On the feasibility and challenges of generating executable ESPRESSO tests

Iván Arcuschin, Christian Ciccaroni, Juan P. Galeotti, and José Miguel Rojas

TABLE I: Tools for automatic ANDROID testing, categorized by type of output.

Output type	Tools & Works	Count
Custom format	SAPIENZ [1], CRASHSCOPE [2], MAMBA [3], SSDA [4], GAT [5], ATT [6], VALERA [7], Ermuth et al. [8], IntentDroid [9], iMPAcT [10], CRAXDroid [11], APSET [12], PBGT [13], Sasnauskas et al. [14], Appstrument [15], Smartdroid [16], AimDroid [17], Polariz [18], ACTEVE [19], JPF-ANDROID [20], SWIFTHAND [21], DRUN [22], MonkeyLab [23], CuriousDroid [24], Mirzaei et al. [25]	25
Robotium	EVODROID [26], GUIRIPPER [27], A^3E [28], ORBIT [29], ALARic [30], QUANTUM [31], TrimDroid [32], MoTiF [33], Zhang et al. [34], Farto et al. [35], Guo et al. [36], Yang et al. [37], LEAKDROID [38], Mahmood et al. [39], LAND [40], ATG [41], AMOGA [42], SIG-Droid [43], AndroidRipper [44], AGRippin [45], ACRT [46], A^2T^2 [47]	22
Not executable	MONKEY [48], APE [49], STOAT [50], DYNODROID [51], Humanoid [52], MATE [53], Xdroid [54], MobiCoMonkey [55], SynthesiSE [56], CrawlDroid [57], AppSPIN [58], Kraken [59], ParaAim [60], IntentFuzzer [61], Pretect [62], DiagDroid [63], AppAudit [64], DroidBot [65], Data Loss Detector [66], DroidFuzzer [67], TimeMachine [68], VANARSena [69]	22
UI Automator	PUMA [70], WCTester [71], DroidMate [72] Sentinel [73], AppCheck [74], Mimic [75], Q-testing [76], Mobolic [77], T+ [78], Fusion [79],	10
JUnit	MobiGUITAR [80], Shahriar et al. [81], Avancini et al. [82], Franke et al. [83], CTGen [84], Hu et al. [85]	6
Espresso	Rohella et al. [86], COBWEB [87], RacerDroid [88], ETR [89], Barista [90], Coppola et al. [91]	6
Appium	Autodroid [92], Ali et al. [93], ACAT [94], JazzDroid [95]	4
MonkeyRunner	EHBDroid [96], Banerjee et al. [97], AppDoctor [98]	3
Robolectric	CATE [99], RoboLIFT [100]	2
Calabash	Griebe et al. [101]	1

REFERENCES

[1] K. Mao, M. Harman, and Y. Jia, "Sapienz: multi-objective automated testing for android applications," in *ISSTA*. ACM, 2016.

[2] K. Moran, M. L. Vásquez, C. Bernal-Cárdenas, C. Vendome, and D. Poshyvanyk, "Automatically discovering, reporting and reproducing android application crashes," in *ICST*. IEEE Computer Society, 2016, pp. 33–44.

[3] J. C. J. Keng, L. Jiang, T. K. Wee, and R. K. Balan, "Graph-aided directed testing of Android applications for checking runtime privacy behaviours," in *AST@ICSE*. ACM, 2016, pp. 57–63.

[4] Y. Hu and I. Neamtiu, "Fuzzy and cross-app replay for smartphone apps," in *AST@ICSE*. ACM, 2016, pp. 50–56.

[5] X. Wu, Y. Jiang, C. Xu, C. Cao, X. Ma, and J. Lu, "Testing android apps via guided gesture event generation," in *APSEC*. IEEE Computer Society, 2016, pp. 201–208.

[6] Z. Meng, Y. Jiang, and C. Xu, "Facilitating reusable and scalable automated testing and analysis for android apps," in *Internetware*. ACM, 2015, pp. 166–175.

[7] Y. Hu, T. Azim, and I. Neamtiu, "Versatile yet lightweight record-and-replay for android," in *OOPSLA*. ACM, 2015, pp. 349–366.

[8] M. Ermuth and M. Pradel, "Monkey see, monkey do: effective generation of GUI tests with inferred macro events," in *ISSTA*. ACM, 2016, pp. 82–93.

[9] R. Hay, O. Tripp, and M. Pistoia, "Dynamic detection of inter-application communication vulnerabilities in android," in *ISSTA*. ACM, 2015, pp. 118–128.

[10] I. C. Morgado and A. C. R. Paiva, "Testing approach for mobile applications through reverse engineering of UI patterns," in *ASE Workshops*. IEEE Computer Society, 2015, pp. 42–49.

[11] C. Yeh, H. Lu, C. Chen, K. K. Khor, and S. Huang, "Craxdroid: Automatic android system testing by selective symbolic execution," in *SERE (Companion)*. IEEE, 2014, pp. 140–148.

[12] S. Salva and S. R. Zafimiharisoa, "Apset, an android application security testing tool for detecting intent-based vulnerabilities," *Int. J. Softw. Tools Technol. Transf.*, vol. 17, no. 2, pp. 201–221, 2015.

[13] R. M. L. M. Moreira, A. C. R. Paiva, and A. Memon, "A pattern-based approach for GUI modeling and testing," in *ISSRE*. IEEE Computer Society, 2013, pp. 288–297.

[14] R. Sasnauskas and J. Regehr, "Intent fuzzer: crafting intents of death," in *WODA+PERTEA@ISSTA*. ACM, 2014, pp. 1–5.

[15] V. Nandakumar, V. Ekambaram, and V. Sharma, "Appstrument - A unified app instrumentation and automated playback framework for testing mobile applications," in *MobiQuitous*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 131. Springer, 2013, pp. 474–486.

[16] C. Zheng, S. Zhu, S. Dai, G. Gu, X. Gong, X. Han, and W. Zou, "Smartdroid: an automatic system for revealing ui-based trigger conditions in android applications," in *SPSM@CCS*. ACM, 2012, pp. 93–104.

[17] T. Gu, C. Cao, T. Liu, C. Sun, J. Deng, X. Ma, and J. Lü, "Aimdroid: Activity-insulated multi-level automated testing for android applications," in *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2017, pp. 103–114.

[18] K. Mao, M. Harman, and Y. Jia, "Crowd intelligence enhances automated mobile testing," in *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2017, pp. 16–26.

[19] S. Anand, M. Naik, M. J. Harrold, and H. Yang, "Automated concolic testing of smartphone apps," in *SIGSOFT FSE*. ACM, 2012, p. 59.

[20] H. van der Merwe, B. van der Merwe, and W. Visser, "Verifying android applications using java pathfinder," *ACM SIGSOFT Software Engineering Notes*, vol. 37, no. 6, pp. 1–5, 2012.

[21] W. Choi, G. C. Necula, and K. Sen, "Guided GUI testing of android apps with minimal restart and approximate learning," in *OOPSLA*. ACM, 2013, pp. 623–640.

[22] Q. Sun, L. Xu, L. Chen, and W. Zhang, "Replaying harmful data races in android apps," in *ISSRE Workshops*. IEEE Computer Society, 2016, pp. 160–166.

[23] M. L. Vásquez, M. White, C. Bernal-Cárdenas, K. Moran, and D. Poshyvanyk, "Mining android app usages for generating actionable gui-based execution scenarios," in *MSR*. IEEE Computer Society, 2015, pp. 111–122.

[24] P. Carter, C. Mulliner, M. Lindorfer, W. K. Robertson, and E. Kirda, "Curiousdroid: Automated user interface interaction for android application analysis sandboxes," in *Financial Cryptography*, ser. Lecture Notes in Computer Science, vol. 9603. Springer, 2016, pp. 231–249.

[25] N. Mirzaei, S. Malek, C. S. Pasareanu, N. Esfahani, and R. Mahmood, "Testing android apps through symbolic execution," *ACM SIGSOFT Software Engineering Notes*, vol. 37, no. 6, pp. 1–5, 2012.

[26] R. Mahmood, N. Mirzaei, and S. Malek, "Evodroid: segmented evolutionary testing of android apps," in *SIGSOFT FSE*. ACM, 2014, pp. 599–609.

[27] D. Amalfitano, A. R. Fasolino, P. Tramontana, S. D. Carmine, and A. M. Memon, "Using GUI ripping for automated testing of android applications," in *ASE*. ACM, 2012, pp. 258–261.

[28] T. Azim and I. Neamtiu, "Targeted and depth-first exploration for systematic testing of android apps," in *OOPSLA*. ACM, 2013, pp. 641–660.

[29] W. Yang, M. R. Prasad, and T. Xie, "A grey-box approach for automated gui-model generation of mobile applications," in *FASE*, ser. Lecture Notes in Computer Science, vol. 7793. Springer, 2013, pp. 250–265.

[30] V. Riccio, D. Amalfitano, and A. R. Fasolino, "Is this the lifecycle we really want?: an automated black-box testing approach for android activities," in *ISSTA/ECOOP Workshops*. ACM, 2018, pp. 68–77.

[31] R. N. Zaeem, M. R. Prasad, and S. Khurshid, "Automated generation of oracles for testing user-interaction features of mobile apps," in *ICST*. IEEE Computer Society, 2014, pp. 183–192.

[32] N. Mirzaei, J. Garcia, H. Bagheri, A. Sadeghi, and S. Malek, "Reducing combinatorics in GUI testing of android applications," in *ICSE*. ACM, 2016, pp. 559–570.

[33] M. Gómez, R. Rouvoy, B. Adams, and L. Seinturier, "Reproducing context-sensitive crashes of mobile apps using crowdsourced monitoring," in *MOBILESoft*. ACM, 2016, pp. 88–99.

[34] H. Zhang, H. Wu, and A. Rountev, "Automated test generation for detection of leaks in android applications," in *AST@ICSE*. ACM, 2016, pp. 64–70.

[35] G. de Cleva Farto and A. T. Endo, "Evaluating the model-based testing approach in the context of mobile applications," in *CLEI Selected Papers*, ser. Electronic Notes in Theoretical Computer Science, vol. 314. Elsevier, 2014, pp. 3–21.

[36] C. Guo, J. Xu, H. Yang, Y. Zeng, and S. Xing, "An automated testing approach for inter-application security in android," in *AST*. ACM, 2014, pp. 8–14.

[37] S. Yang, D. Yan, and A. Rountev, "Testing for poor responsiveness in android applications," in *2013 1st International Workshop on the Engineering of Mobile-Enabled Systems (MOBS)*, 2013, pp. 1–6.

[38] D. Yan, S. Yang, and A. Rountev, "Systematic testing for resource leaks in android applications," in *ISSRE*. IEEE Computer Society, 2013, pp. 411–420.

[39] R. Mahmood, N. Esfahani, T. Kacem, N. Mirzaei, S. Malek, and A. Stavrou, "A whitebox approach for automated security testing of android applications on the cloud," in *AST*. IEEE Computer Society, 2012, pp. 22–28.

[40] J. Yan, T. Wu, J. Yan, and J. Zhang, "Widget-sensitive and back-stack-aware GUI exploration for testing android apps," in *QRS*. IEEE, 2017, pp. 42–53.

[41] F. Paulovsky, E. Pavese, and D. Garbervetsky, "High-coverage testing of navigation models in android applications," in *2017 IEEE/ACM 12th International Workshop on Automation of Software Testing (AST)*, 2017, pp. 52–58.

[42] I. Salihu, R. Ibrahim, B. S. Ahmed, K. Z. Zamli, and A. Usman, "Amoga: A static-dynamic model generation strategy for mobile apps testing," *IEEE Access*, vol. 7, pp. 17 158–17 173, 2019.

[43] N. Mirzaei, H. Bagheri, R. Mahmood, and S. Malek, "Sig-droid: Automated system input generation for android applications," in *ISSRE*. IEEE Computer Society, 2015, pp. 461–471.

[44] D. Amalfitano, A. R. Fasolino, P. Tramontana, S. D. Carmine, and A. M. Memon, "Using GUI ripping for automated testing of android applications," in *ASE*. ACM, 2012, pp. 258–261.

[45] D. Amalfitano, N. Amatucci, A. R. Fasolino, and P. Tramontana, "Agrippin: a novel search based testing technique for android applications," in *DeMobile@SIGSOFT FSE*. ACM, 2015, pp. 5–12.

[46] C. H. Liu, C. Y. Lu, S. J. Cheng, K. Y. Chang, Y. C. Hsiao, and W. M. Chu, "Capture-replay testing for android applications," in *2014*

International Symposium on Computer, Consumer and Control, 2014, pp. 1129–1132.

[47] D. Amalfitano, A. R. Fasolino, and P. Tramontana, "A GUI crawling-based technique for android mobile application testing," in *ICST Workshops*. IEEE Computer Society, 2011, pp. 252–261.

[48] "UI/Application Exerciser Monkey," https://developer.android.com/studio/test/monkey.html, 2019.

[49] T. Gu, C. Sun, X. Ma, C. Cao, C. Xu, Y. Yao, Q. Zhang, J. Lu, and Z. Su, "Practical GUI testing of android applications via model abstraction and refinement," in *ICSE*. IEEE / ACM, 2019, pp. 269–280.

[50] 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 *ESEC/SIGSOFT FSE*. ACM, 2017.

[51] A. Machiry, R. Tahiliani, and M. Naik, "Dynodroid: an input generation system for android apps," in *ESEC/SIGSOFT FSE*. ACM, 2013, pp. 224–234.

[52] Y. Li, Z. Yang, Y. Guo, and X. Chen, "Humanoid: A deep learning-based approach to automated black-box android app testing," in *ASE*. IEEE, 2019, pp. 1070–1073.

[53] M. M. Eler, J. M. Rojas, Y. Ge, and G. Fraser, "Automated accessibility testing of mobile apps," in *ICST*. IEEE Computer Society, 2018.

[54] C. Cao, C. Meng, H. Ge, P. Yu, and X. Ma, "Xdroid: Testing android apps with dependency injection," in *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1, 2017, pp. 214–223.

[55] A. S. Ami, M. M. Hasan, M. R. Rahman, and K. Sakib, "Mobicomonkey: context testing of android apps," in *MOBILESoft@ICSE*. ACM, 2018, pp. 76–79.

[56] X. Gao, S. H. Tan, Z. Dong, and A. Roychoudhury, "Android testing via synthetic symbolic execution," in *ASE*. ACM, 2018, pp. 419–429.

[57] Y. Cao, G. Wu, W. Chen, and J. Wei, "Crawldroid: Effective model-based GUI testing of android apps," in *Internetware*. ACM, 2018, pp. 19:1–19:6.

[58] W. Zhao, Z. Ding, M. Xia, and Z. Qi, "Systematically testing and diagnosing responsiveness for android apps," in *ICSME*. IEEE, 2019, pp. 449–453.

[59] W. Ravelo-Méndez, C. Escobar-Vélasquez, and M. Linares-Vásquez, "Kraken-mobile: Cross-device interaction-based testing of android apps," in *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2019, pp. 410–413.

[60] C. Cao, J. Deng, P. Yu, Z. Duan, and X. Ma, "Paraaim: Testing android applications parallel at activity granularity," in *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1, 2019, pp. 81–90.

[61] K. Yang, J. Zhuge, Y. Wang, L. Zhou, and H. Duan, "Intentfuzzer: detecting capability leaks of android applications," in *AsiaCCS*. ACM, 2014, pp. 531–536.

[62] Y. Kang, Y. Zhou, M. Gao, Y. Sun, and M. R. Lyu, "Experience report: Detecting poor-responsive UI in android applications," in *ISSRE*. IEEE Computer Society, 2016, pp. 490–501.

[63] Y. Kang, Y. Zhou, H. Xu, and M. R. Lyu, "Diagdroid: Android performance diagnosis via anatomizing asynchronous executions," in *SIGSOFT FSE*. ACM, 2016, pp. 410–421.

[64] M. Xia, L. Gong, Y. Lyu, Z. Qi, and X. Liu, "Effective real-time android application auditing," in *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2015, pp. 899–914.

[65] Y. Li, Z. Yang, Y. Guo, and X. Chen, "Droidbot: a lightweight ui-guided test input generator for android," in *ICSE (Companion Volume)*. IEEE Computer Society, 2017, pp. 23–26.

[66] O. Riganelli, S. P. Mottadelli, C. Rota, D. Micucci, and L. Mariani, "Data loss detector: Automatically revealing data loss bugs in android apps," in *Proc. ISSTA*. ACM, 2020, pp. 141–152.

[67] H. Ye, S. Cheng, L. Zhang, and F. Jiang, "Droidfuzzer: Fuzzing the android apps with intent-filter tag," in *MoMM*. ACM, 2013, p. 68.

[68] Z. Dong, M. Böhme, L. Cojocaru, and A. Roychoudhury, "Time-travel testing of android apps," in *ICSE*, 05 2020.

[69] L. Ravindranath, S. Nath, J. Padhye, and H. Balakrishnan, "Automatic and scalable fault detection for mobile applications," in *MobiSys*. ACM, 2014, pp. 190–203.

[70] S. Hao, B. Liu, S. Nath, W. G. J. Halfond, and R. Govindan, "PUMA: programmable ui-automation for large-scale dynamic analysis of mobile apps," in *MobiSys*. ACM, 2014, pp. 204–217.

[71] X. Zeng, D. Li, W. Zheng, F. Xia, Y. Deng, W. Lam, W. Yang, and T. Xie, "Automated test input generation for android: are we really there yet in an industrial case?" in *SIGSOFT FSE*. ACM, 2016, pp. 987–992.

[72] K. Jamrozik and A. Zeller, "Droidmate: A robust and extensible test generator for android," in *2016 IEEE/ACM International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, 2016, pp. 293–294.

[73] H. Wu, Y. Wang, and A. Rountev, "Sentinel: Generating GUI tests for Android sensor leaks," in *IEEE/ACM International Workshop on Automation of Software Test*, 2018, pp. 27–33.

[74] G. Wu, Y. Cao, W. Chen, J. Wei, H. Zhong, and T. Huang, "Appcheck: A crowdsourced testing service for android applications," in *2017 IEEE International Conference on Web Services (ICWS)*, 2017, pp. 253–260.

[75] T. Ki, C. M. Park, K. Dantu, S. Y. Ko, and L. Ziarek, "Mimic: Ui compatibility testing system for android apps," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, 2019, pp. 246–256.

[76] M. Pan, A. Huang, G. Wang, T. Zhang, and X. Li, "Reinforcement learning based curiosity-driven testing of android applications," in *ISSTA*. ACM, 2020, pp. 153–164.

[77] Y. L. Arnatovich, L. Wang, M. N. Ngo, and C. Soh, "Mobolic: An automated approach to exercising mobile application guis using symbiosis of online testing technique and customated input generation," *Softw. Pract. Exp.*, vol. 48, no. 5, pp. 1107–1142, 2018.

[78] M. Linares-Vásquez, "Enabling testing of android apps," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 2, 2015, pp. 763–765.

[79] K. Moran, M. L. Vásquez, C. Bernal-Cárdenas, and D. Poshyvanyk, "FUSION: a tool for facilitating and augmenting android bug reporting," in *ICSE (Companion Volume)*. ACM, 2016, pp. 609–612.

[80] D. Amalfitano, A. R. Fasolino, P. Tramontana, B. D. Ta, and A. M. Memon, "Mobiguitar: Automated model-based testing of mobile apps," *IEEE Softw.*, vol. 32, no. 5, pp. 53–59, 2015.

[81] H. Shahriar, S. North, and E. Mawangi, "Testing of memory leak in android applications," in *HASE*. IEEE Computer Society, 2014, pp. 176–183.

[82] A. Avancini and M. Ceccato, "Security testing of the communication among android applications," in *AST*. IEEE Computer Society, 2013, pp. 57–63.

[83] D. Franke, S. Kowalewski, C. Weise, and N. Prakobkosol, "Testing conformance of life cycle dependent properties of mobile applications," in *ICST*. IEEE Computer Society, 2012, pp. 241–250.

[84] Q. Huynh, D. Nguyen, N. Ha, T. Pham, P. Nguyen, and V. Tran, "A combinatorial technique for mobile applications software testing," in *2019 11th International Conference on Knowledge and Systems Engineering (KSE)*, 2019, pp. 1–6.

[85] C. Hu and I. Neamtiu, "Automating GUI testing for android applications," in *AST*. ACM, 2011, pp. 77–83.

[86] A. Rohella and S. Takada, "Testing android applications using multi-objective evolutionary algorithms with a stopping criteria," in *SEKE*. KSI Research Inc. and Knowledge Systems Institute Graduate School, 2018, pp. 308–307.

[87] R. Jabbarvand, J. Lin, and S. Malek, "Search-based energy testing of android," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, 2019, pp. 1119–1130.

[88] H. Tang, G. Wu, J. Wei, and H. Zhong, "Generating test cases to expose concurrency bugs in android applications," in *ASE*. ACM, 2016, pp. 648–653.

[89] S. Negara, N. Esfahani, and R. P. Buse, "Practical android test recording with espresso test recorder," in *ICSE-SEIP*. IEEE Press, 2019.

[90] M. Fazzini, E. N. D. A. Freitas, S. R. Choudhary, and A. Orso, "Barista: A technique for recording, encoding, and running platform independent android tests," in *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)*, 2017, pp. 149–160.

[91] R. Coppola, L. Ardito, M. Torchiano, and E. Alègroth, "Translation from visual to layout-based android test cases: a proof of concept," in *2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2020, pp. 74–83.

[92] D. Adamo, D. Nurmuradov, S. Piparia, and R. C. Bryce, "Combinatorial-based event sequence testing of android applications," *Inf. Softw. Technol.*, vol. 99, pp. 98–117, 2018.

[93] A. Ali, H. A. Maghawry, and N. L. Badr, "Automated parallel GUI testing as a service for mobile applications," *J. Softw. Evol. Process.*, vol. 30, no. 10, 2018.

[94] A. Rosenfeld, O. Kardashov, and O. Zang, "Automation of android applications functional testing using machine learning activities classification," in *MOBILESoft@ICSE*. ACM, 2018, pp. 122–132.

[95] W. Xiong, S. Chen, Y. Zhang, M. Xia, and Z. Qi, "Reproducible interference-aware mobile testing," in *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2018, pp. 36–47.

[96] W. Song, X. Qian, and J. Huang, "Ehbdroid: beyond GUI testing for android applications," in *ASE*. IEEE Computer Society, 2017, pp. 27–37.

[97] A. Banerjee, L. K. Chong, S. Chattopadhyay, and A. Roychoudhury, "Detecting energy bugs and hotspots in mobile apps," in *SIGSOFT FSE*. ACM, 2014, pp. 588–598.

[98] G. Hu, X. Yuan, Y. Tang, and J. Yang, "Efficiently, effectively detecting mobile app bugs with appdoctor," in *EuroSys*. ACM, 2014, pp. 18:1–18:15.

[99] P. McAfee, M. Wiem Mkaouer, and D. E. Krutz, "Cate: Concolic android testing using java pathfinder for android applications," in *2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, 2017, pp. 213–214.

[100] A. Allevato and S. H. Edwards, "Robolift: engaging CS2 students with testable, automatically evaluated android applications," in *SIGCSE*. ACM, 2012, pp. 547–552.

[101] T. Griebe and V. Gruhn, "A model-based approach to test automation for context-aware mobile applications," in *SAC*. ACM, 2014, pp. 420–427.