A Web-Based Tool for Using Storyboard of Android Apps

Yuxin Zhang*
College of Intelligence and Computing
Tianjin University
Tianjin, China

Sen Chen*‡
College of Intelligence and Computing
Tianjin University
Tianjin, China

Lingling Fan†
College of Cyber Science
Nankai University
Tianjin, China

***Abstract*—The development team usually makes painstaking efforts to review and analyze many existing apps with similar purposes such as competitive analysis, design recommendation, code generation, and app testing. To assist different roles in doing these tasks, in our prior work, two advanced approaches (i.e., StoryDroid and StoryDistiller) have been proposed to automatically generate the storyboards for Android apps with rich features such as UI pages, UI components, layout code, and logic code. These approaches both aim at exploring and parsing as many app pages as possible but lack some consideration of the presentation and interpretability of the results for different users such as PMs, designers, and developers. To improve usability and scalability, this paper presents a web-based offline tool, named StoryDroid+, which provides an operation-friendly platform for using storyboards and helps different stakeholders (e.g., designers, package managers, developers) explore and understand apps from different perspectives through rich visual pages. The tool and datasets are available at: https://github.com/tjusenchen/StoryDroid and the demonstration video can be found at: https://youtu.be/prszxRdkdYU.**

***Index Terms*—Storyboard, App exploration, GUI exploration, App review, Android app**

I. Introduction

Nowadays, mobile applications (apps) are everywhere. In just a few years, smartphones have become an indispensable part of our lives, helping us perform daily tasks, e.g., reading, shopping, banking, and chatting [1]. This has led to the unprecedented growth of the app development business. Today, companies, governments, and various organizations are working hard to develop new and practical apps, hoping to stay on users' mobile devices. Competition among apps in the same category is also growing. More than 3.8 million Android apps and 2 million iOS apps are trying to attract users in the two major mobile app markets, i.e., Google Play and Apple App Store. However, there are still functional bugs [2]–[4], security vulnerabilities [5]–[7], and a lack of marketing competitiveness in mass mobile applications [8]. Therefore, app developers and companies tend to conduct an extensive competitive analysis of existing apps with similar functionality through app reviews. This analysis helps to understand the strengths and weaknesses of competitors and reduce market risks before development, and it inspires developers with innovative ideas for app design and implementation.

However, to achieve the above goals, developers or companies need to download apps from the market and install them on mobile devices for manual exploration and recording, which may be ineffective. To this end, Chen et al. [8] proposed a hybrid method called StoryDistiller, which combines static and dynamic methods to extract the relatively complete activity transition graph (ATG) of apps more effectively. It shows a strong ability to extract app storyboards with rich and useful features for different users. However, the lack of visual management also increases the difficulty for users to understand the storyboards. Therefore, to make full use of the value of the information extracted by these tools, we have built a web-based offline tool named StoryDroid+. StoryDroid+ can automatically extract the storyboards of the app and then visualize the pages of the results. In detail, as shown in Fig. 2, StoryDroid+ can visualize the page of the results, including the basic storyboard features such as ATG with UI pages and the corresponding code for developers, app competitive analysis for product managers (PMs) through app comparison, and several useful searching functionalities for designers including searching for similar UI pages and components.

In summary, we highlight the main functionalities of StoryDroid+ as follows.

- *App exploration module* supports both batch or individual exploration on the app exploration webpage.
- *Storyboard display module* shows the rendered UI pages together with the ATGs are displayed to show the app storyboard, including the corresponding layout code, method calls, and Java code of each activity.
- *Competitive analysis module* supports the comparison of any two or more app functionalities based on the extracted app storyboards.
- *Attribute searching module* includes two sub-modules, i.e., search for similar UI pages and components, which can inspire designers to design more competitive UIs.
- *App management module* supports the management of the analyzed samples, including the downloading of output files, selection of multiple apps for competitive analysis, and deleting the apps.

‡Corresponding author (senchen@tju.edu.cn)

II. The Web-Based Tool

In this section, we first briefly introduce the back-end techniques used in StoryDroid+, and then detail the web-based functionalities of StoryDroid+.

A. Back-end Techniques in StoryDroid+

To extract the complete ATG and rich features of the app as comprehensively as possible, StoryDroid+ integrates the existing static and dynamic methods with reference to the implementation of StoryDistiller [8]. As shown in Fig. 1, three phases are designed as follows: *(1)* We first decompile the APK as input, configure the *AndroidManifest.xml* file to enable the third-party startup process, and repackage it to generate a new installable APK file. *(2)* Static extraction lays the foundation for the next stage of dynamic UI page rendering, which mainly includes two steps: activity transition graph (ATG) extraction and inter-component communication (ICC) data extraction. Since the activity transition in fragments and inner classes is representative and widely used in real-world apps, we consider them to construct the static ATG by analyzing their associations with components. To successfully start an activity, the data required to render the target UI page should be provided, so it is also crucial to obtain the ICC data. For each activity, we obtain the parameters needed to start it (including primitive attributes and extra parameters). For primitive attributes such as "action" and "category", we obtain them by parsing the corresponding fields in the manifest file or by extracting them in the Java code. As for the extra parameter extraction, we first determine the methods related to the activity life cycle and then analyze them successively based on the relation between the additional parameters in these methods and page rendering. *(3)* Dynamic UI page rendering mainly includes two steps: UI page rendering, which uses the extracted ICC data, and the Android toolkit to launch and start activities dynamically; UI component exploration, which identifies activity transitions and UI pages by exploring all interactive components of each activity, so as to enhance the static ATG. By using the mixed ATG construction method, we can obtain a more complete ATG with the corresponding rendered UI pages, together with other rich features such as layout codes, call graphs, and screenshots of UI components.

B. Web-based Offline Platform

By integrating the methods used by existing static and dynamic tools for storyboard generation, StoryDroid+ finally extracts a relatively complete ATG and much rich feature information related to apps. For example, in the process of dynamic UI page rendering, we can obtain ATG, UI page, activity name, and call graph in turn. For the activity code, we use the reverse engineering tool Jadx [9] to decompile the APK file and extract the corresponding Java code. Besides, we obtain the layout code by storing the current activity layout when rendering the UI pages. In addition, we can also obtain other properties of the UI components from their layout code to enrich the final presentation. However, obtaining such information does not mean that users can fully understand it. If this information cannot be displayed in an appropriate form, it would greatly weaken its value and make it difficult for users to understand. To this end, we set up an offline web platform composed by user-friendly operation and a storyboard to better tell the app. We linked the app information obtained by StoryDroid+ in a visual form to help users explore the app from a deeper level. The implementation of StoryDroid+ on the offline web platform mainly includes six function modules:

① *App exploration module*. StoryDroid+ provides users with an operation-friendly webpage that allows them to upload one or multiple apks for analysis (① App exploration module shown in Fig. 2). After uploading, StoryDroid+ will use the hybrid method described above to explore the uploaded apk(s) and parse various features such as ATG, UI pages, and layout code. When the number of uploaded apk(s) is more than one, the exploration results will be displayed in comparison, which allows users to directly see the differences between apps.

② *Storyboard display module*. To clearly show the results of exploration to users, we will display all the features parsed on the webpage. First, we use the component Network [10] (a visual network composed of nodes and edges) to draw ATG, which is implemented based on the dynamic visualization library *vis.js*. As shown in Fig. 2 (② Storyboard display module for developers), we use the rendered screenshots of UI pages and names of the activities as nodes and use "edge" to describe the transition relations between UI pages. Displaying ATG in the form of a visual network can not only enable users to understand the presentation of real UI pages but also show the relation and navigation between app UI pages more clearly. Users can also get the trigger mechanism between each function module while knowing more about the functions of the app. This is equivalent to the process of disassembling the entire app and reassembling it for users. In addition, we also associate the method call graph, activity code, and layout code of each activity with the activity, so that users such as app developers can better understand the specific implementation of each activity. The "Package Name", "Package Version", "Activity Number", and other relevant attributes about the app are also clearly visible on the same webpage. To facilitate users to quickly find the target activity, StoryDroid+ has a query function on this page ("*Search (activity name)*"). Users can quickly locate and view the activity's method call graph, activity code, layout code, and other useful information by entering keywords.

③ *Competitive analysis module*. In order to facilitate users such as PMs comparing different apps (such as conducting competitive analysis), StoryDroid+ supports users selecting two apps they are interested in for visual display (the search function can also be used to retrieve similar apps). As shown in Fig. 2 (③ Competitive analysis module for PMs), users can visually see the storyboards of the two apps, and all information related to the two apps can be found directly on the visualization page. Users can also use the search bar to quickly locate activities with the same name in these two apps. Through these functions, users can conduct competitive analysis on existing apps with similar functionality, understand

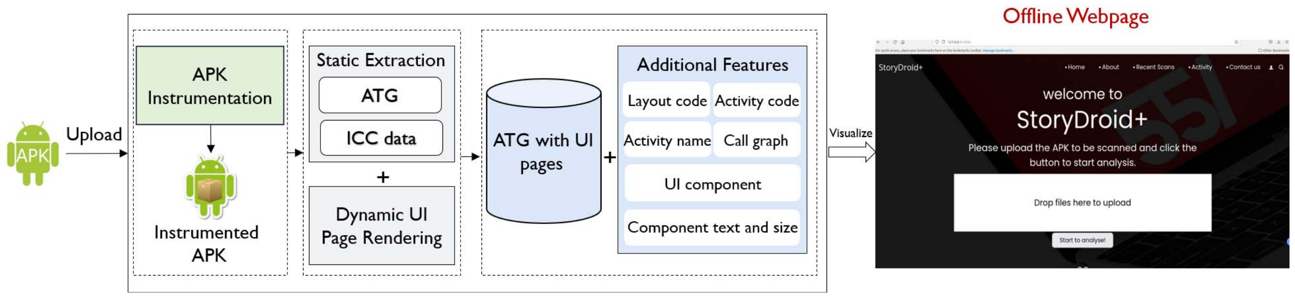

Fig. 1. The back-end techniques used in StoryDroid+.

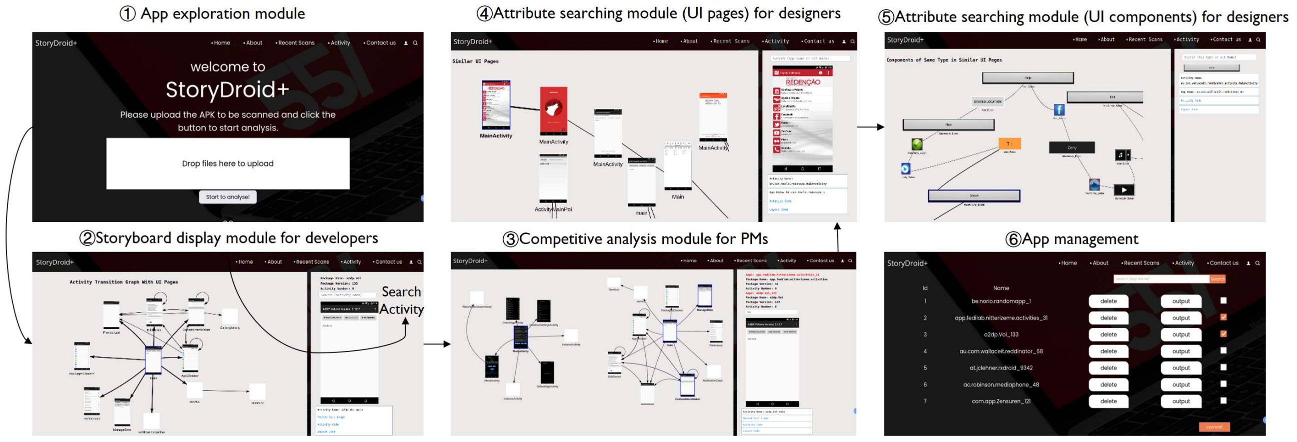
① App exploration module
④Attribute searching module (UI pages) for designers
⑤Attribute searching module (UI components) for designers
②Storyboard display module for developers
③Competitive analysis module for PMs
⑥App management
welcome to
StoryDroid+
Please upload the APK to be scanned and click the button to start analysis.
Drop files here to upload
Start to analyse!
Search Activity
Activity Transition Graph With UI Pages
Similar UI Pages
MainActivity
delete
output

Fig. 2. Main function modules of the web-based offline tool of StoryDroid+.

the strengths and weaknesses of competitors in advance, and reduce market risks before development. This module offers a thorough competitive analysis that PM urgently needs.

④ *Attribute searching module*. To facilitate users, such as designers, in viewing similar activities and UI components, we classify apps analyzed by users according to the semantic names of activities and establish a database of activities with similar functions. As shown in Fig. 2 (④ Attribute searching module (UI pages) for designers), we also visualize this function to facilitate users to query and view the UI, activity code, and layout code of similar activities. This can not only help UI/UX designers get inspiration from similar app designs but also help developers who want to get inspiration from similar apps link the UI screen with the corresponding real code. In addition, in the same way, we split and classify the components on the UI pages according to the component type and the semantic information of their activities. In Fig. 2 (⑤ Attribute searching module (UI components) for designers), users can learn about the design style of a specific type of component in similar activities to get references.

⑥ *App management module*. As shown in Fig. 2 (⑥ App management), to facilitate subsequent viewing and use, StoryDroid+ builds a local database to store the results of apps that have been analyzed and allows users to manage these apps on the "*Recent Scans*" page, including deleting, viewing, etc. In the meantime, we also log the statistical data from over ten dimensions, which can be generated and directly downloaded from the webpage.

III. Evaluation

StoryDroid+ integrated the back-end techniques proposed by StoryDistiller, therefore, the effectiveness of the storyboard extraction has been thoroughly evaluated in our previous work [8]. We randomly selected 75 open-source apps and 75 closed-source apps as test sets and then compare with three existing dynamic and static ATG exploration tools, i.e., IC3 [11], Gator [12], and Stoat [13], to evaluate the effectiveness of StoryDroid+. The number of activity transfer pairs and activity coverage are used to demonstrate the performance of each tool. The back-end extraction technique can obtain a more complete activity transition graph with real UI pages for both open-source and closed-source apps. Specifically, in the aspect of activity transition pairs, StoryDroid+ is better than static method IC3 [11] and Gator [12] (the average value of StoryDroid+ is 23.3, while IC3 is 7.8, and Gator is 10.0). In terms of activity coverage, it is also better than the dynamic method Stoat [13] (77.5% for StoryDroid+ and 36.3% for Stoat). In comparison with the static method, the performance trend of the activity coverage is similar to that of the activity transition pairs, and StoryDroid+ still

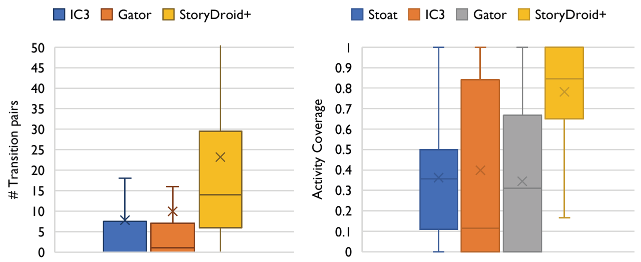

Fig. 3. Comparison of transition pairs and Activity coverage.

outperforms other tools. For the evaluation of the web-based offline platform, we manually uploaded the aforementioned 150 apps and demonstrated StoryDroid+ can work smoothly for different users in practice.

IV. Applications Based on StoryDroid+

StoryDroid+ effectively makes up for the shortcomings of existing ATG construction tools, makes app exploration more comprehensive and complete, and depicts the storyboard of apps more vividly through the operation-friendly visual pages, which can help different stakeholders explore and understand apps from different perspectives.

A. UI/UX Designer

The success of app design depends on the user experience (UX) and user interface (UI). Poorly designed apps would be difficult to be widely accepted and probably lose attraction to potential customers. StoryDroid+ allows designers to explore a large number of apps when designing apps, and learn from the same category of apps or from the activities of the same function of different apps to find inspiration. StoryDroid+ can present a large number of different UI pages to users, which are almost the same as the real UI pages that users will see. In addition, to learning the overall design of UI pages, designers can also learn about the design trend of UI components in a certain type of app through our tools, such as the style, color, and size of buttons.

B. App Developers

StoryDroid+ provides users with ATG with many features (e.g., layout code, activity code, method call graph) and links these features with the corresponding UI screens. Therefore, in the implementation of UI, developers can quickly and accurately obtain the corresponding layout code by searching for similar UI and customize the UI code according to their own purposes to achieve the given UI design. In the implementation of the functions of apps of the same category, in order to have a competitive advantage, apps of the same category will contain more common functions. Therefore, activities with the same semantic name are likely to have similar logic and architecture. Through our tools, developers can refer to activity codes with the same name to help improve the quality of their apps and customize more interesting functions on this basis.

C. App Testers

For app testing, it is necessary to ensure that all important scenarios are covered as much as possible, to avoid service failure caused by untested functions. The relatively complete ATG built by StoryDroid+ can help testers explore more activities and improve test coverage. In the storyboard visualized by StoryDroid+, users can see the navigation relationship between different UI pages and the relatively complete UI page composition of each activity, which can effectively avoid the problem of incomplete testing by testers due to the possible complexity of the app. In addition, StoryDroid+ can help guide the regression testing of apps by identifying the modified ATG and UI components. StoryDroid+ stores the mapping relationship between the UI page and the corresponding layout code and activity code. Testers can compare the differences between different versions of layout code to identify the modified UI components or functions, and update test cases accordingly, rather than designing test cases from scratch, which improves the reusability of test cases and reduces the workload of testers.

V. Related Work

Static GUI exploration (e.g., Gator [12], StoryDroid [1], GoalExplorer [14]) and ICC resolution (e.g., IC3 [11], and ICCBot [15]) are all important way of app abstraction and GUI modeling, however, the completeness of ATGs is limited by the static analysis techniques. They only provided a graph structure of the UI transitions without the rendered UI pages.

A large number of dynamic GUI testing tools (e.g., Monkey [16], A3E [17], Sapienz [18], Stoat [13]) have been exhibited, however, the pure dynamic testing approach is limited by the low activity coverage, which significantly limits the completeness of ATGs.

Compared with these related works, StoryDroid+ optimizes the original tool on ATG construction and UI page rendering by combining the original static method and novel dynamic exploration. Last but not least, StoryDroid+ further provides an operation-friendly web platform for using storyboards and help different stakeholders.

VI. Conclusion

In this paper, we proposed a web-based offline tool named StoryDroid+. By integrating the existing static and dynamic ATG exploration methods, StoryDroid+ can obtain a relatively complete storyboard of apps with rich features. On this basis, StoryDroid+ is designed in a user-friendly way, which can not only make it easier for users to operate but also visualize the storyboard of the app and help different stakeholders explore and understand the app. Compared with the existing tools, StoryDroid+ has better performance and significantly improved efficiency for app review and app understanding.

Acknowledgements

This work was partially supported by the National Natural Science Foundation of China (No. 62102197, 62102284).

References

[1] S. Chen, L. Fan, C. Chen, T. Su, W. Li, Y. Liu, and L. Xu, "Storydroid: Automated generation of storyboard for Android apps," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 596–607.

[2] L. Fan, T. Su, S. Chen, G. Meng, Y. Liu, L. Xu, and G. Pu, "Efficiently manifesting asynchronous programming errors in Android apps," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018, pp. 486–497.

[3] L. Fan, T. Su, S. Chen, G. Meng, Y. Liu, L. Xu, G. Pu, and Z. Su, "Large-scale analysis of framework-specific exceptions in Android apps," in *Proceedings of the 40th International Conference on Software Engineering*, 2018, pp. 408–419.

[4] S. Chen, C. Chen, L. Fan, M. Fan, X. Zhan, and Y. Liu, "Accessible or not? an empirical investigation of Android app accessibility," *IEEE Transactions on Software Engineering*, vol. 48, no. 10, pp. 3954–3968, 2021.

[5] S. Chen, L. Fan, G. Meng, T. Su, M. Xue, Y. Xue, Y. Liu, and L. Xu, "An empirical assessment of security risks of global Android banking apps," in *Proceedings of the 42nd International Conference on Software Engineering*. IEEE Press, 2020, pp. 596–607.

[6] S. Chen, Y. Zhang, L. Fan, J. Li, and Y. Liu, "Ausera: Automated security vulnerability detection for Android apps," in *37th IEEE/ACM International Conference on Automated Software Engineering*, 2022, pp. 1–5.

[7] S. Chen, T. Su, L. Fan, G. Meng, M. Xue, Y. Liu, and L. Xu, "Are mobile banking apps secure? what can be improved?" in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2018, pp. 797–802.

[8] S. Chen, L. Fan, C. Chen, and Y. Liu, "Automatically distilling storyboard with rich features for Android apps," *IEEE Transactions on Software Engineering*, 2022.

[9] (2018) Dex to Java decompiler. [Online]. Available: https://github.com/skylot/jadx

[10] (2022) vis.js Network. [Online]. Available: https://visjs.github.io/vis-network/docs/network/

[11] D. Octeau, D. Luchaup, M. Dering, S. Jha, and P. McDaniel, "Composite constant propagation: Application to Android inter-component communication analysis," in *Proceedings of the 37th International Conference on Software Engineering-Volume 1*. IEEE Press, 2015, pp. 77–88.

[12] S. Yang, H. Wu, H. Zhang, Y. Wang, C. Swaminathan, D. Yan, and A. Rountev, "Static window transition graphs for Android," *Automated Software Engineering*, vol. 25, no. 4, pp. 833–873, 2018.

[13] T. Su, G. Meng, Y. Chen, K. Wu, W. Yang, Y. Yao, G. Pu, Y. Liu, and Z. Su, "Guided, stochastic model-based GUI testing of Android apps," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. ACM, 2017.

[14] D. Lai and J. Rubin, "Goal-driven exploration for Android applications," in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2019, pp. 115–127.

[15] J. Yan, S. Zhang, Y. Liu, J. Yan, and J. Zhang, "Iccbot: fragment-aware and context-sensitive icc resolution for Android applications," in *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings*, 2022, pp. 105–109.

[16] (2018) Google Monkey for Testing. [Online]. Available: https://developer.android.com/studio/test/monkey

[17] T. Azim and I. Neamtiu, "Targeted and depth-first exploration for systematic testing of Android apps," in *Acm Sigplan Notices*, vol. 48, no. 10. ACM, 2013, pp. 641–660.

[18] K. Mao, M. Harman, and Y. Jia, "Sapienz: Multi-objective automated testing for Android applications," in *Proceedings of the 25th international symposium on software testing and analysis*, 2016, pp. 94–105.