[34] Cook JE, Wolf AL. Discovering models of software processes from event-based data. ACM Trans. on Software Engineering and Methodology, 1998,7(3):215−249. [doi: 10.1145/287000.287001]

[35] Dallmeier V, Lindig C, Wasylkowski A, Zeller A. Mining object behavior with ADABU. In: Proc. of the 2006 Int'l Workshop on Dynamic Systems Analysis. New York: ACM Press, 2006. 17−24. [doi: 10.1145/1138912.1138918]

[36] Pradel M, Gross TR. Automatic generation of object usage specifications from large method traces. In: Proc. of the 2009 IEEE/ACM Int'l Conf. on Automated Software Engineering. Washington: IEEE Computer Society, 2009. 371−382. [doi: 10.1109/ase.2009.60]

[37] Li Z, Zhou Y. PR-Miner: Automatically extracting implicit programming rules and detecting violations in large software code. SIGSOFT Software Engineering Notes, 2005,30(5):306−315. [doi: 10.1145/1095430.1081755]

[38] Livshits B, Zimmermann T. DynaMine: Finding common error patterns by mining software revision histories. SIGSOFT Software Engineering Notes, 2005,30(5):296−305. [doi: 10.1145/1095430.1081754]

[39] Yang J, Evans D, Bhardwaj D, Bhat T, Das M. Perracotta: Mining temporal API rules from imperfect traces. In: Proc. of the 28th Int'l Conf. on Software Engineering. New York: ACM Press, 2006. 282−291. [doi: 10.1145/1134285.1134325]

[40] Thummalapenta S, Xie T. Alattin: Mining alternative patterns for defect detection. Automated Software Engineering, 2011,18(3): 293−323. [doi: 10.1007/s10515-011-0086-z]

[41] Liang B, Bian P, Zhang Y, Shi W, You W, Cai Y. AntMiner: Mining more bugs by reducing noise interference. In: Proc. of the 38th Int'l Conf. on Software Engineering. New York: ACM Press, 2016. 333−344. [doi: 10.1145/2884781.2884870]

[42] Murali V, Chaudhuri S, Jermaine C. Bayesian specification learning for finding API usage errors. In: Proc. of the 2017 11th Joint Meeting on Foundations of Software Engineering. New York: ACM Press, 2017. 151−162. [doi: 10.1145/3106237.3106284]

[43] Gabel M, Su Z. Symbolic mining of temporal specifications. In: Proc. of the 30th Int'l Conf. on Software Engineering. New York: ACM Press, 2008. 51−60. [doi: 10.1145/1368088.1368096]

[44] Gabel M, Su Z. Javert: fully automatic mining of general temporal properties from dynamic traces. In: Proc. of the 16th ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering. New York: ACM Press, 2008. 339−349. [doi: 10.1145/1453101.1453150]

[45] Gabel M, Su Z. Online inference and enforcement of temporal properties. In: Proc. of the 32nd ACM/IEEE Int'l Conf. on Software Engineering—Vol.1. New York: ACM Press, 2010. 15−24. [doi: 10.1145/1806799.1806806]

[46] Zhong H, Zhang L, Xie T, Mei H. Inferring resource specifications from natural language API documentation. In: Proc. of the 2009 IEEE/ACM Int'l Conf. on Automated Software Engineering. Washington: IEEE Computer Society, 2009. 307−318. [doi: 10.1109/ase.2009.94]

[47] Nguyen TT, Nguyen HA, Pham NH, Al-Kofahi JM, Nguyen TN. Graph-Based mining of multiple object usage patterns. In: Proc. of the the 7th Joint Meeting of the European Software Engineering Conf. and the ACM SIGSOFT Symp. on The Foundations of Software Engineering. New York: ACM Press, 2009. 383−392. [doi: 10.1145/1595696.1595767]

[48] Wasylkowski A, Zeller A. Mining temporal specifications from object usage. In: Proc. of the 2009 IEEE/ACM Int'l Conf. on Automated Software Engineering. Washington: IEEE Computer Society, 2009. 295−306. [doi: 10.1109/ase.2009.30]

[49] Wei Y, Furia CA, Kazmin N, Meyer B. Inferring better contracts. In: Proc. of the 33rd Int'l Conf. on Software Engineering. New York: ACM Press, 2011. 191−200. [doi: 10.1145/1985793.1985820]

[50] Nguyen HA, Dyer R, Nguyen TN, Rajan H. Mining preconditions of APIs in large-scale code corpus. In: Proc. of the 22nd ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering. New York: ACM Press, 2014. 166−177. [doi: 10.1145/2635868.2635924]

[51] Henkel J, Reichenbach C, Diwan A. Discovering documentation for Java container classes. IEEE Trans. on Software Engineering, 2007,33(8):526−543. [doi: 10.1109/tse.2007.70705].

[52] Ramanathan MK, Grama A, Jagannathan S. Path-Sensitive inference of function precedence protocols. In: Proc. of the 29th Int'l Conf. on Software Engineering. Washington: IEEE Computer Society, 2007. 240−250. [doi: 10.1109/icse.2007.63]

[53] Lorenzoli D, Mariani L, Pezzè M. Automatic generation of software behavioral models. In: Proc. of the 30th Int'l Conf. on Software engineering. New York: ACM Press, 2008. 501−510. [doi: 10.1145/1368088.1368157]

[54] Krka I, Brun Y, Medvidovic N. Automatic mining of specifications from invocation traces and method invariants. In: Proc. of the 22nd ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering. New York: ACM Press, 2014. 178−189. [doi: 10.1145/2635868.2635890]

[55] Dwyer MB, Avrunin GS, Corbett JC. Patterns in property specifications for finite-state verification. In: Proc. of the 21st Int'l Conf. on Software Engineering. New York: ACM Press, 1999. 411−420. [doi: 10.1145/302405.302672]

[56] Robillard MP, Bodden E, Kawrykow D, Mezini M, Ratchford T. Automated API property inference techniques. IEEE Trans. on Software Engineering, 2013,39(5):613−637. [doi: 10.1109/tse.2012.63]

[57] Engler D, Chen DY, Hallem S, Chou A, Chelf B. Bugs as deviant behavior: A general approach to inferring errors in systems code. SIGOPS Operating Systems Review, 2001,35(5):57−72. [doi: 10.1145/502059.502041]

[58] Qu JF, Liu M. Mining frequent itemsets using node-sets of a prefix-tree. In: Proc. of the 23rd Int'l Conf. on Database and Expert Systems Applications (DEXA 2012). Part I. Vienna, Berlin, Heidelberg: Springer-Verlag, 2012. 453−467. [doi: 10.1007/978-3-642-32600-4_34]

[59] Agrawal R, Srikant R. Fast algorithms for mining association rules in large databases. In: Proc. of the 20th Int'l Conf. on Very Large Data Bases. San Francisco: Morgan Kaufmann Publishers Inc., 1994. 487−499.

[60] Kagdi H, Collard ML, Maletic JI. Comparing approaches to mining source code for call-usage patterns. In: Proc. of the 4th Int'l Workshop on Mining Software Repositories. Washington: IEEE Computer Society, 2007. 20. [doi: 10.1109/msr.2007.3]

[61] Kagdi H, Collard ML, Maletic JI. An approach to mining call-usage patternswith syntactic context. In: Proc. of the 22nd IEEE/ACM Int'l Conf. on Automated Software Engineering. New York: ACM Press, 2007. 457−460. [doi: 10.1145/1321631.1321708]

[62] Thummalapenta S, Xie T. Mining exception-handling rules as sequence association rules. In: Proc. of the 31st Int'l Conf. on Software Engineering. Washington: IEEE Computer Society, 2009. 496−506. [doi: 10.1109/icse.2009.5070548]

[63] Lo D, Khoo SC. SMArTIC: Towards building an accurate, robust and scalable specification miner. In: Proc. of the 14th ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering. New York: ACM Press, 2006. 265−275. [doi: 10.1145/1181775.1181808]

[64] Lorenzoli D, Mariani L, Pezz M, #232. Automatic generation of software behavioral models. In: Proc. of the 30th Int'l Conf. on Software Engineering. New York: ACM Press, 2008. 501−510. [doi: 10.1145/1368088.1368157]

[65] Walkinshaw N, Bogdanov K. Inferring finite-state models with temporal constraints. In: Proc. of the 2008 23rd IEEE/ACM Int'l Conf. on Automated Software Engineering. Washington: IEEE Computer Society, 2008. 248−257. [doi: 10.1109/ase.2008.35]

[66] Ammons G, Bod R, #237, Larus JR. Mining specifications. SIGPLAN Notices, 2002,37(1):4−16. [doi: 10.1145/565816.503275]

[67] Chang RY, Podgurski A, Yang J. Finding what's not there: A new approach to revealing neglected conditions in software. In: Proc. of the 2007 Int'l Symp. on Software Testing and Analysis. New York: ACM Press, 2007. 163−173. [doi: 10.1145/1273463.1273486]

[68] Zhong H, Zhang L, Mei H. Inferring specifications of object oriented APIs from API source code. In: Proc. of the 15th Asia-Pacific Software Engineering Conf. (APSEC 2008). IEEE, 2008. 221−228. [doi: 10.1109/APSEC.2008.54]

[69] Meyer B. Applying "Design by Contract". Computer, 1992,25(10):40−51. [doi: 10.1109/2.161279]

[70] Ernst MD, Cockrell J, Griswold WG, Notkin D. Dynamically discovering likely program invariants to support program evolution. In: Proc. of the 21st Int'l Conf. on Software Engineering. New York: ACM Press, 1999. 213−224. [doi: 10.1145/302405.302467]

[71] Csallner C, Tillmann N, Smaragdakis Y. DySy: Dynamic symbolic execution for invariant inference. In: Proc. of the 30th Int'l Conf. on Software Engineering. New York: ACM Press, 2008. 281−290. [doi: 10.1145/1368088.1368127]

[72] Ernst MD, Perkins JH, Guo PJ, McCamant S, Pacheco C, Tschantz MS, Xiao C. The Daikon system for dynamic detection of likely invariants. Science of Computer Programming, 2007,69(1-3):35−45.[doi: 10.1016/j.scico.2007.01.015]

[73] Flanagan C, Leino KRM. Houdini, an annotation assistant for ESC/Java. In: Proc. of the Int'l Symp. of Formal Methods Europe on Formal Methods for Increasing Software Productivity. London: Springer-Verlag, 2001. 500−517.

[74] Guttag JV, Horning JJ. The algebraic specification of abstract data types. Acta informatica, 1978,10(1):27−52. [doi: 10.1007/bf00260922]

[75] Bagge AH, Haveraaen M. Specification of generic APIs, or: Why algebraic may be better than pre/post. Ada Letters, 2014,34(3):71−80. [doi: 10.1145/2692956.2663183]

[76] Legunsen O, Hassan WU, Xu X, Roşu G, Marinov D. How good are the specs? A study of the bug-finding effectiveness of existing Java API specifications. In: Proc. of the 31st IEEE/ACM Int'l Conf. on Automated Software Engineering. New York: ACM Press, 2016. 602−613. [doi: 10.1145/2970276.2970356]

[77] Thung F. API recommendation system for software development. In: Proc. of the 31st IEEE/ACM Int'l Conf. on Automated Software Engineering. New York: ACM Press, 2016. 896−899. [doi: 10.1145/2970276.2975940]

[78] Teyton C, Falleri JR, Blanc X. Mining library migration graphs. In: Proc. of the 2012 19th Working Conf. on Reverse Engineering. Washington: IEEE Computer Society, 2012. 289−298. [doi: 10.1109/wcre.2012.38]

[79] Thung F, Lo D, Lawall J. Automated library recommendation. In: Proc. of the 2013 20th Working Conf. on Reverse Engineering (WCRE). 2013. 182−191.

[80] Chen C, Xing Z. SimilarTech: Automatically recommend analogical libraries across different programming languages. In: Proc. of the 31st IEEE/ACM Int'l Conf. on Automated Software Engineering. New York: ACM Press, 2016. 834−839. [doi: 10.1145/2970276.2970290]

[81] Xie T, Pei J. MAPO: Mining API usages from open source repositories. In: Proc. of the 2006 Int'l Workshop on Mining Software Repositories. New York: ACM Press, 2006. 54−57. [doi: 10.1145/1137983.1137997]

[82] Hindle A, Barr ET, Su Z, Gabel M, Devanbu P. On the naturalness of software. In: Proc. of the 34th Int'l Conf. on Software Engineering. Piscataway: IEEE Press, 2012. 837−847.

[83] Wang J, Dang Y, Zhang H, Chen K, Xie T, Zhang D. Mining succinct and high-coverage API usage patterns from source code. In: Proc. of the 10th Working Conf. on Mining Software Repositories. Piscataway: IEEE Press, 2013. 319−328.

[84] Raychev V, Vechev M, Yahav E. Code completion with statistical language models. SIGPLAN Notices, 2014,49(6):419−428. [doi: 10.1145/2666356.2594321]

[85] Nguyen AT, Nguyen TN. Graph-Based statistical language model for code. In: Proc. of the 37th Int'l Conf. on Software Engineering—Vol.1. Piscataway: IEEE Press, 2015. 858−868.

[86] Nguyen TT, Pham HV, Vu PM, Nguyen TT. Learning API usages from bytecode: A statistical approach. In: Proc. of the 38th Int'l Conf. on Software Engineering. New York: ACM Press, 2016. 416−427. [doi: 10.1145/2884781.2884873]

[87] Fowkes J, Sutton C. Parameter-Free probabilistic API mining across GitHub. In: Proc. of the 2016 24th ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering. New York: ACM Press, 2016. 254−265. [doi: 10.1145/2950290.2950319]

[88] Gu X, Zhang H, Zhang D, Kim S. Deep API learning. In: Proc. of the 2016 24th ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering. New York: ACM Press, 2016. 631−642. [doi: 10.1145/2950290.2950334]

[89] Zhang C, Yang J, Zhang Y, Fan J, Zhang X, Zhao J, Ou P. Automatic parameter recommendation for practical API usage. In: Proc. of the 34th Int'l Conf. on Software Engineering. Piscataway: IEEE Press, 2012. 826−836.

[90] Asaduzzaman M, Roy CK, Monir S, Schneider KA. Exploring API method parameter recommendations. In: Proc. of the 2015 IEEE Int'l Conf. on Software Maintenance and Evolution (ICSME). Washington: IEEE Computer Society, 2015. 271−280. [doi: 10.1109/icsm.2015.7332473]

[91] Dig D, Johnson R. How do APIs evolve? a story of refactoring: Research articles. Journal of Software Maintenance and Evolution, 2006,18(2): 83−107. [doi: 10.1002/smr.v18:2]

[92] Dagenais B, Robillard MP. Recommending adaptive changes for framework evolution. ACM Transactions on Software Engineering and Methodology, 2011,20(4):1−35. [doi: 10.1145/2000799.2000805]

[93] Zhong H, Thummalapenta S, Xie T, Zhang L, Wang Q. Mining API mapping for language migration. In: Proc. of the 32nd ACM/IEEE Int'l Conf. on Software Engineering—Vol.1. New York: ACM Press, 2010. 195−204. [doi: 10.1145/1806799.1806831]

[94] Nguyen TD, Nguyen AT, Phan HD, Nguyen TN. Exploring API embedding for API usages and applications. In: Proc. of the 39th Int'l Conf. on Software Engineering. Piscataway: IEEE Press, 2017. 438−449. [doi: 10.1109/icse.2017.47]

[95] Wang W, Godfrey MW. Detecting API usage obstacles: A study of iOS and Android developer questions. In: Proc. of the 10th Working Conf. on Mining Software Repositories. Piscataway: IEEE Press, 2013. 61−64.

[96] Agrawal R, Srikant R. Mining sequential patterns. In: Proc. of the 11th Int'l Conf. on Data Engineering. Washington: IEEE Computer Society, 1995. 3−14.

[97] Srikant R, Agrawal R. Mining sequential patterns: Generalizations and performance improvements. In: Proc. of the 5th Int'l Conf. on Extending Database Technology: Advances in Database Technology. London: Springer-Verlag, 1996. 3−17.

[98] Pei J, Han J, Mortazavi-Asl B, Pinto H, Chen Q, Dayal U, Hsu MC. PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In: Proc. of the 17th Int'l Conf. on Data Engineering. IEEE, 2001. 215−224.

[99] Zaki MJ. SPADE: An efficient algorithm for mining frequent sequences. Machine Learning, 2001,42(1-2):31−60. [doi: 10.1023/a:1007652502315]

[100] Ayres J, Flannick J, Gehrke J, Yiu T. Sequential PAttern mining using a bitmap representation. In: Proc. of the 8th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. New York: ACM Press, 2002. 429−435. [doi: 10.1145/775047.775109]

[101] Wang J, Han J. BIDE: Efficient mining of frequent closed sequences. In: Proc. of the 20th Int'l Conf. on Data Engineering. Washington: IEEE Computer Society, 2004. 79.

[102] Tatti N, Vreeken J. The long and the short of it: Summarising event sequences with serial episodes. In: Proc. of the 18th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. New York: ACM Press, 2012. 462−470. [doi: 10.1145/2339530.2339606]

[103] Lam HT, Mörchen F, Fradkin D, Calders T. Mining compressing sequential patterns. Statistical Analysis and Data Mining: The ASA Data Science Journal, 2014,7(1):34−52.

[104] Rosenfeld R. Two decades of statistical language modeling: Where do we go from here? Proc. of the IEEE, 2000,88(8):1270−1278.

[105] Nguyen AT, Hilton M, Codoban M, Nguyen HA, Mast L, Rademacher E, Nguyen TN, Dig D. API code recommendation using statistical learning from fine-grained changes. In: Proc. of the 2016 24th ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering. New York: ACM Press, 2016. 511−522. [doi: 10.1145/2950290.2950333]

[106] Elman JL. Finding structure in time. Cognitive Science, 1990,14(2):179−211.

[107] Mikolov T, Karafiát M, Burget L, Cernocký J, Khudanpur S. Recurrent neural network based language model. In: Proc. of the INTERSPEECH 2010, Conf. of the Int'l Speech Communication Association. 2010. 1045−1048.

[108] Cho K,Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Proc. of the EMNLP. 2014. 1724−1734.

[109] Pradel M, Heiniger S, Gross TR. Static detection of brittle parameter typing. In: Proc. of the 2012 Int'l Symp. on Software Testing and Analysis. New York: ACM Press, 2012. 265−275. [doi: 10.1145/2338965.2336785]

[110] Pradel M, Gross TR. Detecting anomalies in the order of equally-typed method arguments. In: Proc. of the 2011 Int'l Symp. on Software Testing and Analysis. New York: ACM Press, 2011. 232−242. [doi: 10.1145/2001420.2001448]

[111] Xing Z, Stroulia E. API-Evolution support with diff-CatchUp. IEEE Trans. on Software Engineering, 2007,33(12):818−836. [doi: 10.1109/ tse.2007.70747]

[112] Dig D, Comertoglu C, Marinov D, Johnson R. Automated detection of refactorings in evolving components. In: Proc. of the 20th European Conf. on Object-Oriented Programming. Berlin: Springer-Verlag, 2006. 404−428. [doi: 10.1007/11785477_24]

[113] Schäfer T, Jonas J, Mezini M. Mining framework usage changes from instantiation code. In: Proc. of the 30th Int'l Conf. on Software Engineering. New York: ACM Press, 2008. 471−480. [doi: 10.1145/1368088.1368153]

[114] Roover CD, Lammel R, Pek E. Multi-Dimensional exploration of API usage. In: Proc. of the IEEE Int'l Conf. on Program Comprehension. IEEE, 2013. 152−161. [doi: 10.1109/ICPC.2013.6613843]

[115] McIlroy MD. Mass-Produced software components. In: Proc. of the 1st Int'l Conf. on Software Engineering. 1968. 88−98.

[116] Parnas DL. On the criteria to be used in decomposing systems into modules. Communications of the ACM, 1972,15(12):1053−1058. [doi: 10.1145/361598.361623]

[117] Plauger PJ. The Standard C Library. Prentice Hall PTR, 1991. 17−415.

[118] Tsai CC, Jain B, Abdul NA, Porter DE. A study of modern Linux API usage and compatibility: What to support when you're supporting. In: Proc. of the 11th European Conf. on Computer Systems. New York: ACM Press, 2016. 1−16. [doi: 10.1145/2901318.2901341]

[119] Atlidakis V, Andrus J, Geambasu R, Mitropoulos D, Nieh J. POSIX abstractions in modern operating systems: The old, the new, and the missing. In: Proc. of the 11th European Conf. on Computer Systems. New York: ACM Press, 2016. 1−17. [doi: 10.1145/2901318.2901350]

[120] Qiu D, Li B, Leung H. Understanding the API usage in Java. Information and Software Technology, 2016,73(C):81−100. [doi: 10.1016/j.infsof. 2016.01.011]

[121] Bissyandé TF, Réveillère L, Lawall JL, Muller G. Ahead of time static analysis for automatic generation of debugging interfaces to the Linux kernel. Automated Software Engineering, 2016,23(1):3−41. [doi: 10.1007/s10515-014-0152-4]

[122] Bissyandé TF, Réveillère L, Lawall JL, Muller G. Diagnosys: Automatic generation of a debugging interface to the Linux kernel. In: Proc. of the 27th IEEE/ACM Int'l Conf. on Automated Software Engineering. New York: ACM Press, 2012. 60−69. [doi: 10.1145/2351676.2351686]

[123] Jezek K, Dietrich J, Brada P. How Java APIs break—An empirical study. Information and Software Technology, 2015,65(C): 129−146. [doi: 10.1016/ j.infsof.2015.02.014]

[124] Lin Z, Zhong H, Chen Y, Zhao J. LockPeeker: Detecting latent locks in Java APIs. In: Proc. of the 31st IEEE/ACM Int'l Conf. on Automated Software Engineering. New York: ACM Press, 2016. 368−378. [doi: 10.1145/2970276.2970355]

[125] Businge J, Serebrenik A, van den Brand M. Analyzing the Eclipse API usage: Putting the developer in the loop. In: Proc. of the 2013 17th European Conf. on Software Maintenance and Reengineering (CSMR). IEEE, 2013. 37−46. [doi: 10.1109/CSMR.2013.14]

[126] Businge J, Serebrenik A, van den Brand MGJ. Eclipse API usage: The good and the bad. Software Quality Journal, 2015,23(1): 107−141. [doi: 10.1007/s11219-013-9221-3]

[127] Businge J, Brand MVD, Serebrenik A. Survival of Eclipse third-party plug-ins. In: Proc. of the 2012 IEEE Int'l Conf. on Software Maintenance (ICSM). Washington: IEEE Computer Society, 2012. 368−377. [doi: 10.1109/icsm.2012.6405295]

[128] McDonnell T, Ray B, Kim M. An empirical study of API stability and adoption in the android ecosystem. In: Proc. of the 2013 IEEE Int'l Conf. on Software Maintenance. Washington: IEEE Computer Society, 2013. 70−79. [doi: 10.1109/icsm.2013.18]

[129] Bavota G, Linares-Vasquez M, Bernal-Cardenas CE, Di Penta M, Oliveto R, Poshyvanyk D. The impact of API change-and fault-proneness on the user ratings of android apps. IEEE Trans. on Software Engineering, 2015,41(4):384−407. [doi: 10.1109/TSE.2014.2367027]

[130] Li L, Bissyandé TF, Le Traon Y, Klein J. Accessing inaccessible android APIs: An empirical study. In: Proc. of the 2016 IEEE Int'l Conf. on Software Maintenance and Evolution (ICSME). IEEE, 2016. 411−422.

[131] Han J, Kywe SM, Yan Q, Bao F, Deng R, Gao D, Li Y, Zhou J. Launching generic attacks on iOS with approved third-party applications. In: Proc. of the 11th Int'l Conf. on Applied Cryptography and Network Security. Berlin: Springer-Verlag, 2013. 272−289. [doi: 10.1007/978-3-642-38980-1_17]

[132] Wang T, Lu K, Lu L, Chung S, Lee W. Jekyll on iOS: When benign apps become evil. In: Proc. of the 22nd USENIX Conf. on Security. Berkeley: USENIX Association, 2013. 559−572.

[133] Deng Z, Saltaformaggio B, Zhang X, Xu D. iRiS: Vetting private API abuse in iOS applications. In: Proc. of the 22nd ACM SIGSAC Conf. on Computer and Communications Security. New York: ACM Press, 2015. 44−56. [doi: 10.1145/2810103.2813675]

[134] Zibran MF, Eishita FZ, Roy CK. Useful, but usable? Factors affecting the usability of APIs. In: Proc. of the 2011 18th Working Conf. on Reverse Engineering. Washington: IEEE Computer Society, 2011. 151−155. [doi: 10.1109/wcre.2011.26]

[135] Ellis B, Stylos J, Myers B. The factory pattern in API design: A usability evaluation. In: Proc. of the 29th Int'l Conf. on Software Engineering. Washington: IEEE Computer Society, 2007. 302−312. [doi: 10.1109/icse.2007.85]

[136] Stylos J, Myers BA. The implications of method placement on API learnability. In: Proc. of the 16th ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering. New York: ACM Press, 2008. 105−112. [doi: 10.1145/1453101.1453117]

[137] Wu W, Guéhéneuc YG, Antoniol G, Kim M. AURA: A hybrid approach to identify framework evolution. In: Proc. of the 32nd ACM/IEEE Int'l Conf. on Software Engineering—Vol.1. New York: ACM Press, 2010. 325−334. [doi: 10.1145/1806799.1806848]

[138] Iyer S, Konstas I, Cheung A, Zettlemoyer L. Summarizing source code using a neural attention model. In: Proc. of the Meeting of the Association for Computational Linguistics. 2016. 2073−2083.

李正(1985－),男,天津人,博士生,主要研究领域为软件工程,系统安全.

吴敬征(1982－),男,博士,副研究员,主要研究领域为系统安全,漏洞挖掘,移动安全.

李明树(1965－),男,博士,研究员,博士生导师,CCF 会士,主要研究领域为操作系统深度设计,可信软件过程以及基础软硬件核心技术与应用.