What is an app store? The software engineering perspective

Zhu, Wenhan; Proksch, Sebastian; German, Daniel M.; Godfrey, Michael W.; Li, Li; McIntosh, Shane

DOI
10.1007/s10664-023-10362-3

Publication date
2024

Document Version
Final published version

Published in
Empirical Software Engineering

Citation (APA)
Zhu, W., Proksch, S., German, D. M., Godfrey, M. W., Li, L., & McIntosh, S. (2024). What is an app store? The software engineering perspective. *Empirical Software Engineering*, *29*(1), Article 35. https://doi.org/10.1007/s10664-023-10362-3

Empirical Software Engineering (2024) 29:35
https://doi.org/10.1007/s10664-023-10362-3

What is an app store? The software engineering perspective

Wenhan Zhu[1] · Sebastian Proksch[2] · Daniel M. German[3] · Michael W. Godfrey[1] · Li Li[4] · Shane McIntosh[1]

Accepted: 22 June 2023

Abstract
"App stores" are online software stores where end users may browse, purchase, download, and install software applications. By far, the best known app stores are associated with mobile platforms, such as Google Play for *Android* and Apple's App Store for iOS. The ubiquity of smartphones has led to mobile app stores becoming a touchstone experience of modern living. App stores have been the subject of many empirical studies. However, most of this research has concentrated on properties of the apps rather than the stores themselves. Today, there is a rich diversity of app stores and these stores have largely been overlooked by researchers: app stores exist on many distinctive platforms, are aimed at different classes of users, and have different end-goals beyond simply selling a standalone app to a smartphone user.
The goal of this paper is to survey and characterize the broader dimensionality of app stores, and to explore how and why they influence software development practices, such as system design and release management. We begin by collecting a set of app store examples from web search queries. By analyzing and curating the results, we derive a set of features common to app stores. We then build a dimensional model of app stores based on these features, and we fit each app store from our web search result set into this model. Next, we performed unsupervised clustering to the app stores to find their natural groupings. Our results suggest that app stores have become an essential stakeholder in modern software development. They control the distribution channel to end users and ensure that the applications are of suitable quality; in turn, this leads to developers adhering to various store guidelines when creating their applications. However, we found the app stores operational model could vary widely between stores, and this variability could in turn affect the generalizability of existing understanding of app stores.

Keywords App store · Software release · Software distribution · Empirical software engineering

Responsible Editor: Tayana Conte

✉ Wenhan Zhu
w65zhu@uwaterloo.ca

Extended author information available on the last page of the article

Published online: 02 January 2024

1 Introduction

The widespread proliferation of smartphones and other mobile devices in recent years has in turn produced an immense demand for applications that run on these platforms. In response, online "app stores" such as Google Play and Apple's App Store have emerged to facilitate the discovery, purchasing, installation, and management of apps by users on their mobile devices. The success of mobile app stores has enabled a new and more direct relationship between app creators and users. The app store serves as a conduit between software creators (often, developers) and their users, with some mediation provided by the app store. The app store provides a "one-stop shopping" experience for users, who can compare competing products and read reviews of other users. The app store might also acts as a quality gatekeeper for the platform, providing varying levels of guarantees about the apps, such as easy installation and removal, expected functionality, and malware protection. To the software creator, the app store provides a centralized marketplace for their app, where potential users can find, purchase, and acquire the app easily; the app store also relieves the developer from basic support problems related to distribution and installation, since apps must be shown to install easily during the required approval process. Indeed, one of the key side effects of mobile app stores is that it has forced software developers to streamline their release management practices and ensure hassle-free deployment at the user's end.

The success of mobile app stores has also led to the establishment of a plethora of other kinds of app store, often for non-mobile platforms, serving diverse kinds of user communities, offering different kinds of services, and using a variety of monetization strategies. Many technical platforms now operate in a *store-centric* way: essential services and functionality are provided by the platform while access to extensions/add-ons is offered only through interaction with the app store. For instance, Google Play, the app store, operates on top of the technical platform *Android*, which provides the runtime environment for the applications. When new technical platforms are introduced, an app store is often expected to serve as a means to host and deliver products to its users (Dixon et al. 2010). Example technical platforms that use app store-like approaches include Steam (Valve 2022), GitHub Marketplace (GitHub 2022), the Chrome Web Store (Google 2022a), WordPress (WordPress 2022), AutoDesk (Autodesk 2022), DockerHub (Docker 2022), Amazon Web Services (AWS) (Amazon 2022), Homebrew (Prévost et al. 2022), or Ubuntu Packages (Canonical 2009).

For platforms that operate in this way, the app store is an essential part of the platform's design. For example, consider source code editors, such as VSCode and IntelliJ. The tool itself — which we consider to be a technical platform in this context — offers the essential functionality of a modern source code editor; however, many additional services are available through the associated app store that are not included by default. Thus, extensions that allow for language-specific syntax highlighting or version control integration must be added manually by the user through interaction with the tool's app store. We conjecture that the app store has fundamentally changed how some classes of software systems are designed, from the overall ecosystem architecture of the technical platform to the way in which add-ons are engineered to fit within its instances.

In this work, we will explore the general space of app stores, and also consider how app store-centric design can affect software development practices. Previous research involving app stores has focused mainly on mobile app stores, often concentrating on properties of the apps rather than properties of the stores. For example, Harman et al. performed one of the first major studies of app stores in 2012, focusing on the BlackBerry App World (Harman et al. 2012). However, concentrating the investigative scope so narrowly may lead to

claims that do not generalize well across the space of all app stores. For example, Lin et al. found that reviews of games that appeared in mobile app stores differed significantly from the reviews of the same game that appeared within the Steam platform's own app store (Lin et al. 2019). In our work, we aim to take a more holistic approach to studying app stores by considering both mobile and non-mobile variants. In so doing, we hope to create a more general model of app stores that fits this broader space.

To achieve a holistic view, we start from the definition of an app store. A precise definition of the term "app store" has been omitted in much of the previous research in this area. Currently, Google Play and Apple's App Store dominate the market and are the main targets of research on app stores; in the past, the BlackBerry App World and Microsoft's Windows Phone Store were also important players, but these stores are now defunct.[1] Wikipedia recognizes Electronic AppWrapper (Wikipedia 2022) as the first true platform-specific electronic marketplace for software applications, but the term became popular when Apple introduced its App Store along with the iPhone 3G in 2008. Since then, the term has largely come to refer to any centralized store for mobile applications. We present our own working definition of the term "app store" in Section 2.4.

The goal of this work is to survey and characterize the broader dimensionality of app stores, and also to explore how and why they may feed back into software development practices, such as release management. As a step towards this goal, we focus on two research questions (RQs) that aim to explore the space of app stores:

RQ1: *What fundamental features describe the space of app stores?*

To understand app stores, we first need a way to describe them. It would be especially useful if this description framework would highlight the similarities and differences of app stores. We start by collecting a set of app store examples, and then extract from them a set of features that illustrate important differences between them. We then expand this list of app stores with search queries to derive a larger set of example stores. We explicitly seek generalized web queries to broaden our search space beyond the common two major mobile app stores of *Apple* and *Google*. By combining the web queries and the initial set of app stores, we selected a representative set of app stores and extracted their features. In the end, we first surveyed app stores and derived a feature-based model to describe them; we then expanded the set of app stores through web queries; and finally, we extracted features based on the model for a representative set of app stores.

RQ2: *Are there groups of stores that share similar features*?

Despite the ability to describe individual stores, it is also important to understand the relationships between different stores. Having a understanding of the natural groupings can help us gain insights into the understanding of the generalizability of results gathered for different app stores. We perform a *K-means* (MacQueen et al. 1967) clustering based on the extracted features of the expanded set of app stores collected previously. The optimal k value is determined by the Silhouette method (Rousseeuw 1987). The clustering results suggest that there are 8 groups in the expanded set of app stores. The differences can be observed in the type of application offered, standalone or extension, and/or type of operation, business or community-oriented.

[1] The Windows Phone Store was absorbed into the broader Windows Store in 2015.

In this study, we make several contributions towards a better understanding of the app store ecosystem.

- We identified a set of descriptive features that can be used to characterize app stores.
- We identified a set of 291 app stores and mapped 53 of them into the feature space.
- We identified 8 coherent groups of app stores based on the similarity of features.
- We discuss our insights on how the features and the diversity of app stores can impact software engineering practices.

Overall, our study contributes towards a holistic view of app stores within software engineering, which can form the basis for subsequent study of app stores in general.

2 Background and Related Work

2.1 Early App Store Research

To date, research in this area has concentrated on a narrow set of app stores that primarily involves mobile platforms. Harman et al. (Harman et al. 2012) proposed app stores as a valid kind of software repository worthy of formal study within the broader research area of mining software repositories; while their work was not specific to mobile app stores, they used BlackBerry App World as their canonical example. Ruiz et al. (Ruiz et al. 2012) studied the topic of reuse within app stores, focusing their work on Android Marketplace.[2] In both cases, these early works did not provide a formal definition of "app store", and tacitly used only app stores for mobile platforms in their studies.

In their 2016 survey on app store research, Martin et al. (Martin et al. 2016a) observed that studies have often focused on only a few specific app stores, and have ignored comparisons between app stores. In a recent literature survey, Dabrowski et al. (Dabrowski et al. 2022) found the median number of app stores studied to be 1, with the maximum being 3. We also note that results from one app store study may not generalize to another store since the two stores may differ in significant ways; for example, if a store does not allow users to provide their own reviews of the apps within the store, app creators will have to rely on other means to gain popularity and trust from users, such as promotion outside of the app store. The same trend can be observed in more specific app store topics such as app reviews; for example, Lin et al. (Lin et al. 2019) found that reviews of games within the Steam app store can be dramatically different from reviews of the same game in mobile app stores.

Existing work has yet to explore the full diversity of app stores, concentrating on Google Play and Apple's App Store, and largely ignoring those such as Steam, AWS, and GitHub Marketplace that are not specific to mobile platforms. With the heterogeneity of app stores and their typical uses, we believe that the research in this area can be strengthened by expanding the breadth to encompass a more diverse perspective on app stores; in turn, this breadth can help to validate the generalizability of the study findings.

2.2 App Stores in Recent Software Engineering Research

To better understand the involvement of app stores in recent research, we reviewed relevant recent papers from the two flagship software engineering research conferences: the

[2] Android Marketplace has since been re-branded as Google Play.

ACM/IEEE International Conference on Software Engineering ("ICSE") and the *ACM SIGSOFT International Symposium on the Foundations of Software Engineering* ("FSE") We used *Google Scholar* to find papers containing the keyword "app store" between January 2020 and April 2022 for the two conferences. We found a total of 34 such papers (listed in Table 1). After reading through all of them, we found that each paper fit into one of two broad categories: *mining software applications* (20/34) and *mining app store artifacts* (14/34). We note that our efforts do not constitute a comprehensive literature survey; instead, our goal was to gain an overview of how app stores are involved in recent research, and why app stores matter in their context.

Mining Software Applications App stores have been extensively used as a mining source of software applications. In these papers, the major focus is often on another subject and app stores provide a source where they can collect applications for either a data source or verification dataset. For example, Zhan et al. (Zhan et al. 2021) proposed an approach to detect software vulnerabilities in third-party libraries of *Android* applications. They leveraged the app store to collect a dataset to verify the effectiveness of their approach. In these studies, the app store is both a convenient and practical source of data collection. However, the involvement of app stores may not be necessary since the purpose is to gather a dataset of application. In Yang et al.'s work (Yang et al. 2021), they leveraged *Android* applications from an existing dataset without the need to collect from an app store. We argue that the importance of app stores in these types of studies is the selection criteria used by the researchers to collect applications from app stores. These features can include star ratings, total downloads, and app category.

Mining App Store Artifacts In these studies, researchers focused on unique software artifacts that come from the operation of the app stores. App stores have a much heavier involvement in these studies compared to the previous group. App reviews is the major software artifact the researchers focused on, where they leverage the data to identify features of applications (Wu et al. 2021), locating bug reports (Haering et al. 2021), and detect undesired app behaviors (Hu et al. 2021). One interesting research practice we observed is where van der Linden et al. (Van Der Linden et al. 2020) leveraged the developer contact information shared on app stores to send out surveys related to security practices.

2.3 Store-Focused Research

As stated above, we found that most recent research involving app stores focuses on the applications they offer rather than on studying the app stores themselves; in particular, most research in the domain focuses on the development of mobile applications. Meanwhile, a few papers have specifically considered app stores and their effects on software engineering, but again these works focus heavily on mobile app stores.

In a recent paper, Al-Subaihin et al. (Al-Subaihin et al. 2021) interviewed developers about how app stores affect their software engineering tasks. They found that developers often leverage the review section from similar applications to help with understanding the expected user experience and anticipated features. App stores also provides a kind of playground for releasing beta version of apps to receive feedback from users. The built-in communication channels also play a large role in informing development. The interviews suggest that developers pay attention to viewing user requests in app store via channels such

Table 1 Recent papers on app stores

Loc	Paper	Store
Mining software applications		
ICSE '21	Atvhunter: Reliable version detection of third-party libraries for vulnerability identification in android applications (Zhan et al. 2021)	Google Play
ICSE '20	How does misconfiguration of analytic services compromise mobile privacy? (Zhang et al. 2020)	Google Play
FSE '21	Algebraic-datatype taint tracking, with applications to understanding Android identifier leaks (Rahaman et al. 2021)	Google Play
FSE '20	Code recommendation for exception handling (Nguyen et al. 2020)	Google Play
FSE '20	Static asynchronous component misuse detection for Android applications (Pan et al. 2020)	F-Droid, Google Play, Wandoujia App Store
ICSE '21	Sustainable Solving: Reducing The Memory Footprint of IFDS-Based Data Flow Analyses Using Intelligent Garbage Collection (Arzt 2021)	Google Play
ICSE '22	DescribeCtx: Context-Aware Description Synthesis for Sensitive Behaviors in Mobile Apps (Yao and Xiao 2022)	Google Play
ICSE '20	Time-travel testing of android apps (Dong et al. 2020)	Google Play
ICSE '20	An empirical assessment of security risks of global android banking apps (Chen et al. 2020a)	Google Play, APKMonk, and others
ICSE '21	Too Quiet in the Library: An Empirical Study of Security Updates in Android Apps Native Code (Almanee et al. 2021)	Google Play
ICSE '20	Accessibility issues in android apps: state of affairs, sentiments, and ways forward (Alshayban et al. 2020)	Google Play
ICSE '21	Dont do that! hunting down visual design smells in complex uis against design guidelines (Yang et al. 2021)	Android
ICSE '21	Identifying and characterizing silently-evolved methods in the android API (Liu et al. 2021)	Google Play
ICSE '21	Layout and image recognition driving cross-platform automated mobile testing (Yu et al. 2021a)	Apple's App Store, Google Play
FSE '21	An empirical study of GUI widget detection for industrial mobile games (Ye et al. 2021)	Android Games
ICSE '21	Fine with 1234? An Analysis of SMS One-Time Password Randomness in Android Apps (Ma et al. 2021)	Google Play, Tencent Myapp
ICSE '21	IMGDroid: Detecting Image Loading Defects in Android Applications (Song et al. 2021)	Android
ICSE '21	GUIGAN: Learning to Generate GUI Designs Using Generative Adversarial Networks (Zhao et al. 2021)	Android
ICSE '20	Unblind your apps: Predicting natural-language labels for mobile gui components by deep learning (Chen et al. 2020b)	Google Play
FSE '21	Frontmatter: mining Android user interfaces at scale (Kuznetsov et al. 2021)	Google Play

Table 1 (continued)

Loc	Paper	Store
Mining app store non-technical attributes		
ICSE '20	Schrödinger's security: Opening the box on app developers' security rationale (Van Der Linden et al. 2020)	Apple's App Store, Google Play
ICSE '20	Scalable statistical root cause analysis on app telemetry (Murali et al. 2021)	Facebook App
ICSE '21	An empirical assessment of global COVID-19 contact tracing applications (Sun et al. 2021)	Android
ICSE '21	Well Fix It in Post: What Do Bug Fixes in Video Game Update Notes Tell Us? (Truelove et al. 2021)	Steam
ICSE '21	Automatically matching bug reports with related app reviews (Haering et al. 2021)	Google Play
ICSE '21	Prioritize crowdsourced test reports via deep screenshot understanding (Yu et al. 2021b)	Android
ICSE '21	A first look at human values-violation in app reviews (Obie et al. 2021)	Google Play
ICSE '21	Does culture matter? impact of individualism and uncertainty avoidance on app reviews (Fischer et al. 2021)	Apple's App Store
ICSE '21	COVID-19 vs social media apps: does privacy really matter? (Haggag et al. 2021)	Google Play, Apple's App Store
ICSE '20	Society-oriented applications development: Investigating users values from bangladeshi agriculture mobile applications (Shams et al. 2020)	Google Play
FSE '21	Checking conformance of applications against GUI policies (Zhang et al. 2021)	Android
ICSE '21	Identifying key features from app user reviews (Wu et al. 2021)	Apple's App Store
ICSE '21	Champ: Characterizing undesired app behaviors from user comments based on market policies (Hu et al. 2021)	Google Play, Chinese android app stores
ICSE '20	Caspar: extracting and synthesizing user stories of problems from app reviews (Guo and Singh 2020)	Apple's App Store

as reviews and forums. The approval period of app stores affects how developers plan their release. App stores introduce non-technical challenges in the development process. Given the app store model of release, app store-specific metrics, such as total number of downloads, are considered highly important to developers.

Running an app store presents both technical and non-technical challenges to the store owner. Technical challenges include verifying that each app will install correctly, while non-technical challenges include ensuring that the promotional information in the app's product page adheres to store guidelines. Wang et al. (Wang et al. 2018) investigated several *Android* app stores in China and compared them to Google Play. Their study showed that these stores were much less diligent in screening the apps they offered, with a significantly higher presence of fake, cloned, and malicious apps than Google Play.

Jansen and Bloemendal surveyed the landscape of app stores from the perspective of the business domain (Jansen and Bloemendal 2013). They selected 6 app stores — 5 mobile stores and 1 *Windows* store — at the time of publication (Jansen and Bloemendal 2013), and investigated each store manually to find *features* (i.e., those actors can interact with) and *policies* (i.e., rules, regulations and governing processes that limit the functional reach of the features) from each app store. The actors they define are the same as the three major stakeholders of the app store model (i.e., the *store owner*, *users*, and *developers*). Our study further contributes to the understanding of app stores. First, we studied a significantly larger set of app stores: our methodology was focused towards the identification of as many different types of stores as possible. In total, we studied 53 stores in various domains including mobile, embedded systems, computer games, application add-ons, and open source distributions and packaging systems. Second, Jansen and Bloemendal studied app stores from the perspective of a software business; for example, in their work they would consider features and policies on whether users are able to generate affiliate links to earn revenue through sharing applications. In contrast, our work focuses on app stores in the perspective of their role in the software engineering process.

In our study, we approach app stores from a broad landscape not limiting to mobile app stores. We focus on the similarity of features offered between stores to understand their natural groupings and discuss the challenges in the diversity of app stores.

2.4 Working Definition of an App Store

Previous researchers have often taken a casual approach to defining the term "app store", when a definition has been provided at all. For example, in their survey paper, Martin et al. define an app store as "A collection of apps that provides, for each app, at least one non-technical attribute", with an app defined as "An item of software that anyone with a suitable platform can install without the need for technical expertise" (Martin et al. 2016a). However, we feel that this definition is too generous. For example, consider a static website called Pat's Apps that lists of a few of someone's (Pat's) favorite applications together with their personalized ratings and reviews; superficially, this would satisfy Martin et al.'s requirements as it is a collection of apps together with Pat's own reviews (which are non-technical attributes). We feel that this kind of "store" is outside our scope of study for several reasons: Pat's software collection is not comprehensive, it is unlikely that Pat provides any technical guarantees about quality of the apps, and a passive list of apps on a web page does not constitute an automated "store".

Jansen and Bloemendal (Jansen and Bloemendal 2013) define app store as "An online curated marketplace that allows developers to sell and distribute their products to actors

within one or more multi-sided software platform ecosystems." We note that this definition ignores that app stores are expected to provide infrastructure for the deployment, installation, and maintenance of the apps, which impacts the software development process. Their model also ignores marketplaces that do not have payment mechanisms, such as the Google Chrome Extensions store and the various open source apps stores, where all of the software products may be free to download and install.

In our work, we seek to define an idea of app store beyond the well-known mobile ones and with an emphasis on how their existence may affect the software development cycle. Because we are focused on exploring the notion of what app stores are, we formulate a working definition of the term; we did so to provide clear inclusion/exclusion criteria for the candidate app stores that we discover in Section 3.

Our working definition was influenced by considering the three major stakeholders of the app store model: the *app creators* who create and submit applications to the store; the *app stores* themselves, and the organizations behind their operation who curate the app collection and coordinate both the store and installation mechanisms; and the *end users* who browse, download, review, and update their applications through the app store (see Fig. 1).

We thus arrived at the following working definition for *app store* as an online distribution mechanism that:

1. offers access to a comprehensive collection of software or software-based services (henceforth, "apps") that augment an existing technical infrastructure (i.e., the runtime environment),
2. is curated, i.e., provides some level of guarantees about the apps, such as ensuring basic functionality and freedom from malware, and
3. provides an end-to-end automated "store" experience for end users, where

(a). the user can acquire the app directly through the store,
(b). users trigger store events, such as browsing, ordering, selecting options, arranging payment, etc., and
(c). the installation process is coordinated automatically between the store and the user's own instance of the technical platform.

We can see that using this working definition, our Pat's Apps example fails to meet all three of our main criteria.

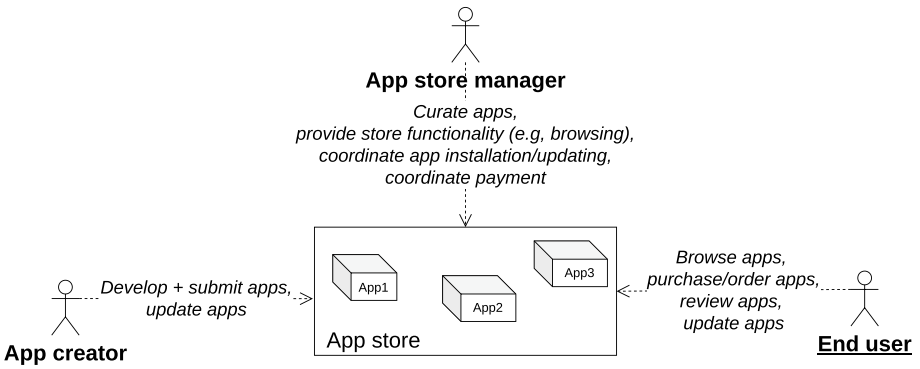

Fig. 1 Three major stakeholders of most app stores

We note that our working definition above evolved during our investigations; it represents our final group consensus on what is or is not an app store for the purposes of doing the subsequent exploratory study. The steps by which the representation is finalized are discussed in Section 3.1.2. For example, our working definition implicitly includes *package managers* such as the Debian-Linux apt tool and Javascript's NPM tool. It is true that package managers are typically non-commercial, and so are "stores" only in a loose sense of the term; furthermore, they usually lack a mechanism for easy user browsing of apps and do not provide a facility for user reviews. However, at the same time, they are a good fit conceptually: they tend to be comprehensive, curated, and offer an automated user experience for selection and installation. Furthermore, some package managers serve as the backend to a more traditional store-like experience; for example, the Ubuntu Software Center builds on a tool *aptitude*, which interacts with software repositories to provide a user experience similar to that of Google Play.

3 Research Methodology

To investigate the research questions, we designed a three-stage methodology that is illustrated in Fig. 2. The goal of the first two stages is to answer RQ1, while the third stage addresses RQ2.

In the first stage (Step 1 and 2) we identified our initial list of features using a small set of well-known app stores (Apple's App Store, Google Play, Steam etc.) In the second stage (Steps 3, 4, and 5) we methodically expanded our list to a conceptually wider ranging set of 53 app stores. We then described these stores using the features identified in the first stage. A major goal of this stage was to evaluate whether the set of available features was sufficient to describe the characteristics of all these stores. This set of features forms the answer to RQ1.

In the third stage (Step 6), we took advantage of the labeling of the 53 stores. We used *K-means* clustering analysis to identify groups of stores that shared similar features. These groupings form the answer to RQ2.

We now describe our methodology in more detail.

3.1 Extracting Features Describing App Stores

Our basic assumption is that an app store can be categorized based on a finite set of features. The features would correspond to traits of the app store where they describe the distinguishing qualities or functional characteristics of the app store. We encode these features as binary values, i.e., each store *has* or *does not have* a given feature.

In order to identify such features, we first created a seeding set of representative app stores. We started by enumerating well-known app stores that we were aware of (Step 1). Once this set of representative app stores was created, we used an iterative process to identify the features that we felt best characterized these stores (Step 2). We then used these features to describe each store.

3.1.1 Stage 1: Identifying Features

First, each of the six authors was tasked with identifying representative characteristics of five stores and the possible features for each. Each author worked alone in this step;

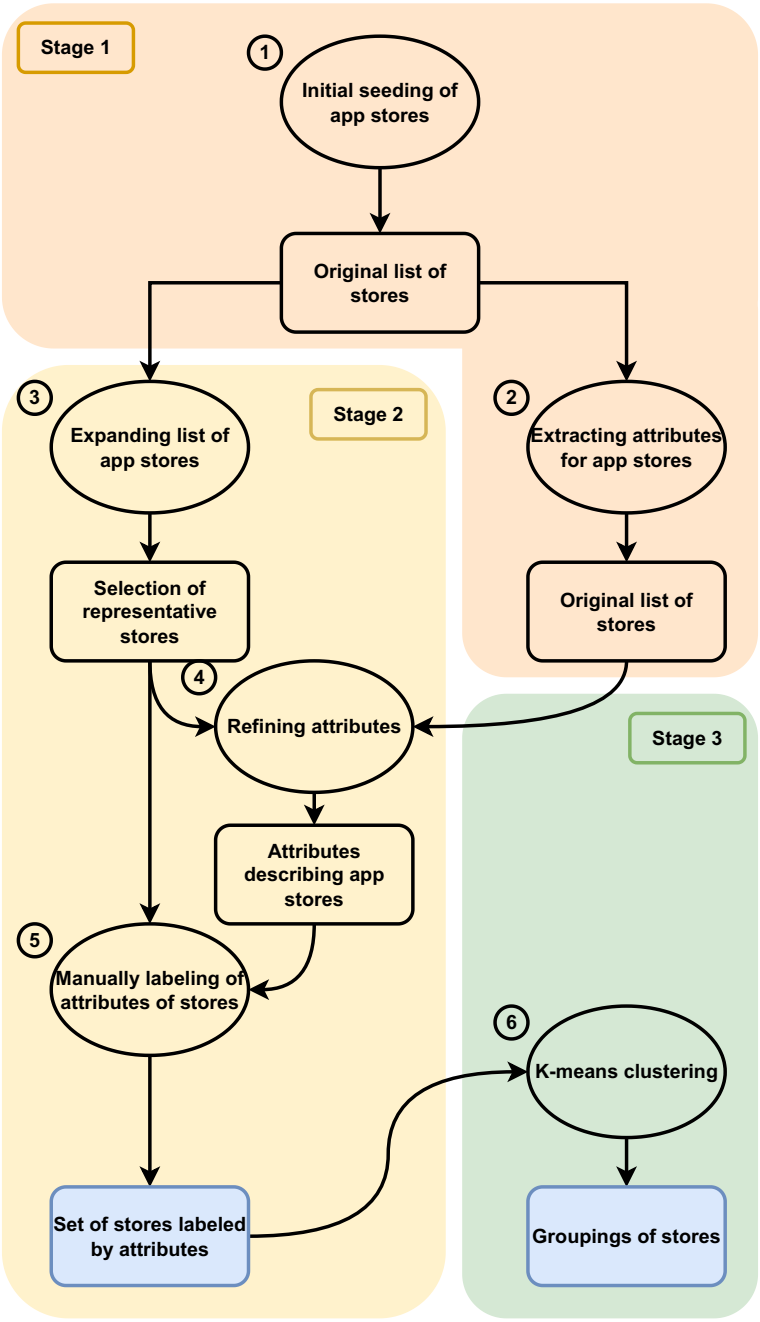

Fig. 2 Methodology overview: There are three main *stages*, further broken down into six *steps*

however, to seek better reliability as well as encourage diverse opinions, each store was assigned to two authors. We list the 15 stores that were assigned in this step with a short description in Table 2. After that, all of the authors met as a group to discuss their findings and further refine the proposed feature set.

In the subsequent iterations, the authors worked in pairs, and the pairings were reassigned after each iteration (Step 2). In these iterations, each author-pair was assigned a set of 2–3 app stores and was asked to describe them using the current set of features; a key concern was to evaluate whether the existing features were sufficient or needed refinement. For each store, each author-pair analyzed both its store-front and its documentation; in some cases, we could navigate the store as users but not as developers, in these cases, we relied on the store's supporting documentation.

After this step, the six authors discussed their findings as a group and updated the set of features. The features were discussed in detail to ensure that they were conceptually independent from each other. We also made sure that each feature applied to at least one store to ensure that it was relevant.

Our process leveraged ideas from the coding process of *Grounded theory* (Walker and Myrick 2006) to extract the features of app stores, and followed the practice of open card sorting (Coxon et al. 1999) to create the categorized feature set. Similar to prior work (Adolph et al. 2011; Hoda et al. 2012; Masood et al. 2020), we followed practices of *Grounded theory*'s coding process to extract the features — where we consider codes as a specific feature of app store operation — and stopped when we reached saturation with no new features added after a new round of describing app stores. Similar to prior work (Vassallo et al. 2020; Chen et al. 2021; Wang et al. 2022), we applied card sorting to the collected features so inter-related features are grouped together. The authors formed a group in this process and discussed how different features belong to the same conceptual group and stopped when consensus was reached.

Table 2 Investigated stores for feature extraction

Store	Description
Google Play Store	Google's app store for Android
Apple App Store	Store for Apple devices
Samsung GalaxyApps	Store specifically for Samsung devices
GitHub Marketplace	Providing applications and services to integrate with GitHub platform
Atlassian Marketplace	Providing applications and services to integrate with various Atlassian products
Homebrew	Package manager for MacOS
MacPorts	A package manager for MacOS
Ubuntu Packages	Software repository for the Ubuntu Linux distribution, with a official front end Ubuntu Software Center
Steam	Gaming focused app store running on multiple operating systems (e.g., Windows, Linux)
Nintendo EShop	Provides applications for Nintendo devices (e.g., Nintendo Switch, Nintendo 3DS)
GoG	Gaming focused store focusing on providing DRM free games
JetBrains Plugin Store	Provides plugins to enhance the behavior of JetBrains IDEs
VSCode Marketplace	Provides plugins to enhance the editor
Chrome Web Store	Provides extensions to enhance Chromium based web browsers
AWS Marketplace	Provides servers and cloud services

3.1.2 Stage 2: Expanding our Set of App Stores and Further Evaluation and Refinement the Features

Once we had agreed on the features, our next goal was to verify that these features were capable of describing other app stores that were not part of the initial seed, or if features were missing or needed refinement. We used a common search engine, *Google*, to expand our set of app stores in a methodical manner (Step 3). To achieve the goal of including a broad range of yet undiscovered app stores, we first derived general search terms by combining synonyms for "app" and "store". More specifically, we have built all possible combinations of the following terms to construct our search queries:

First half software, (extension -hair -lash), (addon OR add-on), solution, plugin OR plug-in, install, app, package
Second half repository, shop, ("app store" OR store), ("market place" OR marketplace), manager

For example, a concrete query was created by combining app and ("app store" OR store). For some queries, it was necessary to refine the term to avoid noise in the results; for example, searching for the term extension would mainly return results related to hair product or eye lashes. In total, with 8 synonyms for app and 5 synonyms for store we were able to create 40 unique *Google* search queries. We felt confident that these search terms were representative when we found that the initial seed list had been exhaustively covered.

Our *Google* search was performed in November 2020. We queried and stored the search results for each search query. Two authors classified each result as to whether or not it corresponded to an app store. We devised two inclusion criteria for this decision: 1) the store in question should offer software or software-based services, and 2) the store in question should offer an end-to-end experience for users (ordering, delivery, installation). We considered only direct hits to the store (e.g., product page), and we explicitly excluded results that contain only indirect references to a store, such as blog posts, videos, or news. Any disagreements were resolved through discussion. However, despite our initial effort of maintaining a clear set of inclusion criteria for app stores, several corner cases became apparent during the labeling process. The first two authors discussed these cases as they arose, and continually updated the inclusion criteria throughout the labeling process. In a few special cases no agreement could be reached, so another author acted as a moderator and resolved the disagreement by a majority vote. Over time, the inclusion criteria and features evolved and eventually reached a stable state (in Step 3). Our final state of the inclusion/exclusion criteria is presented as the working definition for app stores defined in Section 2.4.

The classification of search results was stopped when a new results page did not contain any new links to app stores, or once all 10 retrieved pages were analyzed. Initially, 586 URLs were examined by the first two authors until a saturation of agreement was reached (90.7% agreement rate). The first author continued to label the rest. In the end, a total of 1600 URLs were labeled. Multiple search results can refer to the same store; these duplicates were detected and eliminated by using the root domain of the URL. The most common duplicate references were found for the domains *google.com* (61), *apple.com* (22), and *microsoft.com* (18). In the end, we found 291 stores. We note that the exact number of unique stores may differ since two root domains can point to the same store, *kodi.tv* and *kodi.wiki*, or the same root domain may contain multiple stores, *chrome.google.com* and *play.google.com*.

In the next step (Step 5), we constructed and labeled a set of app stores based on our identified features from Step 2. We began from the URLs labeled in the last step and selected the first three occurring stores for each search term; this resulted in 104 URLs pointing to 48 unique stores. Two of the stores were could not be accessed by the authors: ASRock App Shop requires physical hardware to use it, and PLCnext Store's website was unresponsive at the time of labeling. These stores were removed from the list. In addition, we discussed several more stores that we felt deserved explicit investigation: AWS, Flatpak, GoG, MacPorts, Nintendo eShop, Steam, and Samsung's Galaxy Store. These are the stores that the authors investigated in Step 2 but did not show up in the first three occurring results from the search terms. Meanwhile, the added stores all show up in the list of 291 stores identified by all labeled URLs.

We thus selected and labeled a total of 53 app stores. This sample is non-exhaustive, but we believe that our wide range of search queries has created a representative sample of the population of app stores that enables our experiments.

The first two authors proceeded to describe 12 app stores, selected as the first from each search query, using the set of features. This was done to make sure there was consistency in the interpretation and use of each feature. After that, the first author labeled the remaining stores.

To check the applicability of our dimensions and the labeling guidelines, we have measured the inter-rater agreement between two authors on the 12 stores. We used the Cohen's Kappa (Cohen 1960) as a measurement for our inter-rater agreement. The Cohen's Kappa is widely used in software engineering research (Pérez et al. 2020). We have reached an agreement of 86.3% with Cohen's Kappa (Cohen 1960) of 0.711). Our agreement based on the Cohen's Kappa is considered as a substantial (Lantz and Nebenzahl 1996) inter-rater agreement suggesting a high confidence of agreement between the two raters.

The outcomes of RQ1 were a list of features that describe the main characteristics of app stores grouped by dimensions, and a set of 53 App Stores, each labeled using these features.

3.2 Finding Natural Groupings of App Stores

With the outcomes of RQ1, we next performed a *K-means* clustering analysis to identify groups of similar stores. *K-means* is a well known clustering algorithm widely used in software engineering research (Pickerill et al. 2020; Khatibi Bardsiri et al. 2014; Al-Subaihin et al. 2019; Kuchta et al. 2018). It groups vectorized data points iteratively until k centroids are formed. We used the *K-means++* implementation (Arthur and Vassilvitskii 2006) to conduct the clustering process.

3.2.1 Stage 3: Cluster analysis

To identify related app stores, we decided to cluster them using the *K-means* algorithm (Step 6).

To prepare our labels for the *K-means* clustering process, we converted each label of the feature to a binary value: 1 if the store has the feature, and 0 if it does not. Having binary-encoded data ensured that we do not suffer from having categorical values that do not make sense in the scope of *K-means*. However, performing *K-means* on binary data can also be

problematic, since the initial centroids selected will be binary. To mitigate this issue, we applied Principal Component Analysis (PCA) (Wold et al. 1987) to both reduce the dimensional space and to produce a mapping in the continuous range. We kept all principal components that explained a variance of at least 0.05. Finally, we used the Silhouette method (Rousseeuw 1987) to determine the best number of clusters within a range of 1 to 20. To identify the features that best characterize each cluster, we have calculated the deviation of each cluster *centroid* (i.e, the *center* of the cluster) from the *centroid-of-centroids* (*C*) over all clusters.

As an unsupervised method, the result of *K-means* provides only the clustering result with the stores in each cluster. We then further discussed the results of the *K-means* process and categorized the clusters by the properties of the contained stores. Following our discussion and categorization, we assigned groupings and names to each of the clusters.

4 Results

In this section, we present the results of our investigations into each of the research questions. The results are organized based on the three stages discussed in Section 3.

RQ1: What fundamental features describe the space of app stores?

Stage 1 Features characterizing app stores.

As discussed in Section 3.1, we derived a set of features and organizational categories that describe the set of studied app stores; the results of these efforts are summarized in Table 3. We have modeled the features as a binary representation; thus, each store either has or does not have this feature. We note that for some categories, the features are mutually exclusive; for example, in the category *Rights Management*, a store can have either *Creator managed DRM* or *Store-enforced DRM*, but not both. In other categories, an app store may have several of the features within a given category; for example, there may be several kinds of communication channels between users, app creators, and the store owner for a given app store. We now describe each high-level category in detail (Table 3).

- *Monetization*-describes what, if any, payment options are provided to the user directly by the store. If a product is offered free within the store, but requires an activation key obtained elsewhere, we consider that the product is free. While most of the options are self-explanatory, some may be less obvious. For example, GitHub Marketplace offers *seat-based subscriptions* where app pricing is calculated by the number of installations made to individual machines; usually, this occurs within the context of enterprise purchase. Also, AWS offers *resource-based subscription* where the price charged is determined by the amount of resources — such as cloud storage and CPU time — that are used during the execution of the service.
- *Rights Management*-describes the Digital Rights Management (DRM) policy of the store; the values describe whether the store uses a store-wide DRM feature. For example, for Steam, all games have DRM encryption, whereas the F-Droid store contains only open source apps, so there is no need for DRM.

Table 3 Features for describing app stores

Feature	Description
Monetization	The type of payment options directly offered by the app store.
Free	Free as in in the product can be directly acquired
One-time payment	A single payment needed for the product
Seat-based subscription	The subscription is based on the number of products provided
Time-based subscription	A payment is needed by a set time interval (e.g.,, monthy, yearly)
Resourced-based subscription	A payment is needed by the amount of resource used (e.g., API calls, CPU time)
Micro-transaction	Additional payment can be collected based on additional feature offered in a product
Custom pricing (i.e., "Contact us for price")	The actual price is based on a per case situation; this happens mostly in business-focused app stores
*Rights Management**	How does the store take care of DRM on the product provided.
Creator-managed DRM	No DRM is offered by the store and is taken care of by the creator
Store-enforced DRM	Store wide DRM for every product offered in the store
*Do I need an account?**	Whether it is possible to use the app store without registration.
Account required	An account is required to use the store
No registration possible	The store does not have an account system
Some features requires registration	Some content of the store is locked behind an account, but the store can be used without one.
Product type	The type of product the store offers.
Standalone apps	The product operates by itself
Extension/add-ons to apps/hardware	The product acts as a feature extension to another application/hardware
Service/resources	The software product is a service
Package/library	The product is not an end-user product, but offers functionality to other products
*Target audience**	The intended users of the app store.
General purpose	The app store is intended to be used by everyone.
Domain-specific	The app store have a specific focus and is very unlikely to be used by a normal person
Type of product creators	The type of creators who submits products to the app store.
Business	The creators mostly have a commercial or business focus
Community	The creators are from the community (e.g., open source developers)
Intent of app store	The reason why the app store exists from the app stores' perspective.
Community building/support	The app store aims to serve a technical community
Profit	The app store aims to earn money
Centralization of product delivery	The app store aims to provide a way for customer to gather apps in a centralized way

Table 3 (continued)

Feature	Description
Expanding a platform popularity/usefulness	The app store aims to extend functionality from the platform it is based on
Role of intermediary	The role app store play between the creator of products and the customer of the app store.
Embedded advertisement API	Provides an advertisement method for creators to take advantage of
CI/CD	Offers continuous integration/continuous deployment for creators
Checks at run time	Provide checks when apps installed from the app store is ran
Checks before making available to the customer	Provide checks when an app is submitted to the app store for quality reasons
*Composability**	The relationship between products provided in the app store.
Independent	The products in the app store are unrelated to each other
Vendor internal add-on/extension/unlock	Some products can be based on other products from the same creator (e.g., game DLC, app feature packs)
Package manager type of app relationship	A dependency relationship exists between products in the app store
Analytics	The type of analytical data provided by the app store.
Sentiment and popularity ratings	Information related to the popularity of a product (e.g., downloads, score ratings)
Marketing feedback	Information related to marketing for the creator (e.g., sales, conversion, retention)
Product usage data	Information related to the usage of the product. (e.g., logging, user profiling)
Communication channels	The methods where different parties of the app store can communicate with each other.
Documentation	Information related to the operation of the store (e.g., instructions to install applications)
Product homepage	A homepage for a specific product in the app store
Ratings	Any form of rating customers can give to a product (e.g., star, score, up/down vote)
Written reviews (in text)	A written viewer where customers can write their experience to the product.
Community forum	A forum like feature offered by the store where people can discuss things related to the store/product.
Support ticket	A system where customers can inquiry for support questions related to the product offered by the store.
Promotion/marketing	The store offers a way to provide promotional/marketing feature to the products in the app store (e.g., featured apps, top downloads of the month).

*: Categorical values are mutually exclusive; one and only one categorical value in the dimension can apply to a given store

- *Do I need an account?*-describes whether a user can access and use the store without being registered with the app store. We find that most stores are either *account required* (e.g., Apple's App Store) or *no registration possible* (e.g., Snapcraft). However, we also found that some stores can be used without an account for some purposes, with other features requiring explicit registration; for example, the Microsoft Store allows users to download free applications without an account, but to purchase an app or leave a review, an account is required.
- *Product type*-describes the kinds of applications that are offered by the store. For example, Google Play and Steam focus on *standalone apps*, the VSCode Marketplace store offers *add-ons* to an existing tool, and AWS allows users to "rent" web-based *resources and services*.
- *Target audience*-describes the intended user base of the store. *General-purpose* stores offer products aimed at the broad general public of everyday technology users; this includes stores such as Google Play, Steam, and the Chrome Web Store. *Domain-specific* stores, on the other hand, have a dedicated focus on a specialized field; for example, Adobe Magento focuses on building e-commerce platforms.
- *Type of product creators*-describes the typical focus of creators submitting applications to the store. We distinguish between two groups of creators: those with a commercial or business focus, and those with community focus such as open source developers.
- *Intent of app store*-describes the perceived high-level goals of the app store. The values are derived from the app stores' own descriptions of their goals, often found in *"About us"* web pages. For example, both F-Droid and ApkPure are *Android* app stores; however, F-Droid's focus is to provide a location to download and support FOSS software, while ApkPure's goal is to provide a location for users to be able to download *Android* apps when Google Play may be unavailable.
- *Role of intermediary*-describes the roles that the app store plays in mediating between the users and creators; these are software engineering-related services that are mostly independent of each other. For example, *checks at run time* tracks if the app store ensures that its products function correctly (e.g., Steam tracking game stats). Also, *CI/CD* indicates that the app store provides explicit support for continuous integration and deployment of the apps, which may be linked to specific development tools used by the creator.
- *Composability*-describes the relationship between products offered by the store. App stores of *independent* composability offer products that have no relationship with each other, such as Firefox Add-ons. *Vendor internal add-on/extension/unlock* means that the products within the app store can be based on each other, but only when they are from the same vendor, such as game DLC and micro-transaction unlocks. *Package managers* contain apps that can have complicated dependency relationships regardless of the creator of the products, such as the *Ubuntu* package management tool apt.
- *Analytics*-describes what kind of diagnostic information is provided by the store. We distinguish between three kinds: *Sentiment and popularity ratings* offer user-based information related to store products, such as number of installs in Home Assistant. *Marketing feedback* tracks telemetry information for creators on the performance of their product, such as GitHub Marketplace tracking retention rate for their products for creators. *Product usage data* details the observed usage of the products; for example, Steam tracks the average number of hours users spend on each product.

- *Communication channels*-tracks the types of methods the store directly offers for communications between both users and creators. Since most stores offer a *product homepage* for each of their products, the app creators are largely free to put any information here. This means that if a creator wishes, they can put links to other communication methods external to the store. We do not track such information here since it is product dependent instead of store dependent. While ratings and reviews/comments are often paired together, during our exploration, we found cases where user ratings were permitted but user reviews were not; thus, we have separate values for ratings and reviews. Communication channels can take various forms with different variability, for example, some stores allow responses for reviews. For this aspect, we stay at a high level based on the functionality of the communication channels and consider the variations as detailed implementation for each functionality.

Stage 2 Expanded collection of app stores and labeled set of representative stores.

In stage 1, we identified 53 store candidates. To provide the required data for our experiments, two authors explored these stores to identify which of the *fundamental features* of the previous stage are true for each store. The query results are summarized in Table 4, where we list the search term construction keywords and the first 3 occurrence of stores by the search term. For example, in search term constructed from (addon OR add-on) and ("market place") OR marketplace, the first 3 occurrences are Google Play, PrestaShop, and CS-Cart.

There are many app stores beyond Google Play and Apple's App Store. These app stores exhibit a diverse set of features.

RQ2: *Are there groups of stores that share similar features?*

Using the labeled data of the 53 stores, we were able to perform the *K-means* cluster analysis that we have introduced in Section 3.2. With the number of clusters guided by the Silhouette method to choose the best K value for *K-means*, our clustering resulted in eight clusters.

Due to the nature of unsupervised methods, *K-means* is able to identify only the clusters and their members; no real-world meanings are extracted for why the cluster members belong together. It is also important to note that the *K-means* algorithm performs *hard clustering*; that is, it creates a partitioning of the stores into mutually exclusive groups that together span the whole space. Thus each store will be assigned to the unique cluster that the algorithm considers to best represent it. For this reason, the raw results from *K-means* should not be seen as authoritative, but rather as a vehicle for identifying groups of stores with similar characteristics. Therefore, we leverage the *K-means* clustering and further examine the clusters in detail to try to derive a human understandable categorization of the stores.

We start by analyzing the differences between clusters by analyzing the definitive characteristics of each cluster. In Table 5, we show the details of the top 10 features that deviate the most from the C. Column C contains the *centroid-of-centroids* with values for each

Table 4 First three identified stores for each *Google* query

	("app store" OR store)	("market place" OR marketplace)	shop	repository	manager
app	Apple App Store, Google Play	BigCommerce, Google Play, HubSpot	Apple App Store	F-Droid, Guardian Project, IzzyOnDroid	Google Play
software	Mac App Store	MarketPlaceKit, Sellacious, CS-Cart	ϕ	ϕ	ϕ
(addon OR add-on)	Mac App Store, Home Assistant, Firefox Add-ons	Google Play, PrestaShop, CS-Cart	PrestaShop, Chrome Web Store	Kodi	CurseForge, Ajour, Minion
(plugin OR plug-in)	Google Play, SketchUcation, RICOH THETA	WordPress, JetBrains	Bukkit, Plugin Boutique	WordPress, JetBrains	Jenkins, JMeter, Autodesk
(extension -hair -lash)	Chrome Web Store, Microsoft Edge	VSCode Marketplace, Adobe Magento, Chrome Web Store	Chrome Web Store	TYPO3, GNOME SHELL	Chrome Web Store
install	Google Play, Apple App Store	Google Play, Eclipse	Apple App Store, Google Play, Microsoft Store	Kodi, Home Assistant, DockerHub	Google Play, APKPure, Daz3D
solution	Mac App Store, Microsoft Store	CS-Cart	ϕ	ϕ	ϕ
(software library -book)	Microsoft Store	VSCode Marketplace, QT Extensions, GitHub Marketplace	ϕ	ϕ	ϕ
package	Apple App Store, Google Play, Snapcraft	CS-Cart, concrete5	Google Play	Packagist, PyPI, Ubuntu Packages	Chocolatey, NPM, NuGet

feature. The remaining columns represent each cluster by an index from 1 to 8. The values in these columns represent the proportion of app stores in the cluster with a specific feature, the mean, and the background color of each cell represent the deviation of the particular cluster *centroid* (i.e., difference between the centroid of this cluster and the centroid-of-centroids for the feature). Each row corresponds to a feature of the stores, which makes it easy to understand which features are descriptive of a cluster (Table 5).

Table 5 The 8 clusters found by the *K-means* algorithm, with top deviated features from the centroid of centroids (C). Each cell with a value represents one of the influential features of the corresponding cluster. The number indicates the percentage of stores with the specific feature. The color encodes whether stores in that cluster less (magenta) or more (green) likely to have the feature, compared to the centroid

		Cluster Index							
Features	**C**	1	2	3	4	5	6	7	8
Monetization									
Free	1.00								
One-time payment	0.35	0.00		0.00	0.00				1.00
Seat-based subscription	0.09		0.50						
Time-based subscription	0.30		0.75	0.00					0.86
Resource-based subscription	0.05								
Micro-transactions	0.11								0.86
Custom Pricing	0.01								
Rights Management									
Creator managed DRM	0.72		0.25	1.00					0.14
Store-enforced DRM	0.27		0.75						0.86
Do I need an account to use the store									
Account Required	0.33			0.00		0.75	1.00		0.86
No registration possible	0.35	1.00	0.00		0.00		0.00	1.00	
Some features require registration	0.30		1.00		1.00				
Product Type									
Standalone apps	0.42		0.00					1.00	
Extension/add-ons to apps/hardware	0.68	0.33						0.00	
Service/Resources	0.08								
Package/Library	0.17	0.89							
Target audience									
General purpose	0.33			0.00	0.83		0.00	1.00	
Domain-specific	0.67			1.00	0.17		1.00	0.00	
Type of product creators									
Business	0.67	0.22		0.00			1.00		
Community	0.67			1.00			0.11		
Intent of app store									
Community building / support	0.52		1.00		1.00	0.00	0.11		
Profit	0.38	0.00		0.00	0.00		0.78	0.00	1.00
Centralization of product delivery	0.84								
Expanding the platform	0.76							0.17	
Role of intermediary									
Embedded Advertisement API	0.16								0.71
CI/CD	0.05								
Checks at run time	0.14		0.50						
Quality/security checks	0.74					0.25			
Composability									
Independent	0.56	0.00			1.00	1.00			0.00
Vendor internal	0.15								1.00
Package manager type	0.19	1.00							
Analytics									
Sentiment and popularity ratings	0.73					0.00		0.33	
Marking feedback	0.25								
Product Usage data	0.33								
Communication channels									
Documentation (wikis, FAQs)	0.81					0.25			
Product homepage	0.97								
Star/Score/Up/Downvote rating	0.57	0.11	1.00		1.00	0.00	1.00	0.00	
Written reviews (in text)	0.47	0.00			1.00	0.00	0.89	0.00	
Community Forum	0.45			0.75		0.00			
Support Ticket	0.35								
Promotion/Marketing	0.71					0.25			

The table only shows the top 10 deviations per cluster (i.e., *column*) to focus on the most important contributors to each cluster. Since all features are binary — each store has or does not have the feature — all values of the centroid-of-centroids are between [0, 1]; thus, a positive deviation (shown with a green background) implies that the stores in the cluster are *more* likely to have the attribute, and a negative deviation (shown with a magenta background) implies that the stores are *less* likely to have the attribute.

For example, for cluster 8 the most important contributor is *[Composability] Vendor internal add-on/extension/unlock* where the centroid of the cluster is 1. When comparing against the centroid-of-centroids (at 0.15), the deviation is at 0.85; this implies that all stores in this cluster have this feature. On the other hand, an example of negative deviation for cluster 1 is the feature *[Composability] Independent* with a centroid of 0 indicating that no stores in this cluster have this feature. Since the centroid-of-centroids for this features is at 0.56, this implies the deviation for stores in this cluster is −0.56.

After the top characteristics that make each cluster distinctive had been identified, we leveraged this information to name and describe each cluster accordingly. Using the information from Table 5 which shows the defining features of each cluster, we derived an organization of the clusters based on several dimensions. The results are described in Table 6.

One important dimension focuses on the type of application served by stores in the cluster. We identified three major types of applications that differentiate the clusters: *General*, where the store offers stand-alone programs that run without the need of specific software (aside from a specific operating system, e.g., Google Play, AWS, Steam); *Extensions*, where the store offers extensions to a specific program or platform e.g., VSCode Marketplace for *VSCode*, Chrome Web Store for *Google Chrome*; and *Package manager*, where the store offers stand-alone programs, but also manages dependency-relationships and requirements between different applications in the store e.g., NPM, MacPorts, Ubuntu Packages. Another dimension in which these clusters can be organized is whether they are Commercial (business-oriented) or Community-managed (no money is involved).

App stores are not all alike. Intuitive groupings emerge naturally from the data. Their differences can be due to the type of application they offer — standalone or extensions — and their operational model, either business- or community-oriented. We found that app stores in different groups of our clustering have different properties, and these properties may have bearing on empirical studies involving app stores.

5 Discussion

In this section, we discuss our findings regarding what we consider app stores to be based on our clustering results, and we describe various research opportunities involving the influence of app stores on software engineering practices.

5.1 What is an App Store?

The term *app store* became popular largely through Apple's App Store, which launched in 2008 along with the *iPhone 3G* (Apple 2008). Other online software stores have also appeared and have had the term applied to them. Originally, the term usually referred to stores of applications for mobile devices, but we have found that today there is ample

Table 6 List of stores and descriptions by cluster, with the example store that is closest to cluster centroid

Type	Stores in Cluster	Example Store	Cluster Description	Index
Extensions				
Commercial specialized	Adobe Magento, AutoDesk, BigCommerce, GoG, HubSpot, Plugin Boutique, Presta Shop, SketchUcation, CS-Cart	*Presta Shop* offers addons to the ecommerce solution platform.	Products in the stores are very domain specific. Creators are mostly business and their store front offers rating systems and written reviews.	6
Community specialized	Bukkit, CurseForge, DockerHub, Home Assistant, Jmeter, Kodi, Minion, VSCode Marketplace	*Kodi* add-on components offers extensions to the *Kodi* entertainment center.	These are community focused stores that offers free products to users. Stores also tailor to a specific domain.	3
Community non-specialized	Apkpure, Chrome Web Store, Eclipse Marketplace, Firefox Add-ons, Gnome, Wordpress	*Wordpress* offers free extensions for users using the wordpress platform.	Products in these stores offers extensions to the platform. Essential operations do not need registration (e.g., installing apps). Products offered in the stores face a generic audience and are independent from each other.	4
General				
Commercial	AWS, Google Play Store, Microsoft Store, Nintendo eShop, Steam, Samsung Galaxy Store, Apple App Store	*MicroSoft Store* offers applications for the windows platform.	Typical stores many people encounter everyday. They run for profit and offer vendor internal products supporting most monetization options.	8
Community	Chocolatey, F-Droid, Flatpak, Guardian Project Repository, IzzyOnDroid, Snapcraft	*F-Droid* is a free and open source software only *Android* application store.	These stores contain standalone free products only. Creators for the stores are mostly from the community and the products are majority open source.	7
Package Manager	Ajour, Jenkins, MacPorts, NPM, NuGet, Packagist, PyPI, Typo3, Ubuntu packages	*Packagist* is the main repository for *PHP* packages.	No account system is involved for these stores. Products are free and most in package style with inter-dependency relationships. Communication channels are also limited with ratings and reviews missing for most stores.	1
Subscription oriented	Github Marketplace, JetBrains, Qt Marketplace, concrete5 marketplace	*Github Marketplace* offers applications and actions to improve the workflow related to *git* repositories hosted on *GitHub*	Often offers subscription services and supports DRM management by the store. Products are not standalone applications and either provide service or extends a platform.	2
Other	MarketPlaceKit, RICOH THETA, Sellacious, daz3D	*Sellacious* is a ecommerce platform and provides extensions to the platform.	They do not have much communication channels offered. Rating and reviews do not exist in the stores. The stores mostly exists to distribute extensions centrally to the platform they are based on.	5

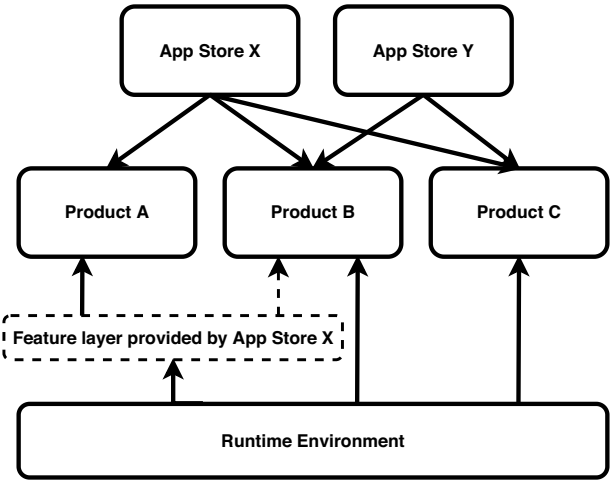

Fig. 3 Stores may offer optional extensions to the runtime environment for applications

diversity of the type of applications that app stores offer and in the features they provide to app developers and users. App Stores are also dynamic: features are continually being added, removed, and altered by store owners in response to changes in their goals and feedback from their socio-technical environments. For example, the Chrome Web Store initially introduced a built-in monetization option that provided a mechanism for applications to receive payments from its users; however, the store later decided to deprecate this monetization option (Google 2022b) and suggested developers to switch to alternative payment-handling options.

In our work, we have employed a working definition through our inclusion/exclusion criteria for app stores to be included in our research. However, due to the complexity, diversity, and constantly evolving nature of app stores, we have decided not to attempt a firm, prescriptive definition of the term. Instead, in the following paragraphs, we will discuss each of several aspects of app stores in detail, and hope that in the future, a more robust definition and operating model can emerge.

5.1.1 Common Features of App Stores

Although we found significant diversity among the example app stores we studied, we were able to identify a set of three common features that appear to span the space of app stores.

- *Simple installation and updates of apps*-An app store facilitates simple installation of a selected application, and can also enable simple updating. For some stores, apps are expected to run on the hardware of the client; in others, the app store provides and manages the hardware where the app runs. In both cases, the app store frees the user from worrying about the technical details of installation, including compatibility with their specific hardware and software configuration, as well as the installation of the app and its dependencies, if any. Typically, app stores will also automate the installation of updates to the application, again freeing the user from worrying about if they have the latest version of the app with the latest features and bug fixes.

- *App exploration and discovery*-App Stores provide mechanisms that allow users to find apps they might want to use. In its simple form, this mechanism might be a search engine that returns a list of apps that match a given set of keywords (such as homebrew, PyPI). In the labeled app stores, 73% of stores provide some kind of aggregated recommendations (e.g., advertisement and trends in WordPress), up to personal recommendations that are based on other apps the user has installed before (e.g., Apple's App Store). User feedback via reviews (present in 47% of the labeled app stores) and forums (present in 45% of the labeled app stores) can provide further information to aid other users in identifying apps of possible interest to them.
- *The app store guarantees the runtime environment*-In practice, app stores often execute within a runtime environment (RTE), such as an operating system (e.g., Google Play on *Android*) or an extensible software application (e.g., Firefox Add-ons on *Firefox*). Many app stores simply sit on top of the RTE, acting primarily as a gatekeeper for adding and deleting apps. However, some app stores are more tightly integrated with the RTE; in extreme cases, the app store can extend the RTE with the app store's own functionality and together provide an augmented RTE for the applications managed through the app store. Steam is a good example for extending the RTE with its own features; developers can integrate with many services offered by Steam, such as an achievement system that offers players recognition when they fulfill certain requirements in the game. Figure 3 illustrates the situation where a product may integrate with additional store-added features to the RTE, which in turn enriches the user experience of the store users. When *Product B* is offered in *App Store Y*, it will not have the features provided by *App Store X*.

The app store ensures that apps are installed only when their runtime requirements are satisfied. The process is often done through running checks on apps submitted to the app store, which 74% of the labeled app stores perform specifically. By specifying the runtime requirements, the assumption for both the developer and the user is that if the application is installed — implying that the requirements are satisfied — it is expected to run properly. This is usually achieved by a software layer on top of the RTE, provided by either the app store or the user. In its simplest form, this software layer is responsible for installing and updating apps (see "Simple installation and updates of apps" above). In some cases, this software layer might also include a set of libraries that the apps can use to provide features specific to the app store thus forming part of the RTE for the applications. These libraries might range in purpose (domain specific, common GUI, resource management, etc.). In extreme cases, this layer includes the operating system, as it is the case with Apple's App Store. However, checks during runtime is a very rare feature, which only 14% of the labeled app stores provides.

Some hardware platforms have become so tightly integrated to the software layer of the app store that they can be considered monolithic: the hardware is rendered unusable without the app store. This is exemplified by the Apple's App Store, where one cannot use the hardware without first having an account in the app store; even operating system upgrades are distributed via the store.

This tight level of integration has clear benefits for all three stakeholders: end users have fewer installation technical details to worry about; app developers can be assured that users will be able to install their apps without the need for technical support; and app store owners can strictly manage who has access to the user's RTE and how. However, such tight integration is technically unnecessary and may even be undesirable. From a software engineering perspective, such tight coupling could be seen as a "design smell", since the operating system and the app store layers address fundamentally different concerns. Also, tight integration can create an artificial barrier to competition, effectively establishing a

quasi-monopoly for the store owner; the store owner may assume the role of gatekeeper not only for streamlining technical issues, but also for business reasons, requiring a kind of toll to be paid by app developers for access to the store. A recent initiative in the European Union (E. Commission 2022) aims to enable fair competition by enforcing that ecosystems are opened up, which will likely also allow the installation of alternative software layers for other app stores, a term called side-loading. In contrast to the *Apple*'s tight control of the operating system as part of its app store, *Android* allows third-party app store software (e.g., F-Droid (F-Droid 2022)) to be installed in co-existence with the system default (often Google Play).

As mentioned above, some stores distribute software that runs on hardware owned by the App Store itself; in these cases, the RTE is fully managed and controlled by the store. For example GitHub Marketplace and Atlassian Marketplace offer applications that run on GitHub and Atlassian servers respectively. In most cases, these applications are not deployed to the user's computers.

5.1.2 Different Types of App Stores

While some features are broadly shared by all app stores, in Section 4, we identified different groups of app stores based on their features. For stores within the same group, they often share common features, whereas across different groups, the stores tend to have less in common. We now discuss the differences across the groups in detail.

Diversity in Goals As a platform focusing on delivering products to customers, the high-level goal of one app store can be dramatically different from the other. Even app stores providing software for the same underlying RTE can have radically different purposes. For example, consider the app stores that run on *Android*. Google Play is the de facto store for *Android* applications. F-Droid store, on the other hand, offers only free and open source *Android* applications, and APKPure offers multiple versions of the same software so the user can decide which version they would like to install.

Apple's app store offers applications for all its RTEs: *MacOS* (laptop and desktops), *iOS* (phones and tables), and the *Safari* browser. In contrast, *Google* has different stores for *AndroidOS* and for its web browser, *Chrome*. The Microsoft Store sells hardware and apps for *Windows* and *XBox*. Alexa Skills offers skills that enhance the voice agent *Alexa*'s capabilities.

In many program language ecosystems, the core language development (focusing on the language features) and packaging system (focusing on extending the functionality of the language) are led by separate organizations (e.g., NPM (npm, "npm About." 2022) and JavaScript (E. International 2022)).

Diversity in Business Model Another important difference we observed is between business-managed and community-managed stores. In business-managed stores (with few exceptions), a primary goal is to generate a profit. These stores provide a payment mechanism between the app creator and the purchaser, with the store keeping a percentage of any sales. These stores have to solve three key concerns: first, implementing registration and authentication of users and developers; second, some type of digital rights management, so only users who have acquired the software can use it; and third, a payment mechanism e.g., subscription, one-time payment, and advertisement.

Community-managed stores, on the other hand, are often run by volunteers, and their features focus on facilitating not-for-profit product delivery from developer to user. Many community stores offer limited community interactions compared to business stores where customer feedback is important. For example, in the Kodi store, add-ons have a web page (e.g., *The Movie Database Python* (Kodi 2022)). This page provides information regarding installation of the add-on, such as known compatibility concerns, download links, and installation requirements. Meanwhile, most communication channels about the add-on are hosted elsewhere; for example, installation and usage instructions, extended descriptions, and screenshots can be found in the community forum instead.

It is important to note that the products contained in community-oriented stores are not limited to open source software; some community-managed app store policies often permit the distribution of proprietary software. In the natural groupings we observed, no rights management is enforced from the store side for Cluster 3; at the same time, most stores in Cluster 8 have some form of rights management built-in to the store. For example, Homebrew permits apps that are not open source if the apps are free to use; these apps might include in-app purchases — such as an upgrade to a full-feature app — that are handled outside of Homebrew.

5.2 Implications for the Main Participant Stakeholders

The results of our study includes an evidence-based detailed view of the broad landscape of app stores. This view can help us improve the understanding of the realities and potentialities of app stores in general. Meanwhile, the results of our work can also benefit the different stakeholders involved with app stores, including app creators, app stores themselves, users, and researchers.

Application Creators Those who create applications — including those who design, develop, test, and market apps — benefit from a holistic view of other stores that will allow them identify potential new markets (stores where they can offer their software) and to understand changing and emerging features that could eventually come to their app store of choice. For new creators, this research emphasizes that a software store has both technical requirements — such as the use of a specific software development kit — and non-technical ones — such as restrictions on what applications can do, approval processes and timelines — and that these requirements vary significantly from one store to another.

App Stores The overview presented herein provides a framework for comparison between app stores, particularly those that operate on the same market, such as *Android* application stores. It can also help promote wide adoption of features that are not universal, such as communication channels between users and developers.

Users With the diversity in app stores, especially when multiple app stores are competing in the same domain, it allows users the choice of where to acquire their applications. This allows for more diversity for how the apps are distributed and the user's choice also affect the competition.

Researchers As discussed in Section 2, most prior research has focused on the applications offered in app stores, and there is a need for research that focuses on studying the store themselves. This emphasis could aid researchers in considering different points of

view when conducting app store-centric studies, and also suggest avenues of exploration concerning how the development process is affected by the existence of app stores.

We describe this point in detail in the next sections.

5.3 App Store Features

In this section, we discuss how each of feature groups from Table 3 has been addressed by current SE research and we suggest some possible future directions.

Monetization App development can be affected by their pricing strategy. For example different software architecture to support a different system of monetization (e.g., locking functionality behind microtransactions) (Pham et al. 2017). Studies have shown a correlation between app features and pricing (Harman et al. 2012; Finkelstein et al. 2017; Sarro et al. 2015). Moreover, in many studies on apps (Alshayban et al. 2020; Aljedaani et al. 2019), free apps and charged apps are often considered as different types of applications. Future work could further explore how different monetization options affect app development.

Rights Management Digital rights management is still an ongoing challenge in software engineering. Existing studies have explored the options of implementing different DRM systems to support developers (Lu et al. 2019; Gaber et al. 2020). DRM can also add challenges in other development activities such as complicating the testing procedure (Sung et al. 2019) and affect performance (Lemon 2018). Often we can observe the store offering means for providing and enforcing DRM. Because DRM is still a nascent technology within software engineering, it remains an open area to explore for future study and how app stores can play a role.

Account Requirement User identity enables telemetry of user behavior. An account system is also the prerequisite of a store-wide DRM system as discussed in the previous paragraph. Existing research has focused on how to leverage the user identity information to create targeted recommender systems (He et al. 2015) and also investigated the concerns of privacy-related issues (Scoccia et al. 2022). The interest of developers (detailed tracing data) and users (privacy) are in conflict, app stores that require user identification could prove to be an excellent study subject for future research in that area.

Product Type Existing research has already shown different software engineering practices based on the software product. For example, gaming development is very different from traditional software development and open source development (Murphy-Hill et al. 2014; Pascarella et al. 2018). Research have shown that different types of software can introduce specific challenges unique to them (Lee et al. 2020a; Ibrahim et al. 2020). Future research should better understand how the product type affects user expectations and development practices, for example, with respect to the delivery of software or the way creators and users can interact.

Target Audience When an app developer decides on a specific app store to sell their app, they are also effectively selecting for a specific type of user (Subramanian et al. 2006;

Manotas et al. 2016; Gholami et al. 2021). Users of a general-purpose store such as Google Play are different and much more diverse than the user population (Guzman et al. 2018) in very specialized stores, such as the add-on store for a particular game. Research needs to understand better which features are relevant in each specific context (Sun et al. 2021), so the experience can be tailored to the concrete situation.

Type of Product Creators Existing research has shown many differences between open source and industrial software development (Pascarella et al. 2018; Lee et al. 2020b). Some studies have touched the aspect of release engineering in open source development (Nayebi et al. 2017a), where developers would strategically select which versions to release on the app store. However, we believe that there is still room for more understanding in how targeting releases towards app stores affects software development.

Intent of App Store While in most domains, there exists a dominant app store, we can also observe situations where multiple app stores compete in the same domain (e.g., game stores on PC, mobile app stores in China (Wang et al. 2018)). In these situations, users have a choice of which app store to use when the same application is offered. In practice, some studies have explored how the high level operation of app stores can affect the software delivery process especially involving security concerns (Sun et al. 2021; Hu et al. 2021). Competition between app stores within the same domain remains largely unstudied, as does how their operations can affect both developers and users.

Role of Intermediary App stores provide a platform for users and developers. Researchers have explored how it affects software development processes such as testing and release management (Nayebi et al. 2016; Shen et al. 2017). There are many opportunities for security (Ferreira et al. 2021) and quality assurance (Al-Subaihin 2012; Tang et al. 2019) to be ensured on the app store side. Future study can explore how the differences between apps managed through an app store and apps that are not. For example, studying the difference between open-source web extensions that are in and not in app stores.

Composability Existing research has explored co-installability in the scope of package manager systems (Vouillon and Cosmo 2013; Claes et al. 2015). However, we only have limited understanding of co-installability for standalone applications in an extension system. For example, if two standalone extensions were to modify the same component of the underlying software, a potential incompatibility could occur. Future research can explore this area by performing empirical studies on existing systems to understand the issue of conflicts.

Analytics App stores as the central hub between developers and users have access to rich information useful for analytics. Previous studies have taken advantage of the app store specific information to help software developers (McMillan et al. 2012; Martin et al. 2016b; Maalej et al. 2019). For example, Ullmann et al. (Ullmann et al. 2022) leveraged records of rating statistics and downloaded information to study the factors in developing successful video games. Another study leveraged analytic information collected by the app store to identify incompatible builds of application and physical devices (Khalid et al. 2014). Future work can explore what are the possible data to collect and form analytics, and how can the analytic data be leveraged to help developers and users.

Communication Channels Communication channels are the most studied area of app store features. Specifically, there has been a heavy focus on app reviews, where researchers have leveraged the information in app reviews to aid software development in areas such as extracting/locating bug reports (Haering et al. 2021), discover feature requests (Wu et al. 2021) and collect user feedback (Guo and Singh 2020). However, existing studies also suggest that the use of communication channels in app stores are often multi-purpose (Dabrowski et al. 2022). Researchers also find that some interaction requirements between interested parties are relegated to other platforms such as Twitter (Nayebi et al. 2017b). Future work can explore different types of communication channels in their functionality and how they can integrate with app stores. The corpora from communication channels are also rich information sources where researchers can leverage to extract information about developer-user interactions.

5.4 Research Opportunities Involving App Stores

App Stores are becoming the primary channel for software delivery and exert considerable influence in many aspects of the software development process. A previous study by Rosen and Shihab (Rosen and Shihab 2016) on Stack Overflow questions by mobile developers has shown that app delivery is one of the biggest challenges developers face. Our results in Section 4 demonstrate that there is a wide variety of types of stores, each with different features and goals. Today, app stores encompass many kinds of applications, from games running on the hardware of the user to add-ons for applications that run on corporate servers such as *GitHub*. However, existing research often focuses heavily on the applications offered inside app stores, especially those of the two major mobile app stores. In the following paragraphs, we discuss several research opportunities to study how app stores can affect software development.

5.4.1 App Stores as Actors in Software Development

App Stores Affect the Software Product Cycle Researchers need to consider how and why app stores can affect the software development life cycle. For example, we know that app stores can constrain and sometimes even dictate software release processes. Some stores go beyond this and exert a kind of socio-technical environmental pressure on other software development practices, becoming a de facto stakeholder in app development. Sometimes these environmental pressures are technical in nature, where the app store might dictate the programming language or deployment platform/OS; some app stores go further and create RTEs, software development kits (SDKs), and user interface (UI) libraries that must be used by all app developers. Sometimes these environmental pressures are non-technical in nature, such as when the app store prescribes the kinds of application that is allowed in the store. For example, *Microsoft* recently announced that it will not permit app developers to profit from open source applications.[3] When an app store operates in a manner such that it has control over what kind of application to include, it creates a software ecosystem and as such, it faces the same challenges that any other ecosystem has: how to thrive. In particular, stores need to

[3] See Update to 10.8.7 https://docs.microsoft.com/en-us/windows/uwp/publish/store-policies-change-history

understand the needs of their developers and users to retain existing ones and attract new ones. However, suggested by what we have observed in Section 4, app stores are diverse with a large number of features that characterize and differentiate between them. While stores are experimenting and evolving, each action is likely to have an effect on the ecosystems they formed, both positively and negatively. Thus, the impact of app stores in the economy and their markets is worthy of further study.

An App may be Offered in Several App Stores Developers want to run their software on the platform that is provided or supported by the store, and as such they must accept the requirements and limitations that such a store may impose. This issue is compounded when the app is being offered in more than one store, as the developers might have to adapt their processes to different sets of requirements, some of which might be conflicting. For instance, an app can be both available in F-Droid (in Cluster 7) and Google Play (in Cluster 8). In Google Play, it is common for applications to collect telemetry data to better understand typical user behavior; however, in F-Droid — an open source and privacy-oriented store — such data collection is highly discouraged. Furthermore, developers must also adapt to the features and limitations that a store provides regarding software deployment, communication with users and — when they exist — the mechanism available to profit from their software and to use digital rights management. This is particularly interesting if the targeted app stores are in different natural groupings. This introduces new areas of studies such as how store policies propagate to applications over time, and how violations of store policies can be detected automatically. Researchers have already begun to investigate this topic through qualitative approaches to identify how applications comply with specific policies that concern accessibility (Alshayban et al. 2020) and human values (Obie et al. 2021).

App Stores Strongly Affect the Release Engineering Process App Stores are especially important in release engineering. Specifically, the release process needs to consider how the application is to be packaged, deployed, and updated. The heterogeneity of the platform provided by RTEs might also affect the number of versions of the application that need to be deployed, e.g., variety of target CPUs, different screen sizes and orientations, and amount of available memory.

When an application is developed for multiple stores, it must effectively be managed as a product line; this is because multiple deliverables must be created, one for each platform-store combination (Wang et al. 2019). Multiple deliverables can also help for telemetry reasons such as tracking the installation source of the application (Ng et al. 2014). The differences between packaged versions might be as significant as requiring the source code to be written in different programming languages, using different frameworks; also, each store is likely to require different deployment processes.

For example, when cross-releasing browser add-ons, developers may have to rewrite part of the functionality in *Swift*/*Objective-C* for better integration with Safari (in the Apple's App Store), while at the same time maintaining a fully *JavaScript* version for Chrome Web Store. Also, the scheduling of release activities is often dictated by the release processes of the stores. A previous study has showed that taking into consideration of app review times is an important factor when planning releases (Al-Subaihin et al. 2021). The app store standardizes, and often simplifies, the release engineering processes for its store; but it also becomes a potential roadblock that might delay or even reject a new release.

5.4.2 The Challenge of Transferring Understanding Between Stores

As noted above, prior work has examined many aspects of app stores, yet the app store itself has rarely been the focus of the research. In many studies, the app store serves as a convenient collection of apps, and the research focuses on mobile development concerns such as testing and bug localization. Even when research focuses on the app store itself, the scope rarely extends beyond Google Play and Apple's App Store. Based on our observations, the diversity of app stores in their operational goals, business models, delivery channels, and feature sets can affect the generalizability of research outcomes. For example, there have recently been many studies (Lin et al. 2019; Dabrowski et al. 2022; Haering et al. 2021; Obie et al. 2021; Fischer et al. 2021; Wu et al. 2021; Guo and Singh 2020) that focus on app reviews. However, for an app store that does not have reviews (e.g., Nintendo eShop) none of the findings and tools can be leveraged (e.g., stores in Cluster 1, 5, and 7).

App Stores that have the same Features may Still Differ Significantly Depending on the problem domain, the details of software development practices can vary dramatically. For example, game development has been compared to both more traditional industrial software development (Murphy-Hill et al. 2014) and to open source software development (Pascarella et al. 2018); in both cases, the development processes can differ greatly. We conjecture that the same may also occur across app stores, where despite the same feature is being offered in the different stores, the convention of using them could be different. As mentioned above, one specific observation has been made between the gaming-focused store Steam and mobile stores (e.g., Google Play) in Cluster 8, where Lin et al. (Lin et al. 2019) found that reviews across the platforms for the same app were often quite different in tone. Such uncertainly invites future research to validate their findings in one store to another to improve the generalizability of the results, and also encourages replication studies to verify existing results on other stores.

A Feature not in the App Store does not mean the Functionality is Missing While some app stores aim to provide a complete experience, where all interactions from the developers and users are expected to be performed within the store, some app stores export part of the work to other platforms. This can even occur for common features that one might find essential. For instance, starred reviews are universal in Cluster 2, 4, and 6 where typical users leverage this information to decide whether an application is good; starred reviews are uncommon for other stores in Cluster 1, 5, 7. The specialized store may have some other metric to indicate popularity or quality, such as total number of downloads, but the focus of the store is often to offer a managed way of installation. Other features, such as application support, are left to other platforms such as social media. Research can further explore the integration between app stores and other platforms.

6 Threats to Validity

Internal Validity Our initial seeding of app stores comes from personal experience of app stores by the authors of the paper. Personal bias could cause us to miss other types of app stores. However, given the number of authors on this paper and our initial effort to consider as many stores as possible, we feel that have created a wide, deep, and collaborative "best effort". When we labeled app stores by their dimensions, it is a qualitative process. As with any qualitative process, the results could be biased by the authors performing the task. We tackled this issue by first labeling a few stores separately by all authors and discussing the

results until a consensus was achieved; thus, we started with a set of "gold standard" labels. Then the labeling task was delegated to two authors who continued to label the stores separately with a portion of the store overlapping. The overlapping labels are then verified by the *Cohen's Kappa* between the two authors to measure the agreement.

We leveraged the *K-means* algorithm for the clustering process. We first applied *PCA* techniques to reduce the dimensions of the initial labeling and provide an orthogonal basis to feed the *K-means* clustering. When using other clustering algorithms (e.g., *Mean-shift*, *DBSCAN*), the clustering result might change; while *K-means* is widely adopted for clustering process in SE research, by nature, determining the proper k value is still a challenge. We followed common best practice to use metrics (i.e., the *Silhouette* method) to determine the best value k. Despite our efforts, the output of the *K-means* clustering is not perfect. We mainly leveraged the *K-means* clustering as the first step to illustrate that app stores forms natural clusters which are different from each other. Based on the *K-means* output, we further grouped the clusters into types based on our qualitative understanding of the app store space.

External Validity During the process of expanding app stores, we relied on the *Google Search Engine* to find web results based on keywords. The results of this step rely on the capability of *Google* and are subject to change over time as *Google* updates its search algorithms. The order may also be affected by SEO operations. Combining results from other search engines (e.g., *DuckDuckGo*, *Bing*) can help to reduce the bias.

When we applied our inclusion criteria, 1) app stores must contain software products and 2) should offer an end-to-end experience for users (ordering, delivery, installation), we excluded stores that focus on digital assets that are not software, such as a pure assets store that offers cosmetic enhancements to desktop environments; we also excluded stores that offer software products but in a way such that installation is completely managed by users. An extreme example, would be the software section of *Amazon* where software is sold as an activation key which users would input to activate the software that they need to install themselves. A more general inspection of all means of distribution software can be performed to gain a broader understanding of software distribution.

We relied on only publicly available information to label each store. So if some functionality (e.g., analytics information) is not documented publicly, we were unable to confirm whether the store has such functionality. We also set a time limit to label each store so in case we were unable to find information about the store, with each store receives the same amount of attention.

One of the main challenges for reproducibility and replicability is that the *Google Search* results and app stores can change overtime. New app stores are likely to emerge and existing app stores may introduce and remove features. The focus of our study is not to establish an exhaustive catalog of app stores, nor to study the historic evolution of a store. Our goal is to establish a framework that can describe app stores and to understand whether the operations of app stores follow different patterns. Based on the granularity which we extracted features from app stores, we expect the majority of the feature groups will remain stable over time. In the future, if researchers would like to repeat our study, the labeling results may differ due to updates in the app store. To mitigate this issue, we have included a snapshot of all *Google Search* results, and documented how we would perform the labeling. So while the final labels may differ, by applying the same process, a replication study would be possible with updated data.[4]

[4] https://zenodo.org/record/7968192

7 Summary

In this paper, we have explored the idea of what an app store is and what features make app stores unique from each other. We labeled a set of representative stores, curated from web search queries, by their features to study the natural groupings of the stores. Our analysis suggests that app stores can differ in the type of product offered in the store, and whether the store is business oriented or community oriented. These natural groupings of the stores challenge the manner in which app store research has largely been mobile focused. Previous studies have already shown empirical differences in activities in mobile app stores and game stores (Lin et al. 2019). Our study further suggests that in the future, when we study app stores, we will need to consider the generalizability of the results across app stores. Since one type of app store may operate under different constraints than another kind, results observed in one app store setting may not generalize to others.

Data Availability A dataset consists of the *Google* query results and the app store labeling results are available on *Zenodo*.

We would like to thank the organizers, administrators, and attendees of the Shonan meeting No. 152 (McIntosh et al. 2019) on "Release Engineering for Mobile Applications", where the paper's idea was conceived.

One of the authors has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 825328 (FASTEN).

Declarations

Conflict of Interests The authors declared that they have no conflict of interest.

References

Adolph S, Hall W, Kruchten P (2011) "Using grounded theory to study the experience of software development," in Empirical Software Engineering, Springer

Aljedaani W, Nagappan M, Adams B, Godfrey M (2019) "A comparison of bugs across the ios and android platforms of two open source cross platform browser apps," in Int. Conf. on Mobile Software Engineering and Systems, IEEE

Almanee S, Ünal A, Payer M, Garcia J (2021) "Too Quiet in the Library: An Empirical Study of Security Updates in Android Apps Native Code," in Int Conf on Software Engineering, IEEE

Alshayban A, Ahmed I, Malek S (2020) "Accessibility issues in android apps: state of affairs, sentiments, and ways forward," in Int Conf on Software Engineering, IEEE

Al-Subaihin A, Sarro F, Black S, Capra L (2019) "Empirical comparison of text-based mobile apps similarity measurement techniques," Empir Softw Eng

Al-Subaihin AA, Sarro F, Black S, Capra L, Harman M (2021) "App store effects on software engineering practices," in Transactions on software engineering, IEEE

Amazon (2022) "AWS Marketplace: Homepage." https://aws.amazon.com/marketplace/. Accessed: Jun. 22, 2022

Apple (2008) "Apple Introduces the New iPhone 3G." https://www.apple.com/ca/newsroom/2008/06/09Apple-Introduces-the-New-iPhone-3G/. Accessed: Jul. 17, 2022

Arthur D, Vassilvitskii S (2006) "K-means++: the advantages of careful seeding," tech rep, Stanford

Arzt S (2021) "Sustainable Solving: Reducing The Memory Footprint of IFDS-Based Data Flow Analyses Using Intelligent Garbage Collection," in Int Conf on Software Engineering, IEEE

Autodesk (2022) "Autodesk App Store : Plugins, Add-ons for Autodesk software, AutoCAD, Revit, Inventor, 3ds Max, Maya ...." https://apps.autodesk.com/. Accessed: Jun. 22, 2022

Canonical (2009) "Ubuntu Software Center in Launchpad." https://launchpad.net/software-center. Accessed: Jun. 22, 2022

Chen J, Chen C, Xing Z, Xu X, Zhut L, Li G, Wang J (2020b) "Unblind your apps: Predicting natural-language labels for mobile gui components by deep learning," in Int Conf on Software Engineering, IEEE
Chen J, Xia X, Lo D, Grundy J, Yang X (2021) "Maintenance-related concerns for post-deployed Ethereum smart contract development: issues, techniques, and future challenges," in Empirical Software Engineering, Springer
Chen S, Fan L, Meng G, Su T, Xue M, Xue Y, Liu Y, Xu L (2020a) "An empirical assessment of security risks of global android banking apps," in Int Conf on Software Engineering, IEEE
Claes M, Mens T, Di Cosmo R, Vouillon J (2015) "A historical analysis of Debian package incompatibilities," in Int Conf on Mining Software Repositories, IEEE
Cohen J (1960) "A coefficient of agreement for nominal scales," in Educational and psychological measurement, Sage
Coxon APM et al. (1999) Sorting data: collection and analysis. Sage
Dabrowski J, Letier E, Perini A, Susi A (2022) "Analysing app reviews for software engineering: a systematic literature review," in Empirical Software Engineering, Springer
Dixon C, Mahajan R, Agarwal S, Brush A, Lee B, Saroiu S, Bahl V (2010) "The home needs an operating system (and an app store)," in SIGCOMM Workshop on Hot Topics in Networks, ACM
Docker (2022) "Explore Docker's Container Image Repository | Docker Hub." https://hub.docker.com/search?q=. Accessed: Jun. 22, 2022
Dong Z, Böhme M, Cojocaru L, Roychoudhury A (2020) "Time-travel testing of android apps," in Int Conf on Software Engineering, IEEE
E. Commission (2022) "Digital Markets Act: Commission welcomes political agreement on rules to ensure fair and open digital markets." https://ec.europa.eu/commission/presscorner/detail/en/IP_22_1978. Accessed: Jul. 13, 2022
E. International (2022) "TC39 - Specifying JavaScript.." https://tc39.es/. Accessed: Oct. 02, 2022
F-Droid (2022) "F-Droid - Free and Open Source Android App Repository." https://f-droid.org/. Accessed: Oct. 02, 2022
Ferreira G, Jia L, Sunshine J, Kästner C (2021) "Containing malicious package updates in npm with a lightweight permission system," in Int Conf on Software Engineering (ICSE), IEEE
Finkelstein A, Harman M, Jia Y, Martin W, Sarro F, Zhang Y (2017) "Investigating the relationship between price, rating, and popularity in the Blackberry world app store," Inf Softw Technol
Fischer RA-L, Walczuch R, Guzman E (2021) "Does culture matter? impact of individualism and uncertainty avoidance on app reviews," in Int Conf on Software Engineering: Software Engineering in Society, IEEE
Gaber T, Ahmed A, Mostafa A (2020) "Privdrm: A privacy-preserving secure digital right management system," in Evaluation and Assessment in Software Engineering, ACM
Gholami S, Khazaei H, Bezemer C-P (2021) "Should you upgrade official docker hub images in production environments?," in Int Conf on Software Engineering: New Ideas and Emerging Results (ICSE-NIER), IEEE
GitHub (2022) "GitHub Marketplace Â· to improve your workflow Â· GitHub." https://github.com/marketplace?type=. Accessed: Jun. 06 2022
Google (2022a) "Chrome Web Store - Extensions." https://chrome.google.com/webstore/category/extensions. Accessed: Jun. 22, 2022
Google (2022b) "Chrome Web Store payments deprecation." https://developer.chrome.com/docs/webstore/cws-payments-deprecation/. Accessed: Mar. 16, 2022
Guo H, Singh MP (2020) "Caspar: extracting and synthesizing user stories of problems from app reviews," in Int Conf on Software Engineering, IEEE
Guzman E, Oliveira L, Steiner Y, Wagner LC, Glinz M (2018) "User feedback in the app store: a cross-cultural study," in Int Conf on Software Engineering: Software Engineering in Society
Haering M, Stanik C, Maalej W (2021) "Automatically matching bug reports with related app reviews," in Int Conf on Software Engineering, IEEE
Haggag O, Haggag S, Grundy J, Abdelrazek M (2021) "COVID-19 vs social media apps: does privacy really matter?," in Int Conf on Software Engineering: Software Engineering in Society, IEEE
Harman M, Jia Y, Zhang Y (2012) "App store mining and analysis: MSR for App Stores," in Int. Conf. on Mining Software Repositories, IEEE
He X, Dai W, Cao G, Tang R, Yuan M, Yang Q (2015) "Mining target users for online marketing based on app store data," in Int Conf on Big Data (Big Data), IEEE
Hoda R, Noble J, Marshall S (2012) "Developing a grounded theory to explain the practices of self-organizing Agile teams," in Empirical Software Engineering, Springer

Hu Y, Wang H, Ji T, Xiao X, Luo X, Gao P, Guo Y (2021) "Champ: Characterizing undesired app behaviors from user comments based on market policies," in Int Conf on Software Engineering, IEEE
Ibrahim MH, Sayagh M, Hassan A. E (2020) "Too many images on dockerhub! How different are images for the same system?," Empir Softw Eng
Jansen S, Bloemendal E (2013) "Defining app stores: The role of curated marketplaces in software ecosystems," in ICSOB, Springer
Khalid H, Nagappan M, Shihab E, Hassan AE (2014) "Prioritizing the devices to test your app on: A case study of android game apps," in Int Symposium on Foundations of Software Engineering
Khatibi Bardsiri V, Jawawi DNA, Hashim SZM, Khatibi E (2014) "A flexible method to estimate the software development effort based on the classification of projects and localization of comparisons," Empir Softw Eng
Kodi T (2022) "The Movie Database Python | Matrix | Addons | Kodi." https://kodi.tv/addons/matrix/metadata.themoviedb.org.python. Accessed: Jul. 13, 2022
Kuchta T, Lutellier T, Wong E, Tan L, Cadar C (2018) "On the correctness of electronic documents: studying, finding, and localizing inconsistency bugs in PDF readers and files," Empir Softw Eng
Kuznetsov K, Fu C, Gao S, Jansen DN, Zhang L, Zeller A (2021) "Frontmatter: mining Android user interfaces at scale," in Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ACM
Lantz CA, Nebenzahl E (1996) "Behavior and interpretation of the κ statistic: Resolution of the two paradoxes," in J Clin Epidemiol, Elsevier
Lee D, Lin D, Bezemer C-P, Hassan AE (2020b) "Building the perfect game–an empirical study of game modifications," Empir Softw Eng
Lee D, Rajbahadur GK, Lin D, Sayagh M, Bezemer C-P, Hassan AE (2020a) "An empirical study of the characteristics of popular Minecraft mods," Empir Softw Eng
Lemon M (2018) "Two Point Hospital no longer uses Denuvo DRM." https://www.vg247.com/two-point-hospital-no-longer-uses-denuvo-drm. Accessed: Mar. 31, 2023
Lin D, Bezemer C-P, Zou Y, Hassan AE (2019) "An empirical study of game reviews on the steam platform," in Empirical Software Engineering, Springer
Liu P, Li L, Yan Y, Fazzini M, Grundy J (2021) "Identifying and characterizing silently-evolved methods in the android API," in Int Conf on Software Engineering: Software Engineering in Practice, IEEE
Lu Z, Shi Y, Tao R, Zhang Z (2019) "Blockchain for digital rights management of design works," in Int. Conf on Software Engineering and Service Science (ICSESS), IEEE
Ma S, Li J, Kim H, Bertino E, Nepal S, Ostry D, Sun C (2021) "Fine with 1234? An Analysis of SMS One-Time Password Randomness in Android Apps," in Int Conf on Software Engineering, IEEE
Maalej W, Nayebi M, Ruhe G (2019) "Data-driven requirements engineering-an update," in Int Conf on Software Engineering: Software Engineering in Practice (ICSE-SEIP), IEEE
MacQueen J et al. (1967) "Some methods for classification and analysis of multivariate observations," in Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, Oakland, CA, USA
Manotas I, Bird C, Zhang R, Shepherd D, Jaspan C, Sadowski C, Pollock L, Clause J (2016) "An empirical study of practitioners' perspectives on green software engineering," in Int Conf on Software Engineering
Martin W, Sarro F, Harman M (2016b) "Causal impact analysis for app releases in google play," in Int. Symposium on Foundations of software engineering
Martin W, Sarro F, Jia Y, Zhang Y, Harman M (2016a) "A survey of app store analysis for software engineering," in Transactions on software engineering, IEEE
Masood Z, Hoda R, Blincoe K (2020) "How agile teams make self-assignment work: a grounded theory study," in Empirical Software Engineering, Springer
McIntosh S, Kamei Y, Nagappan M (2019) Release engineering for Mobile applications — communications of NII Shonan meetings. Springer
McMillan C, Grechanik M, Poshyvanyk D (2012) "Detecting similar software applications," in Int Conf on Software Engineering (ICSE), IEEE
Murali V, Yao E, Mathur U, Chandra S (2021) "Scalable statistical root cause analysis on app telemetry," in Int Conf on Software Engineering: Software Engineering in Practice, IEEE
Murphy-Hill E, Zimmermann T, Nagappan N (2014) "Cowboys, ankle sprains, and keepers of quality: How is video game development different from software development?," in Int Conf on Software Engineering
Nayebi M, Adams B, Ruhe G (2016) "Release Practices for Mobile Apps–What do Users and Developers Think?," in Int Conf On software analysis, evolution, and reengineering (saner), IEEE
Nayebi M, Cho H, Farrahi H, Ruhe G (2017b) "App store mining is not enough," in Int Conf on Software Engineering Companion (ICSE-C), IEEE

Nayebi M, Farahi H, Ruhe G (2017a) "Which version should be released to app store?," in Int Symposium on Empirical Software Engineering and Measurement (ESEM), IEEE
Ng YY, Zhou H, Ji Z, Luo H, Dong Y (2014) "Which Android app store can be trusted in China?," in Computer Software and Applications Conference, IEEE
Nguyen T, Vu P, Nguyen T (2020) "Code recommendation for exception handling," in Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ACM
npm, "npm About." (2022) https://www.npmjs.com/about. Accessed: Oct. 02, 2022
Obie HO, Hussain W, Xia X, Grundy J, Li L, Turhan B, Whittle J, Shahin M (2021) "A first look at human values-violation in app reviews," in Int Conf on Software Engineering: Software Engineering in Society, IEEE
Pan L, Cui B, Liu H, Yan J, Wang S, Yan J, Zhang J (2020) "Static asynchronous component misuse detection for Android applications," in Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ACM
Pascarella L, Palomba F, Di Penta M, Bacchelli A (2018) "How is video game development different from software development in open source?," in Int Conf on Mining Software Repositories, IEEE
Pérez J, Daz J, Garcia-Martin J, Tabuenca B (2020) "Systematic literature reviews in software engineeringâ€"Enhancement of the study selection process using Cohens kappa statistic," in J Syst Softw, Elsevier
Pham VVH, Liu X, Zheng X, Fu M, Deshpande SV, Xia W, Zhou R, Abdelrazek M (2017) "PaaS-black or white: an investigation into software development model for building retail industry SaaS," in Int. Conf. On software engineering companion (ICSE-C), IEEE
Pickerill P, Jungen HJ, Ochodek M, Maćkowiak M, Staron M (2020) "Phantom: curating github for engineered software projects using time-series clustering," Empir Softw Eng
Prévost R, McQuaid M, Lalonde D (2022) "The Missing Package Manager for macOS (or Linux) â€" Homebrew." https://brew.sh/. Accessed: Jun. 22, 2022
Rahaman S, Neamtiu I, Yin X (2021) "Algebraic-datatype taint tracking, with applications to understanding Android identifier leaks," in Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ACM
Rosen C, Shihab E (2016) "What are mobile developers asking about? a large scale study using stack overflow," in Empirical Software Engineering, Springer
Rousseeuw PJ (1987) "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," in J Comput Appl Math, Elsevier
Ruiz IJM, Nagappan M, Adams B, Hassan AE (2012) "Understanding reuse in the android market," in Int. Conf. on Program Comprehension, IEEE
Sarro F, Al-Subaihin AA, Harman M, Jia Y, Martin W, Zhang Y (2015) "Feature lifecycles as they spread, migrate, remain, and die in app stores," in Int requirements engineering conference (RE), IEEE
Scoccia GL, Autili M, Stilo G, Inverardi P (2022) "An empirical study of privacy labels on the apple iOS mobile app store," in Int Conf on Mobile Software Engineering and Systems
Shams RA, Hussain W, Oliver G, Nurwidyantoro A, Perera H, Whittle J (2020) "Society-oriented applications development: Investigating users values from bangladeshi agriculture mobile applications," in Int. Conf. on Software Engineering: Software Engineering in Society, IEEE
Shen S, Lu X, Hu Z, Liu X (2017) "Towards release strategy optimization for apps in Google play," in Proceedings of the 9th Asia-Pacific symposium on Internetware
Song W, Han M, Huang J (2021) "IMGDroid: Detecting Image Loading Defects in Android Applications," in Int Conf on Software Engineering, IEEE
Subramanian GH, Pendharkar PC, Wallace M (2006) "An empirical study of the effect of complexity, platform, and program type on software development effort of business applications," Empir Softw Eng
Sun R, Wang W, Xue M, Tyson G, Camtepe S, Ranasinghe DC (2021) "An empirical assessment of global COVID-19 contact tracing applications," in Int Conf on Software Engineering, IEEE
Sung A, Kim S, Kim Y, Jang Y, Kim J (2019) "Test automation and its limitations: a case study," in Int Conf On automated software engineering (ASE), IEEE
Tang C, Chen S, Fan L, Xu L, Liu Y, Tang Z, Dou L (2019) "A large-scale empirical study on industrial fake apps," in Int Conf on Software Engineering: Software Engineering in Practice (ICSE-SEIP), IEEE
Truelove A, de Almeida ES, Ahmed I (2021) "Well Fix It in Post: What Do Bug Fixes in Video Game Update Notes Tell Us?," in Int Conf on Software Engineering, IEEE
Ullmann GC, Politowski C, Guéhéneuc Y-G, Petrillo F (2022) "What makes a game high-rated? towards factors of video game success," in Int ICSE Workshop on Games and Software Engineering: Engineering Fun, Inspiration, and Motivation
Valve (2022) "Welcome to Steam." https://store.steampowered.com/, Accessed: Jun. 22 2022

Van Der Linden D, Anthonysamy P, Nuseibeh B, Tun TT, Petre M, Levine M, Towse J, Rashid A (2020) "Schrödinger's security: Opening the box on app developers' security rationale," in Int Conf on Software Engineering, IEEE
Vassallo C, Panichella S, Palomba F, Proksch S, Gall HC, Zaidman A (2020) "How developers engage with static analysis tools in different contexts," in Empirical Software Engineering, Springer
Vouillon J, Cosmo RD (2013) "On software component co-installability," Trans Softw Eng Methodol (TOSEM)
Walker D, Myrick F (2006) "Grounded theory: An exploration of process and procedure," in Qualitative health research, Sage
Wang H, Liu Z, Liang J, Vallina-Rodriguez N, Guo Y, Li L, Tapiador J, Cao J, Xu G (2018) "Beyond google play: a large-scale comparative study of chinese android app markets," in Internet measurement conference 2018
Wang H, Wang X, Guo Y (2019) "Characterizing the global mobile app developers: a large-scale empirical study," in Int Conf on Mobile Software Engineering and Systems, IEEE
Wang P, Brown C, Jennings JA, Stolee KT (2022) "Demystifying regular expression bugs," in Empirical Software Engineering, Springer
Wikipedia (2022) "Electronic AppWrapper - Wikipedia." https://en.wikipedia.org/wiki/Electronic_AppWrapper. Accessed: Jun. 22, 2022
Wold S, Esbensen K, Geladi P (1987) "Principal component analysis," in Chemometrics and intelligent laboratory systems, Elsevier
WordPress (2022) "WordPress Plugins | WordPress.org." https://wordpress.org/plugins/. Accessed: Jun. 22, 2022
Wu H, Deng W, Niu X, Nie C (2021) "Identifying key features from app user reviews," in Int Conf on Software Engineering, IEEE
Yang B, Xing Z, Xia X, Chen C, Ye D, Li S (2021) "Don't do that! hunting down visual design smells in complex uis against design guidelines," in Int Conf on Software Engineering, IEEE
Yang S, Wang Y, Yao Y, Wang H, Ye YF, Xiao X (2022) DescribeCtx: context-aware description synthesis for sensitive behaviors in mobile apps. In: Int Conf on Software Engineering, IEEE
Ye J, Chen K, Xie X, Ma L, Huang R, Chen Y, Xue Y, Zhao J (2021) "An empirical study of GUI widget detection for industrial mobile games," in Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ACM
Yu S, Fang C, Cao Z, Wang X, Li T, Chen Z (2021b) "Prioritize crowdsourced test reports via deep screenshot understanding," in Int Conf on Software Engineering, IEEE
Yu S, Fang C, Yun Y, Feng Y (2021a) "Layout and image recognition driving cross-platform automated mobile testing," in Int Conf on Software Engineering, IEEE
Zhan X, Fan L, Chen S, Wu F, Liu T, Luo X, Liu Y (2021) "Atvhunter: Reliable version detection of third-party libraries for vulnerability identification in android applications," in Int. Conf. on Software Engineering, IEEE
Zhang X, Wang X, Slavin R, Breaux T, Niu J (2020) "How does misconfiguration of analytic services compromise mobile privacy?," in Int Conf on Software Engineering, IEEE
Zhang Z, Feng Y, Ernst MD, Porst S, Dillig I (2021) "Checking conformance of applications against GUI policies," in Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ACM
Zhao T, Chen C, Liu Y, Zhu X (2021) "GUIGAN: Learning to Generate GUI Designs Using Generative Adversarial Networks," in Int Conf on Software Engineering, IEEE

Authors and Affiliations

Wenhan Zhu[1] · Sebastian Proksch[2] · Daniel M. German[3] · Michael W. Godfrey[1] · Li Li[4] · Shane McIntosh[1]

Sebastian Proksch
s.proksch@tudelft.nl

Daniel M. German
dmg@uvic.ca

Michael W. Godfrey
migod@uwaterloo.ca

Li Li
lilicoding@ieee.org

Shane McIntosh
shane.mcintosh@uwaterloo.ca

[1] David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada

[2] Delft University of Technology, Delft, Netherlands

[3] Department of Computer Science, University of Victoria, Victoria, Canada

[4] School of Software, Beihang University, Beijing, China