのビルドの失敗原因は何か．またそれを解消するにはどうすればよいか

RQ1 では現在の Java プロジェクトや Android アプリケーションにおけるビルドの成功率を調査することで，これらのビルドに問題が存在するか否か明らかにする．

RQ2 では，Android アプリケーションのビルドの失敗原因や対処方法を調査し，Android アプリケーションのビルドを改善させる方法を明らかにする．

これらの Research Question に回答するために，まず，Android アプリケーションではない一般の Java プロジェクトに対して先行研究と同様の調査を実行し，2016 年と 2020 年の結果と比較する．次に，Android アプリケーションを収集し，ビルドを実行する．その後，得られたログからビルドの成否やビルドエラーをまとめたデータセットを構築する．また，エラーをその原因によって手動で分類し，それぞれ対処方法について調査する．

3.1 調査対象

本研究では，先行研究と同様の条件（Java を用いる，OSS のライセンス記述がある，1 回以上のフォークがある，JNI/Java Mava/Android API を用いていない）で，現在の GitHub から Java プロジェクトを抽出し調査対象とした．Android アプリケーションについても条件を一部変更し，GitHub から抽出した．変更した条件は，用いている言語が Java であるという条件を Kotlin にした点と，Android の API を用いていないという条件を Android の API を用いているとした点である．

これらの条件を満たす Java プロジェクト，Android アプリケーションをそれぞれ 10,000 個を GitHub から無作為に抽出し，調査対象とした．

3.2 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．

3.2.1 ビルドツールの選択

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，存在している場合はビルドを行った．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 の順に優先した．一方 Android アプリケーションの調査については，Gradle のみを用いる．これは，Android アプリケーションは開発環境の影響でほぼ 100%が Gradle を用いているためである．

調査対象の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．結果として Java プロジェクトは 7,196 個，Android アプリケーションは 7,362 個にビルドファイルが確認された．これらの Java プロジェクト，Android アプリケーションに対してビルドを行う．

3.2.2 ビルドエラーの分類

Android アプリケーションのビルドエラーの種類を把握しやすい形に分類した．まず，Gradle の実行エラーを記録したログにあるスタックのように構造化された原因記述のなかから，最上位に記述された原因で分類した．その結果 154 種類が確認された．これらのうち頻繁に発生している 54 種類を著者の 1 人が手作業で 10 種類に分類した．これら 10 種類の原因が全体の 83.68%を占めている．その他の原因のビルドエラーは全体の 16.32%であり，未分類とした．

4. 調査結果

本章では，Java プロジェクトと Android アプリケーションのビルドの状況について調査した結果をまとめる．

4.1 RQ1：現在の Java プロジェクトや Android アプリケーションにおけるビルド成功率はどの程度か

2016 年と 2020 年の先行研究における Java プロジェクトのビルド成功率と 2022 年現在の Java プロジェクトと Android アプリケーションのビルド成功率を**表 1** にまとめる．ここで Java8 とあるのは，Java8 を用いた環境でビルドを試みたことを示す．Java11 も同様である．

2016 年の結果では成功率が 61.87%となっているがほかの結果はすべて 40%程度となっている．

RQ1 の回答

近年の Java プロジェクトおよび Android アプリケーションの多くでは自動的なビルドがうまく機能しておらず，成功率は 40%から 46%程度である．

表 1 ビルド成功率の変化（太字は本研究結果）
Table 1 Evolution of Build Success Rate (Bold lines are the result of this work).

言語	成功率	失敗率	時間切れ
Java プロジェクト 2016 年	61.87%	38.05%	0.08%
Java プロジェクト 2020 年	40.49%	59.38%	0.12%
Java プロジェクト (Java8)	**46.1%**	**53.75%**	**0.15%**
Java プロジェクト (Java11)	**43.02%**	**56.6%**	**0.38%**
Android アプリ (Java8)	**40.97%**	**58.95%**	**0.08%**
Android アプリ (Java11)	**48.11%**	**51.83%**	**0.05%**

表 2 Android アプリケーションのビルドエラーの分類結果
Table 2 Classification of Android-Application Build Errors.

分類名	割合
File not found	27.84%
依存関係	19.03%
間違った JDK のバージョン	12.59%
ビルドファイルの設定エラー	7.57%
ビルドファイルの構文エラー	6.51%
Kotlin コンパイル	6.33%
パッケージング	1.78%
外部プログラムの実行	1.42%
他の言語のコンパイル	0.53%
ドキュメント生成	0.04%
バージョン管理システム	0.04%
未分類	16.32%

4.2 RQ2：Android アプリケーションのビルドの失敗原因は何か．またそれを解消するにはどうすればよいか

Android アプリケーションのビルドエラーを手動で排他的に分類した結果について**表 2** に示す．

この表に示すように，File not found が最も多い失敗原因で，依存関係が次に続いており，これらについては次章で述べる．Kotlin コンパイルとは，Kotlin のソースコードに問題がありコンパイルに失敗するエラーで，パッケージングとは成果物の出力時のエラーである．外部プログラムの実行は，他のプログラムやプロセスとの通信時のエラーを意味する．

RQ2 の回答

ビルドエラーの原因として最も多いのが File not found で約 28%であり，次いで依存関係が約 19%と多い．

5. ビルドエラーの原因の分析

RQ2 において，Android アプリケーションのビルドエラーは 10 種類に分類された．これらの原因と対処方法について分析を行った．ここでは File not found，依存関係，ビルドファイルの設定エラーについて述べる．

5.1 File not found の原因と対処方法

File not found はファイルが見つからないという一般的なエラーである．そのため原因や対処方法は様々であるが，特に割合が多かったものに対して調査を行った．

5.1.1 Classnotfound の原因と対処方法

File not found の最も大きい割合を占めるビルドエラーは「Classnotfound」であり，ビルドエラー全体の約 6%，File not found 全体の 21.6%を占めていた．また，「Classnotfound」の発生原因のほとんどは gralde-wrapper.jar というファイルが見つからないことであった．これは開発者がソースコードを git に push する際，gitignore の設定ミスにより gralde-wrapper.jar ファイルが push されないことが一因と考えられる．gradle wrap というコマンドにより，gradle-wrapper.jar を作り直すことで対処することが可能であった．

5.1.2 Gradle:processDebugGoogleServices の原因と対処方法

「Gradle:processDebugGoogleServices」が File not found で 2 番目に大きい割合を占めるビルドエラーであり，ビルドエラー全体の約 5.4%，File not found 全体の 19.4%を占めていた．このビルドエラーの原因は google-services.json というファイルが存在しないことである．このファイルは，Firebase という Google のモバイルプラットフォームのサービスを利用する際に必要となる．このファイルを Firebase のホームページから作成し，適切な場所に配置することによってビルドエラーを解決することができた．

5.1.3 File not found を防ぐための注意

上記の 2 つのエラーを合わせるとビルドエラー全体の 10%以上が，開発者が配置しておくべきファイルが存在しないことにより生じていることが分かる．これらのビルドエラーが発生していた Android アプリのドキュメントを複数調査したが，これらのファイルを用意させるような指示は確認できなかった．そのため，修正するには利用者が自力で調査する必要があり，ある程度の知識と時間を要すると考えられる．開発者はソフトウェアの公開や更新の際に，ビルドに必要なファイルがすべて存在しているかについて注意を払うべきである．

5.2 依存関係の原因と対処方法

依存関係のエラーは依存先の消滅やバージョンの互換性の問題など様々な原因により，依存関係の解決ができないことが原因である．依存関係の問題に対処するためには，ビルドの依存関係が記述されている build.gradle ファイルを修正する必要がある．たとえば依存関係が原因でビルドが失敗した udex-app-android というリポジトリでは，依存先として GitHub のリポジトリのある特定のコミットを指定していた．しかし，このコミットがマージが原因で消滅したことにより，依存先がなくなり，ビルドが失敗した．そのため，同時期のコミットを新たに指定することによって解決した．

5.3 ビルドファイルの設定エラーの原因と対処方法

ビルドファイルの設定エラーに関するエラーのなかで最も多かったビルドエラーは InvalidUserData であり，ビルドエラー全体の約 2%を占めていた．このビルドエラーの原因は Android アプリケーションを Google Play ストアに公開するために必要な署名が存在しないことが原因であった．keytool コマンドにより適当な署名を作成することで対処ができた．また，署名に関するタスクを削除することによっても対処することができた．

6. まとめ

本研究では，GitHub から Java プロジェクトと Android アプリケーションを抽出し，ビルドツールを用いたビルドの成功率や失敗原因などについて調査した．2 つの RQ について調査を行い，RQ1 については，近年の Java プロジェクトおよび Android アプリケーションの多くでは自動的なビルドが機能していないことが明らかとなった．また RQ2 では Android アプリケーションのビルドエラーを手動で分類した結果，File not found が原因の場合が最も多く，次いで依存関係が原因の場合が多いことが分かった．そして，それらの対処方法について分類結果ごとに考察した．

今後の課題として，Java や Kotlin 以外に対する調査が考えられる．Java や Kotlin とまったく異なる言語，たとえば C 言語や Python などの現在広く用いられている言語や，web サイトなどで用いられる JavaScript など，異なる特性を持つ言語のビルドに対して調査を行う予定である．

謝辞 本研究は，科研費基盤（B）23H03375，基盤（C）21K02862，若手研究 19K20239，および 2023 年度南山大学パッヘ研究奨励金 I-A-2 の助成を受けたものである．

参考文献

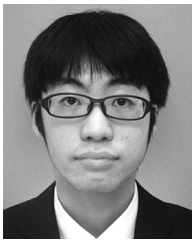
[1] Kerzazi, N., Khomh, F. and Adams, B.: Why Do Automated Builds Break? An Empirical Study, *2014 IEEE International Conference on Software Maintenance and Evolution*, pp.41–50 (online), DOI: 10.1109/ICSME.2014.26 (2014).

[2] Sulír, M. and Porubän, J.: A Quantitative Study of Java Software Buildability, *Proc. 7th International Workshop on Evaluation and Usability of Programming Languages and Tools*, pp.17–25 (2016).

[3] Sulír, M., Bačíková, M., Madeja, M., Chodarev, S. and Juhár, J.: Large-Scale Dataset of Local Java Software Build Results, *Data*, Vol.5, No.3, 86 (2020).

[4] ジョエル・スポルスキー（著），青木 靖（訳）：*Joel on Software*, オーム社 (2005).

[5] Ragkhitwetsagul, C. and Krinke, J.: Using Compilation/Decompilation to Enhance Clone Detection, *2017 IEEE 11th International Workshop on Software Clones (IWSC)*, pp.1–7, IEEE (2017).



小池 耀

1998 年生．2021 年大阪大学基礎工学部情報科学科卒業．2023 年同大学大学院修士課程修了．同年日本信号入社．在学中 OSS の分析に関する研究に従事．

眞鍋 雄貴 （正会員）

2006 年大阪大学基礎工学部情報科学科退学．2011 年同大学大学院情報科学研究科博士課程修了．2011 年同大学院特任助教．2013 年熊本大学自然科学研究科助教．2016 年熊本大学大学院先端科学研究部助教．2020 年福知山公立大学情報学部講師．現在に至る．ソフトウェア工学，特にソフトウェアの再利用，ソフトウェアライセンス，リポジトリマイニングの研究に従事．博士（情報科学）．電子情報通信学会，日本データベース学会，ACM，IEEECS 各会員．

神田 哲也 （正会員）

2016 年大阪大学大学院情報科学研究科博士後期課程修了．同年奈良先端科学技術大学院大学情報科学研究科博士研究員．2017 年大阪大学大学院情報科学研究科特任助教．2018 年より同大学院情報科学研究科助教．博士（情報科学）．ソフトウェア進化，ソースコード解析に関する研究に従事．IEEE 会員．

井上 克郎 （正会員）

1979 年大阪大学基礎工学部情報工学科卒業．1984 年同大学大学院博士課程修了（工学博士）．同年同大学基礎工学部助手．2002 年同大学大学院情報科学研究科教授．2022 南山大学理工学部教授．ソフトウェア工学，特にソフトウェア開発手法，プログラム解析，コードクローン，再利用技術の研究に従事．本会フェロー．

肥後 芳樹 （正会員）

2002 年大阪大学基礎工学部情報科学科中退．2006 年同大学大学院博士後期課程修了．2007 年同大学院情報科学研究科コンピュータサイエンス専攻助教．2015 年同准教授．2022 年同教授．博士（情報科学）．ソースコード分析，特にコードクローン分析，リファクタリング支援，ソフトウェアリポジトリマイニングおよび自動プログラム修正に関する研究に従事．電子情報通信学会，日本ソフトウェア科学会，IEEE 各会員．本会シニア会員．