**Varun lines**

1. Future work might investigate further the accuracy with regard to the distribution of online snippets: an aspect that, due to the strict design of a BF method, at this stage was not achievable.
2. We will combine TaskKG4Q with existing automatic Q&A systems (e.g., AnswerBot or APIBot).
3. We will investigate semantic grouping that aims to recognize both interaction and content semantics of perceptual groups.
4. With RoboFuzz, we attempt to excite the faults in robotic systems by injecting valid (i.e., correctly formatted) messages on the ROS level.
5. Future studies can evaluate whether the effect of warnings’ position influences developers. This effect might be particularly significant.
6. We will evaluate AutoPruner on call graphs constructed using additional static analysis tools and for other programming languages, as well as assess the impact on call graph pruning on other client analyses.
7. We will continue to enrich the dataset and dedicated on proposing better Review2Code approach.
8. It would be interesting to investigate the results of the combination of our approach with TIE.
9. Our future work will focus on the following topics: Fuzzer integration, Testing strategies, Constraint synthesis.
10. Future research can investigate whether developers can improve their understanding of the security impact of the code changes through the use of code change visualization tools.
11. To study the vulnerability proneness of single-language projects-prior works examined single-language code’s proneness to defects in general but did not particularly investigate the proneness to vulnerabilities.
12. We plan to improve our approach by taking into account other JIT optimizations and CPU-level optimizations that also introduce timing side-channels in practice.
13. We may consider different graph representations of programs.
14. We will devise automated program repair techniques for text accessibility issues.
15. Future work will further explore whether AUGER can learn knowledge of projects in different programming languages.
16. Investigating other possible impacts of company domination on the development of OSS ecosystems is also an avenue for future work.
17. We will further investigate the cost-effectiveness of Tscope in real-world practice.
18. In the future, in addition to UQ, we will investigate other importance sampling mechanisms to further enhance the effectiveness of transfer learning.
19. We aim to improve both precision and recall by completing our dataset and investigating larger time windows
20. We want to further explore patterns of interdependencies among feature flag clusters.
21. Ideas for future development include Fixie as a ‘developer’s assistant’ and an IDE-based system.
22. Will work on empirical validation, automated tools for measurement, and best practices for implementation.
23. Future works focus on empirical validation on the claims of PTNNs in model zoos to check for the existence of false advertising.
24. Future research directions, including identifying and analyzing failure causal chains, standardizing the conduct of failure studies, and tool support for faster defect analysis.
25. We will empirically demonstrate PolyFax’s merits in efficiency and effectiveness against real-world open-source projects on GitHub.
26. We plan to do a user study to assess how useful is TAPHSIR in practice.
27. We plan to conduct a user study to assess how useful COREQQA is in practice.
28. SFLKit opens the door for uniform and comparable future research in the area of statistical fault localization.
29. We plan to utilize WikiDoMiner for addressing new analytical problems beyond ambiguity analysis.
30. We will enhance RegMiner by supporting manual correction of recommended critical changes and providing relevant APIs from RegMiner.
31. future work includes an empirical study to investigate the cost and benefit of large-scale mutation testing in Simulink models.
32. We seek to study the expression of negative sentiment in SATD and how developers react to bot actions on GitHub.
33. We plan to build the rest of the framework, integrate it into IDE code inspections, and test it with a number of both positive and negative examples.
34. future research agenda on the topic, mainly focused on designing and developing a new generation of code qualitycheckers.
35. whether your test suite is good at finding subtle errors or you want to only seed faults that are hard to detect (by some testing technique).
36. In our future work, we plan to reduce complexity in creating the ordering using approximation algorithms.
37. We plan to undertake an extensive evaluation of our approach with a large set of programs and test suites in our future work.
38. We will explore these along with scalability challenges, with respect to program size (that exceeds cache size) and number of test cases, in our future work.
39. In the future, we could incorporate code snippets and images into our approach.
40. In the future, we will enhance our approach by incorporating image and code-snippet semantics into our framework.
41. we limit the PeVi test input to the properties of the vehicle and the pedestrian, and leave the problem of testing PeVi for other input properties to the future.
42. In future, we plan to perform more experiments and to further improve our search algorithm by dynamically refining the neural network models using simulations performed during the search.
43. The evaluation of ICSD&MLBDO against the other two types of privacy threats will be our future work.
44. For the future work, we would like to utilize more effort datasets to validate the effectiveness of our approach.
45. The exact optimum should be the subject of future research, as a forgetful FPS could have a higher long-term prediction performance than a time-global.
46. Our findings raise questions open for future research: Can a hybrid of existing algorithms combine the advantages of those algorithms?
47. Future experiments should compare artificial and real evaluation techniques (using the same dataset) across multiple recommenders.
48. future work should investigate assumptions on the expectations used to evaluate recommender systems.
49. we need to analyze whether RSSEs should propose the correct answer or the expected answer
50. In the future, we will capture more information about the type of project and the role of the developer to get a better picture about the task the developer was working on.
51. Using the bellwether, generate quality predictors on new project data. That is, having learned the bellwether on past data, we now apply it to future projects.
52. our future work will consist in further improve AutoJMH to address these situations.
53. In the future, we plan an extended and more systematic user evaluation.
54. We would also like to expand the interactivity of our visualizations by enabling on-the-fly updates to the views following changes in the underlying data layer.
55. we plan to investigate the use of visualization to help IBM FOCUS users compare between different versions of a combinatorial model and between alternative combinatorial test plans.
56. We would like to investigate Rails language extensions that support direct semantic encoding in logic, which would eliminate the need for extraction and logic translation.
57. potential direction for future work is evaluating different types of displays (e.g., different notations and colors) to find the most efficient way of delivering Android inter-app vulnerabilities.
58. We plan to extend SEALANT to support other types of inter-app attacks that exploit covert channels in the Android core system components, such as the file system.
59. In our future work we plan to integrate further statistics views and want to implement warnings informing users about trends such as an increase of violations for a particular constraint.
60. future work to validate detected FIC issues and associated fixes by other means such as crowdsourcing.
61. we obtained several important findings that can facilitate the diagnosis and fixing of FIC issues, and guide future studies on related topics.
62. we proposed to use API-context pairs that capture issue-inducing APIs and issue-triggering contexts to model FIC issues.
63. In future, we plan to explore the possibility of leveraging crowdsourcing to help developers verify certain devicespecific FIC issues and validate possible fixes.
64. plan to enhance the detection capability of our technique by mining new FIC issues in popular apps using the proposed APIcontext framework.
65. In our future research, we expect to develop an automatic GUICC selection technique for automated model-based GUI testing that analyzes the inherent characteristics of the target app and selects adequate criteria.
66. Future research directions for detecting non-termination bugs in real-world projects.
67. Future work might investigate further the accuracy with regard to the distribution of online snippets.
68. we find interesting contracts that require additional language features, including contract composition, recursion, user-defined functions, etc.
69. There are extreme cases where DeCon compiler generates contracts with non-negligible overhead to the reference hand-written code.
70. Save the overhead of run-time verification, we can leverage the high-level abstraction of DeCon programs to perform static verification.
71. For future work, a larger collection of studies, including participation from industry professionals.
72. In the future, we will combine TaskKG4Q with existing automatic Q&A systems (e.g., AnswerBot [72] or APIBot [63]).
73. In the future, Copilot [? ] can be utilized for the automated code recommendation.
74. We will investigate semantic grouping that aims to recognize both interaction and content semantics of perceptual groups.
75. Improvements are planned for future work to improve Burt’s accuracy.
76. We plan to examine FIMAX with other metrics, e.g., Precision and Recall, which are widely used in information retrieval.
77. First, we suggest future research to consider more characteristics of source code, like syntactic structures.
78. we preserve SketchFix’s practicality and leave the improvement of fix patterns [28, 30, 58] as future work.
79. Further understanding of developer activities that takes an integrated perspective combining both role-based relationships and different types of activities beyond technical contributions
80. Mechanism and workflow designs for broadening participation in and sharing nontechnical responsibilities
81. Tool support for relieving elite developers from routine administrative burdens by synthesizing existing techniques to their routine workflow
82. we plan to investigate if software changes can also improve the performance of bug localization based on other IR models besides VSM.
83. plan to incorporate more properties of software changes such as change patterns [10], authorship and their proneness to contain faults [16] into Locus to filter out uninteresting changes
84. We ignore the effects of the tests being at fault in this paper, but here as well point out that this is something to be studied in future work.
85. We ignore the potential effect of multiple faults in this paper. Nevertheless, future research should study the effect of multiple faults.
86. , we plan to evaluate our multiobjective approach on further projects in other different programming languages.
87. we will extend our work to address the problem of the software bugs management and prioritization using multi-objective search techniques.
88. we plan to replicate our experiments for dependence clusters at different granularities (e.g., statement level) and on systems written in other languages and other programming paradigms.
89. One of our future work is to leverage the advanced binary reassembling development toolkits such as Uroboros
90. future work to speed up offline taint analysis is to construct a recompilable straightline program from execution trace.
91. We plan to use IncA in the future to enforce secure coding standards (e.g. Misra, CERT) in mbeddr.
92. In future, we will apply RPro to trigger other kinds of concurrency bugs.
93. We plan to find a solution to exploring the environment variables for API methods and revealing latent locks on Se in our future work.
94. we plan to extend our technique to handle the remaining query node types and release our tool for endto-end automated testing of PostgreSQL procedures.
95. we intend to make seamless symbolic execution from Java applications to PostgreSQL stored procedures that can share a symbolic map.
96. working on a technique to automatically generate larger tables if it enables a particular path exploration in code to be executed.
97. An input model is constructed once and can be used across all future testing sessions.
98. we will integrate the analysis of Pthreads synchronisation primitives other than mutexes, e.g. condition variables, and extend our algorithm to Java synchronisation constructs.
99. As future work, we plan to continue ongoing development on an open source TM tool Capra,10 which will be used and evaluated in the context of the Amalthea4public project.
100. we aim to address especially configurability (G7) and integrating manual and automation techniques (G9) better than existing TM solutions, applying model-driven technologies to enable truly domain-specific traceability solutions.
101. In the future, CI will have an even greater influence than it has today.
102. We are considering interoperability of our tool with mindmapping tools.
103. we are working on traceability mechanisms between the meta-model and the DSL-map.
104. we are improving the transformation to permit generating metamodel packages from the ideas
105. providing a platform for rapidly integrating new information resources or performing complex data manipulation.
106. we wish to study the generality of our algorithm in other verification domains that rely on annotations for constructing program proofs.
107. One source of the client code might be the library itself, which often call into its own public API.
108. we conclude with a discussion of observed characteristics that may inform the design of future analysis approaches and may also be informative for developers concerned about interactions in their code
109. we would like to explore the possibility of applying delta refinement to debugging and fault localization
110. see room for improvement in terms of performance by combining different slicing approaches and parallelizing test executions as much as possible.
111. , a source of incompleteness of our approach is related to the way we interpret divergent tableau paths, applying a kind of generalisation through the application of temporal operators.
112. considering the use of inductive learning techniques to support the inference of more generalised expressions.
113. In future we will extend our tool to take a set of UCS for a given problem as input, and to generate the analysis class diagram for the problem.
114. We made the following experimental material along with other details of the experiment available at8 for other researchers of this domain to replicate such experiments in future.
115. we plan to further investigate continuous design-flaw detection scenarios by emulating entire version histories available in real-life software repositories.
116. we plan to conduct further long-term experiments on continuously evolving systems, based on program edits derived from version histories of real-life software repositories.
117. We also plan to extend our pattern language with logical constructs such as ‘or’, enlarge the catalog of design flaws using our framework and to conduct user studies
118. For future work, further development tools will be integrated with our infrastructure (i.e., create tool-adapters).
119. We leave it for future work to experimentally compare our tool with ICPL; these tools assume different input formats.
120. We leave it for future work to implement our constraint handling approach in the IPOG algorithm to fairly compare our constraint handling and the minimum forbidden tuple approach.
121. future work to develop such a framework where one can plug-in their own OTAT heuristics (such as the AETG or PICT heuristics), without being concerned about constraint handling.
122. In future work we plan on investigating approaches for summarizing multiple assertion that are similar to the action dependency graph
123. we plan to enhance this part of the technique to detect more types of matches.
124. dynamic program analysis could be an alternative technique for identifying the information of a test.
125. although it is difficult to summarize and name large and complex test bodies with multiple assertions, our technique may be modified to help test developers split a large test into several smaller tests with appropriate names.
126. we plan to improve BIT-TAG in the following directions with our own interests.
127. we plan to optimize the performance of existing constraint handling and also include support for new types of constraints to better capture requirements.
128. we will conduct experiments on more and even larger data sets, and compare BIT-TAG with other approaches
129. we plan to use machine-learning techniques to identify important test values from the original data source
130. we would like to refactor BIT-TAG to support more data types such as spatial types.
131. BIT-TAG could be improved to reuse existing data sets to generate a new data set when constraints change.
132. , generating a minimal (in terms of row counts) test data set to satisfy all constraints is an interesting problem to study.
133. In the future work, we will first complete the implementation of our test case generation tool.
134. conduct more comprehensive studies to evaluate the effectiveness and efficiency of our approach.
135. The aim of future work will be to discover the capabilities of our approach, identifying the types of software and tests it is most effective for.
136. We will extend our approach for use with images that have a more complex structure, such as those which involving several distinct parts.
137. As future work, we are planning to develop a ranking mechanism in order to help developers prioritize the mixin opportunities based on their expected benefit on maintainability.
138. plan to evaluate our recommendation system with actual web developers and find ways to further eliminate recommendations that are irrelevant to them
139. The results show that CLCMiner achieves high accuracy and point to promising future work.
140. In our future work, we plan to investigate whether the combination of more attributes, together
141. In our future work, we plan to more systematically investigate how comments in code are related with clones.
142. In the future work, we will optimize our matching algorithm and analyze how the file names impact cross-language clones that may be from different projects.
143. In our future work, we plan to adapt CLCMiner to more languages and explore more attributes that can identify co-change relations and be used to detect clones and facilitate code reuse across different languages.
144. In future work, we will refine CLCMiner to handle such cases.
145. We intent to look into more sophisticated measures of energy as part of future work.
146. In a future study we plan to evaluate the approach with more applications to see how much energy can possibly be saved for applications from different categories
147. future explorations of this space, we intend to use a more fine-grained and accurate approach such as the Monsoon Power Meter
148. In general, we want to continue studying the compromise between the usage of advertisement and analytics and energy consumption more in-depth.
149. we plan to build n-gram models from multiple projects to perform cross-project bug detection, which may help us find more bugs more accurately.
150. We also plan to explore different approaches to segment sequences when building n-gram models
151. we plan to extend Bugram to C/C++ projects, and combine Bugram with rule-based approaches to detect more bugs.
152. In our future work, we will borrow from existing language learners to identify structural repetitions and factor these out into (anonymous) entities.
153. Our main future work will be using the inferred grammars for systematic fuzz testing, bypassing lexical and syntactical checks to deep into the program.
154. compare the effectiveness of our approach with the parsing approach on a reasonable-sized dataset;
155. discover newly introduced problems on several popular apps and contact developers to verify if our approach really helps them discovering problems.
156. we intent to develop a technique that can automatically highlight the important opinions on actual reviews to support developers on the reading task using our approach.
157. Use of fine-grained cochange information. Although our algorithms are granularity agnostic, we expect interesting differences in the change patterns used to mine evolutionary coupling at different levels of granularity.
158. We are implementing our system as an online service which could benefit millions of Chinese developers who want to reuse English knowledge in Stack Overflow.
159. In the future, we would like to design some techniques to support continuous upgrading Android specifics and effective analysis.
160. In addition, without the significant reduction of the efficiency, we will implement more accurate analysis techniques (e.g. symbolic execution) to eliminate a number of false positives.
161. In future work, we will invent a high precision service discovery method that can be applied into ServiceAccess.
162. intend to construct an approach that facilitates end-users to create their required services if they cannot find available services from Internet.
163. In the future, code generators to Java and Verilog will also be added, code verifiers will also be integrated for the verification of the generated code.
164. An empirical investigation on how MACKE can effectively impact the productivity of developers and security of the resulting applications is left to future work.
165. As future work, we intend to add to ProcessPAIR the capability of recommending detailed improvement actions for the identified causes of performance problems.
166. We also intend to apply ProcessPAIR for other software development processes.
167. we plan to extend this approach by including additional data in order to provide a more unified view of a developer.
168. we aim to better identify the locations of developers and to pre-process the free-text locations in order to obtain a more reliable developer to location matching.
169. We are also interested in the potential of library extraction directly from a developer’s code on GitHub repositories to gain a more accurate depiction of their technical activities.
170. We also plan to evaluate the accuracy of our skills extraction step, by comparing our extracted skills to those that can be manually identified from a developer’s GitHub profile.
171. future evaluation will mainly focus on cases where data import costs slow down the overall process significantly, such as the import of large, local Git repositories.
172. future development will focus on creating explorers for additional source systems and supporting further target databases.
173. we aim to deploy DataRover within real companies, to observes how the instant availability of holistic, interlinked software development data is used by development teams in practice.
174. we intend to use contract extraction to support visual debugging, evaluating its effectiveness more comprehensively.
175. we will use our tool for learning of complex domain-specific model editing operations
176. we will improve our web application and analyze the website traffic and user behaviors in our website to enhance the accuracy of analogical-libraries recommendation.
177. we are very interested in extending our approach to fine-grained level of analogy relationships
178. Our current version of testing endpoint does not support QoS testing and it will be our future work.
179. there are some robustness requirements on SUT for handling endpoint malfunctioning situations.
180. we plan to conduct further evaluation with the industry to validate our approach and tool.
181. plan to embed a requirement prioritization method to our tool for prioritizing the generated tests for better organization of requirements validation based on the generated test cases.
182. we intend to enhance this collaborative validation tool with better graphical annotation for making comments and function to convert the generated test to spread sheet for future use as a test plan to the testers.
183. works towards enhancing both effectiveness and efficiency in order for APR to be practically adopted in foreseeable future.
184. In the future, we plan to synergize many existing APR techniques, improve our predictive model, and adopt the advances of other fields such as test case generation and program synthesis for APR.
185. we expect that the history-driven approach, in complement with test-driven approaches, would be able to help avoid generating “plausible” fixes
186. we expect a greater success of combining syntactic and semantic repair candidates, to potentially enhance repair capability of APR
187. we would like to use the history of bug fixes to mine specifications of past patches and inform the future patch generation process in a way that generated patches are likely to conform to desired behaviours
188. Future work would be to design more useful features, e.g., features that characterize patch quality, to generalize our model for both search-based and semantics-based APR techniques.
189. As future work, we will continue working towards enhancing both effectiveness and efficiency of APR, rendering it to be adopted in practice in foreseeable future.
190. These not only include structural models such as UML class diagrams or feature models, but also behavioural models such as statecharts and Simulink models.
191. We plan to extend the prototype and publish it as an open framework, after a process of modularizing, documenting the code and providing a simple GUI.
192. we are looking for different datasets, both from public and industrial domains.
193. I also plan to realize API composition recommendation for the given task.
194. I am also planning to create an API composition recommendation system.
195. planning to do API composition recommendation work in the future.
196. In the future, some automatic or semi-automatic question classification approach will be studied and included in our work.
197. As for future work, to improve the effectiveness of our proposed approach further, we plan to improve the classification accuracy of our classifiers further
198. in the future our aim is to assess our approach on mailing lists of more projects of different nature (both industrial and open source) asking developers to validate the sentences according to the categories defined in this paper.
199. we plan to implement specific heuristics to detect paragraphs useful to re-document Java methods (classes) performing a larger study on data from different communications
200. future work should be devoted to providing developers with tools for automatic documentation that support specific tasks
201. future efforts on automatic approaches for artifacts documentation should also consider information from the database and should be tasks-oriented instead of being general purpose
202. Snark [34] is another FOL theorem prover, also supporting sorts. We plan to, as part of our future work
203. A potential future application for our work is cost and resource analysis.
204. we plan to use different interpolation techniques to generate the assumptions and to compare the verification results based on different interpolation techniques.
205. Extending CRUST’s Rust-to-C translation to support these features remains as future work.
206. We plan to extend memoization to other intermediate checking state in our future work.
207. We think that data-temporal properties can be useful in test-case generation and for bug detection. Evaluation of datatemporal properties is part of our future work.
208. It may be possible to define our notion of log completeness on top of the InvariMint framework. We leave this for future work.
209. We also plan to use contract extraction to support testing and debugging, evaluating their effectiveness for these tasks more comprehensively.
210. we will be investigating techniques such as Hyper/J [31] to support tracing and extraction for multithread Java applications.
211. We plan to extend SpecForge with other specification mining techniques, such as Synoptic [2] and Perfume [30].
212. We also plan to investigate alternative selection heuristics. Currently SpecForge applies the same constraint selection heuristic across all the constraint templates.
213. In the future we would like to incorporate the ability to evaluate correct GUI states, such as those presented in [35] as our system currently only looks at uncaught exceptions.
214. like to investigate an algorithm for finding good inputs to the GUI widgets
215. SLIMSTATE builds footprints dynamically, which involves some run time overhead.
216. Adapting SLIMSTATE to access patterns and shadow compression for objects is another avenue of future work
217. We view JaConTeBe as ongoing work and plan to collect more bugs in the future.
218. In the future, it would be important to study not only what bugs can be found by various approaches with focused test cases
219. We manually prepare a test case for each bug in JaConTeBe, but the mentioned approaches could reduce the effort for preparing such test cases when we enlarge JaConTeBe in the future.
220. we plan to enhance CONFIX by addressing its current limitations, and evaluate its bug finding capability of the approach, and (3) conduct experiments on a larger set of JavaScript applications.
221. The presentation of a series of insights gained from the study as to how the test generators could be improved to support better fault discovery in the future.
222. the idea of symbolic execution is orthogonal to the framework of GRT and could be integrated as another analysis component in GRT in the future.
223. We leave the study on the usefulness of ART as a GRT component as future work.
224. Enabling the application of GRT in more scenarios [35] is another direction of our future work
225. we intend to analyze the evolution of external attributes of mobile apps and investigate their potential correlation between mobile apps quality
226. We would use sentiment analysis [24] to track the evolution of end users feelings towards apps along with apps quality evolution.
227. We plan to (1) generalize the export step to additional platforms such as iOS and cross-platform JavaScript-based frameworks.
228. We plan to provide a graphical notation to allow users to connect several input screens drawings, which REMAUI could use to generate a single application
229. We plan to integrate REMAUI with tools that generate mobile application functionality either via keyword-based code search [44] or from high-level models
230. We plan to index a screenshot corpus by running REMAUI on it and storing REMAUI’s intermediate results.
231. we plan to perform deeper natural language analysis of query and code, aiming for achieving better understanding of query and code and more accurate search results.
232. we plan to conduct experiments and user studies involving more subjects, API methods, and programming tasks to further reduce this threat.
233. we plan to reduce the threats to external threats by investigating more queries over a much larger codebase.
234. to support more programming languages such as Java.
235. We will also work on methods for synthesizing sample usage code from the code search results.
236. we will perform a qualitative user study involving RESTful API developers to evaluate the effectiveness about the aforementioned HTTP proxy based RESTful API documentation approach.
237. we will investigate activity-centric virtual workspace and community-of-practice for better supporting integrated knowledge work
238. as a future work, the framework will allow configurations that throttle recommendations based on the frequency of requests.
239. : We use AST for our analyses. ASTbased analyses work well even under compilation errors, which is critical for novices.
240. One major pitfall of AST representations of source code is the loss of behavioral information.
241. As a future work, we plan to conduct user studies to assess the quality of recommendations in a controlled environment
242. We also plan to evaluate the framework by using it as a part of our introductory programming courses.
243. Future work will be devoted to automatic definition of concepts.
244. We leave consideration of global variables for future work.
245. In the future, we could study empirically on a larger dataset to investigate how many bugs could be fixed using our approach.
246. The future research directions for EvoChecker include extending its applicability to other modelling formalisms and verification logics by exploiting established quantitative model checkers such as UPPAAL
247. The further exploration of the EvoChecker usability and performance, and its applicability to other domains represent areas of future work.
248. we plan to extend LED to support DOM element selection from multiple DOM states as well.
249. plan to extend our approach to analyze the hidden DOM state and allow users to provide hidden DOM elements as input examples.
250. Because web applications evolve over time, the DOM element locators used today may not be prevalent in the future.
251. we plan to extend our approach to cover the CSS3 locators that require integrating additional information from the user as well as the DOM state.
252. We also plan to extend our approach to support XPath generation for multiple DOM elements.
253. we plan to extend the evaluation of our work and conduct user studies to analyze how effectively can LED assist the web developers in writing JavaScript code that interacts with DOM.
254. we plan to extend the implementation of our tool to support multiple DOM states.
255. we will perform a systematic sensitivity analysis of the cost ratio between measurement effort and prediction accuracy in our cost model.
256. future work involves using several appropriate performance metrics (e.g. Macro-averaged mean absolute error [30]) to measure the performance of our models
257. we will investigate which types of task relationships should be selected to give optimal results and how this can be done automatically for software projects.
258. A related future investigation would involve applying different weights to different task relationships and assessing their impact to the results.
259. We also plan to investigate if there are any other kinds of implicit relationships which can be inferred from the task information and the general context of a software project.
260. How it could be modified to be able to deal with partially defined functions is not immediately clear, hence can be explored in future work.
261. Future work should investigate whether this assumption would hold in practice.
262. One further future work is related to Batory’s [2] proposal of quantifying feature interactions in software systems.
263. g future work would involve using the tools we proposed so far in estimating Fourier coefficients and translating them into a formal model of detecting feature interactions
264. These constraints serve as a good starting point for specifying schedulability constraints with our ASL in future.
265. we plan to formalize our approach using the framework by Midtgaard et al. [32], which allows to derive variability-aware analyses for software product lines
266. In future work we wish to test the hypothesis that this route is applicable to more expressive formalisms.
267. Design For Future — Managed Software Evolution, and by the EU project QUANTICOL, 600708.
268. using a short period of history delays would make the scheduling much more accurate. We leave such improvements as future work.
269. Our approach takes a fixed mean delay of each hardware component and schedules events.
270. Event sequences generated by our approach can contain important timing information.
271. we would like to assess the ease of use of each tool through a proper user study
272. we identified and discussed strengths and weaknesses of the different techniques and highlighted potential venues for future research in this area
273. we will extend our empirical comparison of input generation tools for Android apps by (1) considering additional benchmarks
274. future work on testing mobile apps created using Web-based frameworks
275. While X-Checker has been highly effective, it suffers from a number of limitations that we plan to remedy in future work.
276. We plan to investigate techniques to invoke methods with random, yet meaningful arguments, which would further increase the coverage of the API during testing.
277. X-Checker has primarily focused on testing the framework libraries that provide support for the platform-independent part of cross-platform apps.
278. we have a plan to extend CLA/CLAMI models with human effort as future work.
279. Since this can be a limitation of CLA/CLAMI, we have a plan to conduct additional experiments on various defect datasets as future work.
280. we will add more mutation operators targeting features in multilingual programs and modify existing mutation operators to reduce equivalent mutants.
281. we will apply MUSEUM to an interactive debugger such as Blink [24] and/or advanced automated testing techniques
282. we will investigate effective methods to utilize MUSEUM to improve automated program repair and/or search-based program analysis for multilingual programs.
283. We do not consider the problem of determining whether the spec itself is correct (e.g., proving the soundness of the type system), though that is an interesting problem to tackle in the future.
284. to investigate the efficiency of our TCA algorithm in generating higher-strength CAs.
285. Encouraging developers to write better test names in this manner is an additional benefit of the approach, potentially simplifying future maintenance tasks
286. n. Currently, this manually generated rule system is adequate, but we plan to improve it in future work by using machine learning techniques to generalize
287. we plan to avoid this incomplete summarization by allowing the constraint solving engine to reason about RPLs of the form P : sc ∗ : s: ρ
288. when the parallelism is unsafe (and thus the constraints are unsatisfiable), the solver takes too long to fail.
289. Although they will require extensive adaptations, these optimizations could prove beneficial to our algorithm. These are all exciting future directions.
290. support of the remaining DPJ constructs, such as index-parameterized arrays and atomic blocks, as well as scoped locks (cf. KMeans)
291. techniques to improve the performance of the algorithm
292. ) better diagnosis information when the constraints are unsatisfiable, to assist programmers in correctly parallelizing their programs.
293. Future work includes automating the annotation of permission regions based on the sharing detection in JPF; automating the validation of any counter-example;
294. ; applying symbolic techniques to reason over input; and studying benchmarks that are representative of real world Habanero programs.
295. In future work we plan to extend the collection of antipatterns to cover additional problematic cases.
296. like to better understand those cases where an anti-pattern detects a potential problem that is still resolvable to see if that could provide insights.
297. , it may also be possible to derive a set of names by using regular expression matching in cases where we have a partial string literal
298. we plan to investigate techniques that can direct tests towards uses of these features, allowing us to profile their execution and get a clearer picture of which values each occurrence can take on and where these values come from.
299. we are planning to develop a pseudocode generator uses the proposed SMT framework to handle multiple statements, close to the standard setting of automatic comment generation.
300. We also plan to investigate the use of automatic pseudocode generation by experienced programmers in large software project environments.
301. we plan to conduct usability study and to measure effort saving with mppSMT as well as to study other factors in entire migration process (e.g., when to migrate and cost).
302. , we need to investigate the role of developers and automated tools in code migration.
303. An alternative would be to pre-compute τb, which would allow them to be re-used with different abstract domains. We plan to explore this approach in our future work.
304. introducing the more abstract summaries, we also managed to abandon any need for k-limiting within the access-path model, herein we see a huge profit for existing and future analyses.
305. we plan to administer a full empirical evaluation of our proposed approach and other improvements.
306. Exploring more sophisticated techniques for identification of covert connections could be a subject of possible future work.
307. we do not consider RPC communication with applications installed on the same device. We might explore that direction as part of future work.
308. Extending the analysis to cover such cases, while maintaining its scalability and precision, is another subject for possible future work.
309. . Future work could investigate whether such analyses benefit from the WTG representation.
310. In the future, it is important to generalize this work to handle more comprehensive control flow in Android applications.
311. we must discover all the strings that define an intent first, and then build up the correct flow of the program, which is needed for future analysis
312. In the future, we plan to test a number of different categories based on the inferred properties and code characteristics that our tool is currently able to extract.
313. As part of our future work, we plan to integrate our string analysis engine into BlueSeal
314. through more sophisticated program analysis techniques, e.g., data-flow analysis [21], differential symbolic execution [22] or differential static analysis [23]. We leave it for future work.
315. a natural next step is to raise the precision of the functional set computation using more sophisticated static analysis techniques.
316. Another interesting direction is to integrate CSLICER algorithm with language-aware merge tools and investigate possible tradeoffs.
317. Future work will evaluate how well Codebase Manipulation satisfies this requirement.
318. Future work will evaluate how well Codebase Manipulation satisfies this requirement.
319. . Future work will investigate how these external edits affect information retrieval.
320. Solstice detects some developer actions initiated via tools as typing actions, and therefore Codebase Manipulation records them as such.
321. We plan to explore these directions further in future research as well as improving our implementation and scaling it through parallel techniques
322. An initial run of Memoise performs standard symbolic execution as well as builds the trie on-the-fly and saves it on the disk for future re-use.
323. The core focus of our future work is to filter the results generated by REDECHECK so that only unintentional changes, such as the one presented in Section II-B.
324. future work with the sophisticated method presented in this paper will yield a useful tool that effectively aids developers and testers as they create high-quality responsive web sites.
325. we consider our evaluation a good starting point representing a realistic case. We plan to conduct further evaluations.
326. future needs might require switching to a checker providing higher performance for certain types of constraints.
327. Assessing productivity and reliability requires longitudinal studies, which we plan to conduct as part of our future work
328. In our future work we aim to perform a usability assessment of our tools – particularly the constraint DSL editor – involving industrial end users.
329. We further want to apply our approach to systems in other domains and in longitudinal studies.
330. we are currently working on a framework to support the systematic comparison of existing constraint languages and formalisms regarding their usefulness for particular monitoring scenarios.
331. In future work we plan to continue to apply this falsification-driven approach to the RCU verification, and to other critical systems-software targets
332. in future work, we plan to investigate in more sophisticated classification techniques to achieve higher levels of accuracy.
333. This result suggests that their developers should be more careful in quality control of their future releases.
334. In the future, we will mitigate this threat by investigating user reviews of commercial apps and apps in other markets.
335. In the future, we will further reduce this threat by inviting more developers for labeling.
336. In the future, we will summarize other review categories such as feature requests
337. we will propose techniques to summarize other software text data such as code comments and bug reports.
338. Our approach could be used by software developers and others involved in software evolution for analyzing user reviews and prioritizing their tasks.
339. In our future work we will be working on further applications of our tool suite to more scenarios – such as capture and compare to support SoS
340. As future work, enhancements to the DRIVER platform are currently under way, focusing on improving recommendation heuristics, gamification techniques to improve user adhesion
341. well-established algorithms depending either on the next advances of those algorithms and on the traits of the missions to be performed in the future.
342. As future work we are planning to enhance the current implementation of FLYAQ by integrating methodologies to control the mission execution at run-time
343. interesting future work concerns the ability of accounting for time and resources consumption that are extremely important in this domain.
344. We will investigate also the possibility of making communication between drones more flexible, so to enable also emerging behaviours;
345. As future directions, we plan to extend the CSeq framework with more sequentialization algorithms, support for other classes of concurrent programs (embedded programs, distributed programs) and counterexample generation modules for other backends.
346. In the future we plan to conduct a quantitative evaluation by using SpyREST to auto generate documentation for a large set of RESTful APIs.
347. a future version of the tool, we plan to introduce a User selection, where the user can choose between abstraction patterns.
348. we plan to create default models for the tool for other programming languages, including major languages such as Java, C++, and C#.
349. We also plan to develop a pseudo-code generator which outputs more abstract descriptions, handling larger structures than single lines of code, consisting of multiple statements.
350. there is still a problem due to semantic ambiguities. For example, in Python, a%b can be evaluated as “a mod b” if a is a numeral, or % notation that replaces % to b in string a if a is a string.
351. we plan to integrate our refactorings with Google ShipShape [3], which is a static program analysis platform allows custom analyzers to plug in through a common interface.
352. Methods may be marked as deprecated if they contain or expose design flaws. Such methods are usually removed from the program in the future.
353. The technique implemented by GRT is orthogonal to systematic approaches, and the combination of both kinds of techniques is an important direction of our future work.
354. We plan to add graphical tool support, and more ways to visualize model traces and their code coverage in the future, to mitigate this problem.
355. In the future, we will investigate activitycentric interapplication virtual workspace and community-ofpractice for more effectively supporting integrated knowledge work
356. Regarding future work, there are three main aspects: implement an heuristic to improve the decision of the next event to fire; refine the exploration’s stop condition
357. we are currently testing our above-described research method.
358. we will conduct our first evaluation with an industrial partner.
359. we will focus on the design principles identified as important according to the design-principle survey.
360. which is likely to continue into the foreseeable future, has motivated the need for longlived software.
361. architecture structural design decisions will retain intact in the face of future changes and operate in unpredictable and uncertain environments.
362. The architects of async constructs in C#, F#, and Scala also confirmed that our findings are useful and will influence the future evolution of these constructs.
363. To improve our work, we propose a tool to fix an advanced misuse of async/await constructs.
364. cyclomatic complexity, may lead to improved budget distribution, and we will investigate this in future research.
365. In the future we plan to use other adequacy metrics, such as DOM coverage [22], to measure the quality of a given test suite.
366. the learning-based approach can be tuned in various ways (e.g., selecting other features, changing the SVM parameters.
367. the size of the DOM subtree (region) to be checked can be increased to detect changes more effectively
368. we plan to evaluate the effectiveness of other state space exploring strategies
369. diversification of test-paths, and investigate correlations between the effectiveness of the original test suite and the generated test suite.
370. In future work, we first plan to explore techniques to handle the dynamic nature of modern web applications.
371. We plan to do a more extensive evaluation using programs from different application domains in the future.
372. With existing code search engines, the results returned are very dependent on the selection of search terms.
373. More flexibility in matching the specifications to the generated user interface lets the technique be used for exploration
374. In the future, we will try some fuzzy learning model in our classifier so as to gain higher classification precision.
375. it shows that past accuracy information on a training set does not improve future prediction accuracy in the evaluation set.
376. we are investigating orthogonal directions and it is possible to combine our approach and Santelices et al.’s approach in a future work.
377. we plan to reduce this threat to external validity further by investigating more bugs from more systems written in various programming languages.
378. we plan to create new technique selection and fusion methods that are specifically designed for fault localization.
379. we may remark that the weights could be adjusted and the algorithm run as many times as needed supporting future changes in the client’s priorities.
380. In future work we will extend PrefFinder to handle additional use cases
381. we will consider constraints between options and multiple options for a single query.
382. We will continue to work on the accuracy of our results by adding additional lexical information and will develop traceability links between the menu, preference names and code.
383. We will also include additional artifacts such as code and documentation to enrich the knowledge returned back to the user.
384. For future work, we plan to increase the accuracy and efficiency of our static analysis method.
385. We assume that inputs can bind to arbitrary values because a sound analysis of software applications should include the scenarios arising from malicious or malformed inputs.
386. We can enrich the database with such information by using more system test cases in the future.
387. Although this work focuses on automatic identification of true warnings, it would be interesting to extend the technique to categorize warnings into possible failure causes in future work.
388. we plan to further extend it by a careful evaluation of methods to adaptively choose the size of the initial population and possibly to reduce the execution time of the inner loop in the initial stages of the algorithm.
389. We also intend to extend our approach to account for architectures which are a mixture of both event-triggered and time-triggered components
390. An access to a new unconstrained pointer u allocates space so future accesses are consistent.
391. We limited our analysis to one language and leave evaluation of multiple languages to future work.
392. Another direction of future work is to identify and measure the accuracy of string operations and how it affects the overall accuracy of a solver.
393. we would be able to determine if the set of values represented by a symbolic value in EJSA is a subset of the set of values represented by a symbolic value in EZ3-str.
394. we are also employing sampling to determine amounts of data at different points in analysis jobs in order to better select between different options.
395. Our future work will aim at investigating other strategies for this purpose.
396. We believe this is especially true in scientific computing domains, which have often taken design cues from DLA in the past. We will study other domains as part of our future work.
397. , while we have not studied the pedagogical value of identifying stair steps, we see potential.
398. Exploiting the pedagogical benefits to identifying stairs and being able to enumerate the search space (or important parts of it with the LnB heuristic) is future work.
399. we plan to extend our tool to extract modifications that are performed across files.
400. We also plan to introduce new algorithms that can automatically process an edit script of GumTree to produce higher-order edit scripts (for instance to identify refactorings).
401. we consider moving to more fuzzy parsers, in order to accept not well formed file and reduce the parsing time.

**Sudhakar lines**

* + - * We suggest that future research should fur- ther explore the intricacies of development in a broader spectrum of collaboration (e.g., as an individual, between collaborators, in teams, and on the organizational level) and to consider what the fluidity and dynamism of these collaborative structures mean for the success of software projects.
      * As new technologies come to the fore and as the risks of large-scale data harvesting become more appar- ent, this is a stepping stone towards systematic understanding of privacy concerns at scale.
      * In the future, we plan to extend Aper to support more types of runtime permission bugs, such as library-induced [39] or device-specific [65] bugs.
      * Moreover, the experiment about generalization ability demonstrates that ARCLIN can extract APIs from a new library even though ARCLIN is trained from another libraries.
      * In future work, we plan to improve AST-Trans by incorporating more features of the code snippet, such as API sequence and node type, into the self-attention mechanism.
      * Our work has conveyed an important message that an IR-based approach can be competitive and worthwhile to pursue for software engineering tasks such as assertion generation, and should be seriously considered by the research community given that in recent years DL solutions have been over-popularly adopted by the research community for software engineering tasks.
      * While GitHub is the largest public code hosting platform, many other code hosting services like GitLab or BitBucket. The leakage in such platforms has not been systematically investigated. We leave this for future work.
      * In the future, we would like to expand our work to other ambiguity types, particu- larly semantic ones, that are still under-explored. Furthermore, and to more conclusively evaluate the usefulness of our current results, we plan to conduct user studies involving practicing engineers.
  + For future work, we are interested in the possibility of auto- matically generating fixing rules, from the existing patches that solve unreproducible builds. Also, an empirical study to gain deeper insights into the fixed patches is also an interesting direction. Be- sides, we would like to extend the fixing technique to more software repositories which have not considered reproducible builds practice.
    - * We will also investigate debugging techniques that can help DBMS developers investigate the root cause of a performance bug after it has been reported.
      * In addition, we provided a replication pack- age to facilitate future work to replicate our approach on different repositories and programming languag
      * In future work, we would like to provide guarantees for our approach by evaluating measures such as the residual risk [7] and reliability of a SUT after we terminate a fuzzing campaign.
      * we further provide the perspective that our results can generalise beyond empirical software engineering. Highlighting pitfalls in the analysis of big data, our work finally demonstrates that the use of bigger data sets does not automatically lead to more reliable insights.
      * 1Compared to the previous literature, thesefindings provide a comprehensive understanding and exploration of design ideas to enhance the integration between bots, humans, and social coding platforms
  + In the future, it would be interesting to combine our approach and prompt-based learn- ing [25] to further exploit the potential of generative pre-trained models on code generation tasks.
    - * It is a promising avenue of further work to extend our causal framework to determine root causes and identify feature interactions in the dynamic setting [11].
      * The empirical eval- uation on 106 realistic collaborative BPMN workflows and eight monorepos confirms that we enable efficient, safe dynamic software updating in long-running and frequently executed workflows.
      * We further propose WeDetector to detect WeBugs of three bug patterns. The evaluation on real-world Mini-Programs indicates that WeDetector is effective. We hope our study can inspire more researchers and practitioners to combat Mini-Program bugs.
      * In the future, we plan to extend CLEAR to other tasks such as third-party API recommendation, Linux command search, code snippet search, and program patch search
      * In future work, we will further explore the following two dimen- sions: (1) as shown in Figure10, statistical results on large-scale data sets show that most code snippets have no more than 20 lines. Within this range, TranCS is robust and stable. Constructing repre- sentations of long code snippets is still an open problem, and we leave it to future work.
      * (2) LSTM encoder is just a component of TranCS, which can be easily replaced with more advanced (includ- ing pre-trained) models in [6, 39]. We will explore more advanced models in future work.
      * In the future, we plan to incorporate more domain specific knowl- edge on aspects of training and evaluating a training ML models. For instance, one can limit the context fed to the model based on the programming language to better incorporate related information of functions and nested scopes in a piece of code. We also plan to further investigate statement completion, including better metrics for its evaluation.
      * We hope that our open-source release of Confetti and its CI workflow will help to support the growing community of practitioners and researchers engaged in fuzzing JVM-based software [51, 52].
      * In future work, we will acquire knowledge for more Python packages and improve the coverage. We also plan to add the depre- cation information of modules and APIs into the KGs, and use it to further infer appropriate versions. Besides, we will extend the dependency inference to the entire project instead of single-file code and consider the compatibility with local dependencies, which is more general in practice. Finally, we will apply our approach to other languages with transitive dependencies such as Node.js.
      * In the future, we will investigate our method in more languages and other software engi- neering tasks.
      * In the future, We conduct extensive experiments to evaluate the efficiency and effectiveness of our approach. Our tool significantly outperforms the state-of-the-art termination analysis tools.
      * Furthermore, enabling reusability and replaceability reduces CO2e emission significantly. For this work, we omit the heterogeneous inputs (the input size for modules are not the same) while reusing and replacing modules,
      * for the future to study how an interface could be built around the modules to take different types of inputs. Also, we will also look into how decomposing model into modules help to debug, local- ize, and fix bugs [40, 41]. Another direction could be looking into other properties of the DL model e.g., fairness [5, 6], in light of the modularity.
      * In future work, we will investigate the decomposition of further verification approaches as well as explore the options for paral- lelization.
      * We plan to extend DeepAnalyze to solve these challenges with the overarching goal of simplifying debugging in the large.
      * 1.In future future work we Design and develop new algorithms to improve the efficiency of the software that we are using
      * 2.Create new tools and technologies to enhance software development in future
      * Future work to Improve the security of the software to prevent cyber-attacks
      * Develop new features and functionalities to improve the user experience in future
      * Future scope to optimize the software for performance and scalability.
      * Implement better data analytics to gain insights into user behavior and preferences.
      * Integrate machine learning and artificial intelligence to automate tasks and improve the software in future
      * Implement better error handling and reporting mechanisms to improve debugging.
      * Develop better testing methodologies to ensure quality and reliability of the software in future

mplement better version control systems to improve software maintenance and collaboration

* + - * In future Improve the software architecture to make it more modular and scalable.
      * In future Integrate with third-party APIs to enhance the functionality of the software.
      * In this project for the future workDevelop better documentation to improve software maintenance andcollaboration.
      * Wetry to Implement better accessibility features to make the software usable for people with disabilities in fuure that is accessable.
      * I future to Integrate with other software systems to enhance the overall ecosystem in the project.
      * We recommend to develop better user interfaces to improve the user experience in this experiment
      * For the future experiment Implement better caching mechanisms to improve performance and reduce server load.
      * For the upcoming work Implement better resource management to reduce costs and improve efficiency.
      * In the future work develop better deployment methodologies to improve the efficiency of software releases.
      * Integrate this project with blockchain technologies to enhance security and reliability in future
      * Develop new techniques for software optimization to reduce resource consumption.
      * Develop new techniques for software visualization to enhance understanding and communication in future
      * Integrate with natural language processing technologies to enhance the user experience
      * In future implement new techniques for software testing automation to reduce costs and improve efficiency.
      * Develop new techniques for software debugging to improve efficiency and reduce time to market in future.
      * Implement new techniques for software reverse engineering to enhance understanding and maintenance infuture
      * Fr future reference better monitoring and logging mechanisms to improve software performance and eliability in future
      * Integrate with cloud technologies to enhance scalability and reduce costs and time in future
      * Implement better concurrency and synchronization mechanisms to improve software performance.
      * Develop new techniques for software documentation to improve understanding and maintainability.
      * Develop new techniques for software verification and validation to improve reliability.
      * Integrate with data privacy and security technologies to enhance compliance.
      * Develop new techniques for software project management to improve efficiency and reduce risks.
      * Develop new techniques for software maintenance and evolution to improve longevity.
      * Integrate with virtual and augmented reality technologies to enhance the user experience.
      * Develop new techniques for software education and training to improve skills and knowledge.
      * Implement better error recovery mechanisms to improve software reliability.
      * Develop new techniques for software metrics to measure and improve software quality.
      * Integrate with social media platforms to enhance user engagement and communication.
      * Develop new techniques for software debugging in distributed systems.
      * Implement better load balancing mechanisms to improve software performance.
      * Develop new techniques for software requirement elicitation and management.
      * Integrate with microservices architectures to improve scalability and modularity.
      * Develop new techniques for software architecture evaluation to improve design.
      * Implement better fault-tolerance mechanisms to improve software reliability.
      * Develop new techniques for software testing in distributed systems.
      * Implement better data processing and storage mechanisms to improve software performance.
      * Integrate with edge computing technologies to improve latency and reduce network traffic in future
      * Develop software design evaluation to improve modularity and maintainability.
      * To increase the coherence score Implement better resource allocation mechanisms to improve softwareperformance in future
      * For betterr results Develop new techniques for software requirement prioritization to improve efficiency infuture
      * Integrate with big data technologies to enhance data analytics and processing. In future for better results
      * The prototype has helped us find 13 zero-day bugs in the USB stack shipped with the vendor SDKs, and eight CVEs have been allocated.
      * Furthermore, it can greatly improve the fuzzing performance.
      * To mitigate this threat partially, we included CWEs which are not time dependent as an additional data source.
      * We find that there are four major cate- gories for each of these three branches, i.e., Code, Program, System, and Network, and conduct a systematic literature review to verify to what extent previous studies corroborate with our findings
      * The limitations discussed in the previous section prompt several potentially fruitful avenues for future work. For example, although our participants have experience with standardized practices of software development and create products for international clients, our findings are context-dependent.
      * Further- more, the proposed theory can be transformed into a nomological network—a set of causal propositions relating constructs that can be operationalized using metrics and scales [33]—to support broad val- idation via questionnaire survey.
      * An important direction for future work would be to perform an extensive evalua- tion of the existing test set generators by estimating the maximum achievable 𝑡 -wise coverage and exploring the possibility to improve them based on the evaluation results.
      * In future work, we intend to use more subject programs and replace CVAE by more powerful generative models.
      * For future work, we will extend our algorithms to multivariate metric time series. We will also try to provide more detailed in- formation about failures by exploring the correlations among the metric patterns.
      * Our findings preliminarily open the “black box” of DL SCs, putting forward practical insights and allowing for multiple paths of future research.
      * We have identified several future work directions. First, we would like to extend our approach to support additional model types, failure symptoms, and automatic repair.
      * In future, we would like to conduct studies on how to improve DNN bug repair on non- functional bugs such as fairness bugs [11, 12]. Third, we would like to extend our approach to support additional types of bugs in different stages of the ML pipeline
      * In future, we would like to explore how to leverage our findings to improve the performance of AutoML models [35].
      * we can enable interesting future research and benefit the three domains. For example, SE researchers can use our findings to design program analyses and tools targeting numerically unstable computations.
      * We believe that this observation can be further explored in the future to de- velop an even more robust translation mechanism and thus further strengthen requirements engineering methodologies.
      * Our future work will ex- tend DeepTraLog to support more different kinds of anomalies of microservice systems such as response time anomalies, and on the other hand evaluate DeepTraLog with different kinds of microser- vice systems.
      * A series of exciting and valuable findings are obtained, which provides new knowledge to the research community and can help researchers improve Android APIs’ design.
      * Finally, we reveal many unexpected observations and point to new research directions that will improve kernel fuzzing.
      * Our study indicates opportunities and challenges for future work. It highlights the need for community effort to build a large and reliable benchmark and to compare newly proposed approaches with strawman baselines. A replication package is provided at
      * https://github.com/soarsmu/SA\_retrospective
      * We there- fore improve over the performance of the current state of the art, notably TriggerScope, which yields significantly more false posi- tives, while detecting less logic bombs.
      * Our findings strongly suggest that using diversity for improving automated formal verification is fruitful and warrants further research
      * As future work, we plan to grow our catalog of anti-patterns, and refine the existing anti-patterns to exclude corner cases where successful refactoring is unlikely.
      * We suggest the following future work directions
      * First, we con- sider developing means to deal with the case of an unrealizable dynamic update, i.e., where switching cannot be forced.7 Currently, our technique is limited to detecting unrealizable dynamic updates, but not to further deal with them. When the system cannot force a strategy update, it is still possible that an update can be done with the cooperation of the environment. That is, there may exist a sequence of inputs by the environment that enables a strategy switch. This calls for synthesizing a cooperative bridge-controller.
      * Second, we hope to complete the case study of deploying our implementation in Nokia Bell Labs system and report on our expe- rience in a follow-up paper.
      * In the future, the rules we describe could be combined to detect bugs in more complex API interactions within DL libraries.
      * As part of future work, we plan to further investigate and explain the safety violations detected by our approach; for example, we can derive association rules between the test scenario attribute values and the resulting safety violations using association rule mining algorithms [15]. We also plan to increase the number of case studies by using high-fidelity simulators in the domain of autonomous drones, such as PEDRA [3] and AirSim [32], to increase the generalizability of our results.
      * In our opinion, the real excitement of Eflect is the bridge role it plays for future research: enables existing application-level energy- aware solutions to embrace platforms such as data centers and cloud servers with little additional effort.
      * The evaluation results demonstrate that PreFuzz can significantly out- perform Neuzz and MTFuzz in terms of edge coverage. Furthermore, our results also reveal various findings/guidelines for advancing future fuzzing research.
      * We believe that these two approaches highlight the usefulness and generality of our semantics, which will also enable further ap- plications. In the future, we aim to explore different test generation methods and AI model synthesis.
      * There are multiple exciting future directions. For example, extending our methodology to support deep learning frameworks is an interesting future work.
      * A possible future research direction could be to evaluate Query- Max and the other baseline techniques with other client analyses such as taint analysis or type-state analysis. Additionally, one could also extend the approach to the Android platform, with the help of frameworks such as WALA that support Android analysis. Finally, a third direction could be to study how the QueryMax approach translates to benchmarks in other popular languages such as C/C++ and Javascript.
      * We further analyze the effectiveness of each component in FIRA by an ablation study and case analysis. The results further confirm that major components both positively contribute to the effectiveness of FIRA.
      * we further perform a human study to evaluate the quality of generated commit messages from the perspective of developers, which consistently shows the effectiveness of FIRA.
      * With FlakiMe as a framework to conduct controlled and fine-grained experiments, researchers can further analyze mit- igation techniques to improve the resilience of software testing techniques under flakiness.
      * Future work might investigate further the accuracy with regard to the distribution of online snippets: an aspect that, due to the strict design of a BF method, at this stage was not achievable.
      * We will combine TaskKG4Q with existing automatic Q&A systems (e.g., AnswerBot or APIBot).
      * We will investigate semantic grouping that aims to recognize both interaction and content semantics of perceptual groups.
      * With RoboFuzz, we attempt to excite the faults in robotic systems by injecting valid (i.e., correctly formatted) messages on the ROS level.
      * Future studies can evaluate whether the effect of warnings’ position influences developers. This effect might be particularly significant.
      * We will evaluate AutoPruner on call graphs constructed using additional static analysis tools and for other programming languages, as well as assess the impact on call graph pruning on other client analyses.
      * We will continue to enrich the dataset and dedicated on proposing better Review2Code approach.
      * It would be interesting to investigate the results of the combination of our approach with TIE.
      * Our future work will focus on the following topics: Fuzzer integration, Testing strategies, Constraint synthesis.
      * Future research can investigate whether developers can improve their understanding of the security impact of the code changes through the use of code change visualization tools.
      * To study the vulnerability proneness of single-language projects-prior works examined single-language code’s proneness to defects in general but did not particularly investigate the proneness to vulnerabilities.
      * We plan to improve our approach by taking into account other JIT optimizations and CPU-level optimizations that also introduce timing side-channels in practice.
      * We may consider different graph representations of programs.
      * We will devise automated program repair techniques for text accessibility issues.
      * Future work will further explore whether AUGER can learn knowledge of projects in different programming languages.
      * Investigating other possible impacts of company domination on the development of OSS ecosystems is also an avenue for future work.
      * We will further investigate the cost-effectiveness of Tscope in real-world practice.
      * In the future, in addition to UQ, we will investigate other importance sampling mechanisms to further enhance the effectiveness of transfer learning.
      * We aim to improve both precision and recall by completing our dataset and investigating larger time windows
      * We want to further explore patterns of interdependencies among feature flag clusters.
      * Ideas for future development include Fixie as a ‘developer’s assistant’ and an IDE-based system.
      * Will work on empirical validation, automated tools for measurement, and best practices for implementation.
      * Future works focus on empirical validation on the claims of PTNNs in model zoos to check for the existence of false advertising.
      * Future research directions, including identifying and analyzing failure causal chains, standardizing the conduct of failure studies, and tool support for faster defect analysis.
      * We will empirically demonstrate PolyFax’s merits in efficiency and effectiveness against real-world open-source projects on GitHub.
      * We plan to do a user study to assess how useful is TAPHSIR in practice.
      * We plan to conduct a user study to assess how useful COREQQA is in practice.
      * SFLKit opens the door for uniform and comparable future research in the area of statistical fault localization.
      * We plan to utilize WikiDoMiner for addressing new analytical problems beyond ambiguity analysis.
      * We will enhance RegMiner by supporting manual correction of recommended critical changes and providing relevant APIs from RegMiner.
  + future work includes an empirical study to investigate the cost and benefit of large-scale mutation testing in Simulink models.
* We seek to study the expression of negative sentiment in SATD and how developers react to bot actions on GitHub.
  + - We plan to build the rest of the framework, integrate it into IDE code inspections, and test it with a number of both positive and negative examples.
  + For future work, we plan to consolidate the results of thissystematic mapping and providing an in-depth classificationof the available game approaches and support for softwareengineering education knowledge areas
  + .In the future, we will first improve our model for achieving better quality by taking the app metadata into the consideration. Second, we will also try to test the quality of existing labels by checking if the description is concise and informative
* In future, an empirical study will be conducted to validate these challenges identified through literature as well as to investigate new challenges at different levels
* The current findings provide a useful starting point for future investigations, and future efforts at developing a deeper understandingof cognitive biases will help developers and researchers to implement more effective preventive practices, and guide tool builders in creating curated support.
* our future work involves applying it to many more papers and asking other researchers to apply it. We hope that our work will lead to the increased adoption of design science in software engineering and that the visual abstract template will assist in more rapid communication of research contributions in our community
  + - We therefore hope this annotated review help researchers find and
    - select the most appropriate guidelines. By using the proper
    - guides, studies can achieve better quality thus providing the
    - SE community with more meaningful evidence in future
    - In our future work we plan to use the presented taxonomy as a guidance to improve DL systems testing and as a source for the denition of novel mutation operators.
    - In future The attendees can use such tools to explore their data and integrate data mining techniques in their research and day-to-day work.
    - As future work, to reduce the threat to external validity, we will integrate of CC2Vec into other tools and experiments on other task involving software patches
    - As a next step, we will conduct the non-academic counterpart of this study, focusing on the state of the practice and practitioners’ needs so as to identify relevant gaps and opportunities between academia and industry
    - In future A deep investigation and experimentation of new approaches to aggregate holistic data is needed to understand which solutions achieve better performance.
    - For future work, we aim to reproduce bugs we found in this study.
    - Achieving this reproduction requires overcoming numerous research
    - challenges including understanding and specifying driving scenarios in a simulator, and controlling it carefully
    - our empirical evidence indicates that a key to rein in NFRs is to leverage CSE practices, such as the quick feedback loop or the capability to offload NFRs to third-parties in future
    - The practices and research implications we presented in this paper represent useful avenues for future research
    - In future ref,Simple text classification performs much better
    - on the clean datasets than the noisy datasets. (3) With the
    - clean datasets, simple text classification outperforms the baseline approaches proposed by Peters et al. and Shu et
    - In future work, we aim to develop a theory of CI which
    - captures the practices and project context factors as the main
    - constructs, and the benefits and challenges those practices
  + may lead to as additional constructs.
    - Understanding the trade-offs among practices in particular contexts is another future research avenue.
    - As future work, we intend to expand the analysis of
    - the projects (considering a wider time window) to better
    - understand the impact of a year-long pandemic on software development.
    - As future work, we plan to leverage the general-purpose
    - metrics to extend our approach to configuration orchestration languages, starting from
    - TOSCA [52], a YAMLbased OASIS standard for defining infrastructure topologies.
    - In future, We hope that this survey will help researchers and developers in future to have a holistic perception on
    - optimization approaches for mobile devices,the impact of these approaches, and the significance of different performance characteristics
    - We end our study by providing several guidancefor future studies, e.g., future research is needed to investigate the impact of advanced feature interaction removal methods on computed feature importance ranks of different CS methods
    - Our study paves the way for future work in this area.First, our results show that NLUs tend to perform well whenthey are trained on more examples
    - Two other areas of future work is to study such
    - organizational metrics in the context of global software
    - development [13], and collaborate with cognitive psychologists
    - and organizational behavior researchers to look at social and
    - cognitive aspects of our work by doing observational studies of
    - engineers.
    - Furthermore,
    - observing developers at the workplace also opens opportunities
    - to build more sophisticated classifiers, which are able to identify in future
    - As future work, we plan to investigate the characteristics of
    - conflicts in different domains to determine if the application
  + context can have any impact on merge conflicts
    - So in future work, if we are to move
    - beyond simple assertions that the context is important, we need to
    - articulate more clearly how contextual influences operate.
    - Future work can make V2S applicable to different software
    - maintenance tasks, such as: (i) producing scripts with coordinateagnostic actions, (ii) generating natural language user scenarios, (iii)
    - improving user experience via behavior analysis, (iv) facilitating
    - additional maintenance tasks via GUI-based information, etc
    - We plan to further increase the number of respondents of our survey, so we can perform even more fine-grained
    - analyses. We may also add more questions, for example to better
    - measure the effects of practices related to AutoML, a relatively new
    - direction that is receiving sharply increasing attention in academia
    - and industry.
      * Our finding suggests that what developers claim as “fixes” for
      * flaky tests in bug reports, commit messages, etc. can be unreliable,
      * and future work should be more cautious when basing their results
      * on changes that developers claim to be “fixes”
      * We plan to adapt Comet to
      * new information sources, investigate tailoring Comet’s analysis to
      * infer security-related links, and further deploy the Comet plugin
      * with our industrial collaborators for feedback in future
      * we plan to prepare a white paper that could serve
      * to raise the discussions on a MBEBOK to wider forums and other
      * communities (SLE, general SE), and eventually lead to a concrete
      * and agreed proposal for the contents of a MBEBOK
      * As future work, we will complement the current analysis with a systematic
      * DESMET [44] evaluation over technical features, and a usability
      * study.
      * In the future, it is worthwhile to investigate how to leverage GitHub and collaborative bug finding
      * for teaching other software engineering courses or principles (e.g., test-driven development)
      * Additional evaluations are required to draw further
      * conclusions and are part of future work, which also comprises
      * comparing the hybrid approach to BS and FS
      * In the future, we envision a distributed modeling environment in
      * which multiple stakeholders of different types (e.g., developers, end users, sales) edit (part of) models, possibly in realtime.
      * ComboDroid sheds light on a new research direction for obtaining high-quality test inputs, either fully automatic or with human
      * aid. Based on this proof-of-concept prototype, a diverse range of
      * technologies can be applied in the future enhancement of ComboDroid.
      * in future we plan to investigate whether the decreasing ratios of junior researchers are a naturally occurring phenomenon (e.g., obtaining the expertise to conduct high-quality
      * research) or caused by artificial barriers
      * We suggest, as a future work, that empirical studies should be performed with sufficient rigor to enhance the body of evidence in RE within ASD.
      * These results have implications
      * for future applications of behavioral code clones, such as enabling
      * robust language migration tools or mastery of a new programming
      * language once one is known
  + In the future, we can use the gradient of a DNN with respect to the input to efficiently solve optimization problems using gradient ascent.
  + Investigating how agile methodologies impact software development and evaluating the effectiveness of automated testing in improving software quality are potential areas of future research.
  + Future research could analyze the relationship between code quality and software maintainability to improve results.
  + A possible future study could investigate the effectiveness of pair programming in increasing software development productivity.
  + Investigating the impact of code review on software quality could lead to better results in the future.
  + Evaluating the effectiveness of software metrics in predicting software quality is a potential area of future research.
  + Future research could analyze the impact of software design patterns on software quality and the maintainability of code.
  + Investigating the effectiveness of static code analysis tools in detecting and preventing software defects and reducing errors in code is a possible area of future research.
  + Analyzing the impact of software process maturity on software quality and productivity is a potential area of future research.
  + Evaluating the effectiveness of test-driven development in improving software quality is a potential area of future research.
  + Studying the relationship between software complexity and software quality could lead to better results in future research.
  + Investigating the effectiveness of refactoring in improving software maintainability could lead to better results without errors in future research.
  + Analyzing the impact of software architecture on software quality and maintainability is a potential area of future research.
  + Studying the effectiveness of code coverage analysis in improving software testing could lead to better results in the future.
  + Investigating the impact of software engineering education on software quality and productivity is a potential area of future research.
  + Evaluating the effectiveness of code documentation in improving software maintainability could improve data quality in the future.
  + Analyzing the relationship between software development team size and software productivity could lead to better results in the future.
  + Studying the impact of software testing techniques on software quality and productivity could lead to better results in future research.
  + Investigating the effectiveness of software process improvement models in improving software quality and productivity is a potential area of future research.
  + Analyzing the relationship between software development methodology and software quality is a potential area of future research.
  + Studying the effectiveness of software traceability in improving software maintenance and evolution could lead to better results in the future.
  + Investigating the impact of software reuse on software quality and productivity is a potential area of future research.
  + Evaluating the effectiveness of continuous integration and delivery in improving software quality and productivity is a potential area of future research.
  + Analyzing the relationship between software testing coverage and software quality in the future is a potential area of research.
  + Studying the impact of software engineering tools and environments on software quality and productivity is a potential area of future research.
  + Investigating the effectiveness of software quality assurance and testing standards in improving software quality is a potential area of future research.
  + Analyzing the relationship between software performance and software quality is a potential area of future research.
  + Studying the effectiveness of software fault prediction models in identifying and preventing software defects is a potential area of future research.
  + Investigating the impact of software design quality on software maintainability is a potential area of future research.
  + Evaluating the effectiveness of software project management techniques in improving software productivity and quality is a potential area of future research.
  + Analyzing the relationship between software documentation and software quality is apotential area of future research.
  + Studying the impact of software development processes on software quality and productivity is a potential area of future research.
  + Investigating the effectiveness of software testing automation in improving software quality and productivity is a potential area of future research
  + 34.Future research will explore how software engineering tools and environments affect software quality and productivity.
  + 35.The effectiveness of software quality assurance and testing standards in enhancing software quality will be investigated in upcoming studies.
  + 36.The connection between software performance and software quality will be analyzed in the future.
  + 37.Future studies will assess the effectiveness of software fault prediction models in detecting and preventing software defects.
  + 38.The impact of software design quality on software maintainability will be explored in future research.
  + 39.The effectiveness of software project management techniques in improving software productivity and quality will be evaluated in upcoming studies.
  + 40.The relationship between software documentation and software quality will be analyzed in the future.
  + 41.The impact of software development processes on software quality and productivity will be studied in upcoming research.
  + 42.The effectiveness of software testing automation in enhancing software quality and productivity will be investigated in the future.
  + 43.The connection between software testing efforts and software quality will be analyzed in upcoming studies.
  + 44.The effectiveness of software fault localization techniques in detecting and repairing software defects will be studied in the future.
  + 45.The impact of software engineering education on software development team performance will be investigated to improve future work.
  + 46.Future studies will evaluate the effectiveness of software quality control and assurance processes in improving software quality.
  + 47.The relationship between software development team structure and software productivity will be analyzed in future research.
  + 48.The impact of software maintenance practices on software quality and productivity will be studied in upcoming research.
  + 49.The effectiveness of software risk management techniques in improving software quality and productivity will be investigated in future work.
  + 50.The relationship between software testing tools and software quality will be analyzed in future research.
  + 51.Future studies will explore the impact of software development team communication on software productivity.
  + 52.The effectiveness of software process improvement programs in enhancing software quality and productivity will be investigated in the future.
  + 53.Future research will analyze the relationship between software development team morale and software productivity.
  + 54.The impact of software project complexity on software quality and productivity will be studied in future research.
  + 55.The effectiveness of software defect prediction models in detecting and preventing software defects will be investigated in the future.
  + 56.The relationship between software testing automation tools and software quality will be analyzed in future research.
  + 57.The impact of software development team diversity on software productivity will be studied in future research.
    - * As future work, we plan to empirically assess the impact
      * of external quality metrics (e.g., testability and readability)
      * as documented by developers in their commit messages on
      * quality and compare and contrast them with the findings for
      * the internal ones
      * In future, we plan to further improve the detection
      * capability of Watchman and generalize our technique to other
      * Python library ecosystems such as Anaconda to make it accessible
      * to more developer communities.
      * As a consequence of this study, we are currently working on a
      * reusable framework to guide educators in adopting principles of
      * PBL for the introduction of practical assignments in SE education.
      * As with any research at this stage, there is much left to do. We
      * will investigate in more detail how structural information can be
      * best integrated with the lexical information in order to generate
      * better summaries in future.
      * For future work, we plan to use the results of this study to
      * design static and dynamic analysis tools that would simplify
      * how programmers write reliable JavaScript code.
      * we suggest using concepts of social translucence and awareness to
      * open up software components in a way supportive of
      * collaborative software development
      * future work will address the merging of document versions based on 3-way merging. Virtually all advantages of
      * our approach are carried forward to the problem of merging, notably the number of merge conflicts will be reduced.
      * Challenges for the future of value-based monitoring and control include:
      * Integrating advanced financial instruments such as real options [3] into the planning and control of software-intensive
      * systems [24].
      * Integrating the explicit information in plan-driven feedback
      * control methods with the tacit information used in agile methods [13] to support both agility and discipline in controlling
      * software projects [5].
      * Integrating project-level and organization-level feedback control of software-intensive projects, programs, portfolios, and
      * organizations, using such techniques as the Experience Factory [4], Balanced Scorecards [15], the Benefits Realization
      * Approach [22], and the CeBASE Method [8].
      * Integrating value- based methods into the full range of disciplines involved in software engineering: requirements, design,
      * development, test, COTS integration, planning and control.
      * In future work, we plan on making the summarization
      * interface more interactive, using more effective visualizations of feedback distributions.
      * In future, We propose that clone genealogy information can be used to
      * identify clones that may benefit from new approaches to
      * clone management.
      * The next step for this work is to replicate our feedback surveys in
      * other companies based in various countries. This is important to get
      * a broader view on software engineering.
      * Perhaps, in the future Bots can be used to study the
      * impact of Bots and then self adapt to avoid negative consequences.
      * But for the future work.I suggest that it be approached
      * through organized discussions and dialogs, perhaps pursued
      * through the vehicle of a special series of seminars or meetings.
      * In future With more support for retrospection on productivity,
      * developers may be able to better share best practices, such
      * as goal setting best practices discussed by participants in
      * our observational study.
      * In future necessary next steps are (1) to replicate our survey in
      * other regions of the world to obtain a more global picture about
      * what soware engineers care about, and
      * (2) to understand the processes driving soware engineering research that sometimes seems
      * away from the claimed need of the soware industry
      * For future work, we plan to consolidate the results of this
      * systematic mapping and providing an in-depth classification
      * of the available game approaches and support for software
      * engineering education knowledge areas
      * In future we plan to we plan to examine the effects of the GUI structure on its testability.
      * .Continued exploration of the field is planned for the 2nd SEMAT
      * Workshop on a General Theory of Software Engineering (GTSE 2013),
      * held in conjunction with ICSE 2013 in San Francisco on May 26, 2013.
      * This event will feature full ten-page research papers as well as fourpage position papers.
      * We plan to use ARdoc as a preprocessing support for summarization techniques in order to generate summaries of app
      * reviews in future.
      * future works and discussions
      * Page replacement is the process of evicting pages that are currently resident in memory in order to make room for new pages.
      * A page replacement strategy is invoked any time the system
      * is low on available memory
      * high-level context important for page replacement can be explicitly set using madvise. This allows
      * eviction to use the pattern of access as part of the selection
      * criteria for removal
      * Scheduling involves sharing the processor between all active
      * processes. A scheduling policy tries to ensure that all currently
      * executing processes make progress
      * The path-specific customizations we are exploring with regard
      * to scheduling are related to high-level process state and lowlevel blocking I/O requests
      * Layer violations in this case stem
      * from the need to reconcile cross-layer information such as access patterns, process priority, and disk requests in order to
      * make good scheduling decisions.
      * futher Requirements tracing is important to keep track of the interdependencies between requirements and other
      * artifacts and to support project teams and software maintenance personnel in several tasks in future.
      * The ultimate challenge will be to find reliable
      * ways to evaluate the cost and benefits of design sustainability, so as to build a body of evidence that can be used to
      * demonstrate.
      * Going forward we hope to use the tool to support
      * the developer reuse examples instead of just using them to
      * highlight how a framework is to be used.
      * In the future, we will explore the applications of this model
      * to these problems. We will also investigate the synthesis of
      * sample code from the generated API sequences.
      * We further found that the quality of the change description significantly affects the efforts of later change understanding.
      * Engineers should be aware of the change description’s
      * importance and be responsible for providing informative descriptions upon check-in in future.
      * Further work is also required
      * towards systematic techniques for generating better designs from
      * the insights gained in the evaluation of the current alternatives
      * a future application data entry can be labeled (manually or automatically); however, using such labels
      * as the test oracle is not feasible. The reason is that there exists some inaccuracy
      * (i.e., predicting a wrong label) in the learned classification model.
      * The case study suggests some directions for future work, both in
      * socio-technical congruence, and in aspect-mining. The dynamics of these subcommunities such as turnover rate and
      * migration are topics that we plan to investigate as well.
      * FUTURE WORK
      * Developer networks provide a promising foundation for several
      * novel metrics to be introduced into failure prediction models.
      * In future work, we hope to explore a more sophisticated analysis of a
      * project’s developer network and how developer information can
      * be applied to products at the file level for failure prediction to
      * improve our model.
      * Examining connection weighting schemes,
      * network robustness, clustering, and the evolution of developer
      * networks over time are among the many possible areas to explore.
      * Future Work
      * The study and promulgation of assertions remains an
      * active endeavor for researchers and practitioners
      * alike.
      * One of the most promising recent
      * developments in research with assertions is the
      * automatic discovery of likely program
      * invariants , and the related technique of
      * correlating failure data with execution history data
      * from field installations of software systems to help
      * While the automated reporting of failure data from the field is
      * already a staple of many commercial software
      * systems, albeit in rudimentary form (as evidenced by
      * periodic requests to the user for consent to report data
      * from Microsoft Windows XP, Apple Mac OS X, and
      * other systems)
      * it remains to be seen how well some
      * of the more sophisticated approaches will be able to
      * scale for sufficient impact on development practice
      * Such information might include
      * anecdotes about ad hoc use of assertions by
      * individual developers, systematic use of assertions in
      * large-scale development projects
      * Much of this
      * information is difficult to come by since it is to be
      * found primarily in the undocumented lore of software
      * development practice rather than well-documented in
      * the research literature
      * It is hoped that the future joint
      * NUS/SES projects can bridge the gap between XVCL as a mere
      * powerful language and its integration into the best practices of
      * industrial software engineering. In particular, it is planed to apply
      * reuse techniques in more application domains, formulate
      * systematic methods based on project
      * In future The attendees can
      * use such tools to explore their data and integrate data mining techniques in their research and day-to-day work.
      * But for the future work.I suggest that it be approached
      * through organized discussions and dialogs, perhaps pursued
      * through the vehicle of a special series of seminars or meetings.
      * Further work is also required
      * towards Integrating value- based methods into the full range of disciplines.
      * In future With more support for retrospection on productivity,
      * developers may be able to better share best practices
      * As future work, we plan to investigate the characteristics of
      * conflicts in different domains to determine if the application
      * context can have any impact on merge conflicts
      * In future necessary next steps are high-level context important for page replacement can be explicitly set using emperical.
      * In future by using sci-bert we can get better result and accuracy rather than bert.
      * As future work, we intend to expand the analysis of
      * the projects (considering a wider time window) to better
      * understand the impact of a year-long pandemic on software development.
      * In future, We propose that clone genealogy information can be used to
      * identify clones that may benefit from new approaches to
      * clone management.
      * But for the future work.I suggest that it be approached
      * through organized discussions and dialogs, perhaps pursued
      * through the vehicle of a special series of seminars or meetings
      * future work will address the merging of document versions based on 3-way merging. Virtually all advantages of
      * our approach are carried forward to the problem of merging, notably the number of merge conflicts will be reduced.
      * .General Theory of Software Engineering (GTSE 2013),
      * held in conjunction with ICSE 2013 in San Francisco on May 26, 2013.
      * This event will feature full ten-page research papers as well as fourpage position papers.
      * So in future work,our empirical evidence indicates that a key to rein in NFRs is to leverage CSE practices and get better
      * ac.
      * In the future, we are planning to extend IntelliCode Composecapabilities.
      * The feature extraction step extracts microservice- and trace-level features fromtrace logs.
      * In this paper, we have proposed MEPFL, an approach for latent error prediction and fault localization of microservice applications by learning from system trace logs
      * .In future work, we will explore more features (e.g., code changes,project properties), advanced machine learning techniques such as
      * deep learning to learn hidden representations from the data, andneutral networks to train our models
      * Creation of software tools for smart contract languages. Enhancement of testing and debugging in future.
      * In the future, we will further evaluate the proposed model on larger scale datasets in different programming languages and for a variety of software engineering tasks.
      * In future We will also explore other neural models to capture more deep semantics of source code.
      * The result indicates that studying refined ranking techniques for specific repair techniques is promising and calls for more studies on more different types of repairs.
      * In future work, we will investigate techniques for applying it to other types of artifacts such as source code or formatted data.
      * We encourage future work to use our findings to construct in-depth surveys that can be distributed to a much wider audience so we can get a much wider understanding.
      * As future work, we will study the impact of the only specific proposal, i.e. Carver’s guidelines [3], on the reporting of
      * replications. In particular, we will focus on how authors report
      * changes between original experiments and replications in
      * order to elaborate a proposal
      * In future the risks that programmers might face, by relying too much on their personal experience; subjective, personal recollection is notoriously error-prone
      * we would like to stress that our goal is not to judge
      * or offend any reviewers or authors. On the contrary, we highly
      * appreciate the time and effort the participants took to answer
      * our questions, which documents their interest in this issue.
      * It was our intent to
      * focus on issues in a specific community to focus improvement
      * efforts in software engineering empirical evaluation
      * In future This tool will be able to capture and explain the causes of data
      * quality problems at source and should then inform the
      * development of suitable preventive measures.
      * We still need significant empirical evidence to be part
      * of every Software Engineering research project. We need
      * to better understand the extent to which conclusions we
      * can draw about open source and proprietary software systems are the same and different.
      * .Future works include an a characterization of the available empirical methods,
      * based on the level of impact of the above mentioned challenges.
      * .In future AR should be to enact a
      * process based on a declared-in-advance methodology in such
      * a way that the process is recoverable by anyone interested in
      * subjecting the research to critical scrutiny.
      * In future we identify the internal threats to the
      * validity of our systematic literature review, traceability
      * survey, traceability model, and trace links taxonomy
      * .In future we built a proof of concept with
      * two iterations of feedback cycles to tune the extraction of user
      * stories’ topics and reported improvements.
      * We need to acknowledge that we cannot overcome all issues.
      * Several “corollaries in research” [5] are relevant to “applied” and
      * industry-sponsored/-relevant research as done in SE
      * This has led to very little primary data regarding their effectiveness,
      * and generally only speculation over their potential. An
      * empirical investigation to assess the effectiveness of SLR
      * tools could be a beneficial contribution to the topic.
      * Since many factors were similar between the
      * projects a comparative analysis was possible. The study at hand
      * is one of few studies in a large-scale agile software
      * development context.
      * For future reference , on average,
      * TCtracer is more effective at both the method level and the class
      * level than any single existing technique. This makes TCtracer the
      * most effective approach for test-to-code traceability to date.
      * Comparison between SpecuSym and state-of-theart abstract interpretation based method [78] shows that SpecuSym
      * not only successfully detects leaks in the reported programs but
      * also eliminates false positives for a more precise results in future
      * this design is able to automatically detect non-trivial and accurate taint
      * flow specifications in widely used Node.js modules, which enables
      * an existing static analyzer, LGTM, to discover many previously
      * unknown security vulnerabilities.
      * In future The source-code of the XRust compiler can be obtained through
      * https://github.com/parasol-aser/XRust. The configured docker image can be obtained through https://hub.docker.com/repository/
      * docker/geticliu/xrust-icse2020.
      * A complete evaluation of such a hybrid is part of future work. We hope that
      * this will be just one of many avenues that the present paper
      * will open for dynamic invariant detection.
      * ComboDroid sheds light on a new research direction for obtaining high-quality test inputs, either fully automatic or with human
      * aid. Based on this proof-of-concept prototype, a diverse range of
      * technologies can be applied in the future enhancement of ComboDroid
      * In that direction, we plan to investigate whether the decreasing ratios of junior researchers are a naturally occurring phenomenon (e.g., obtaining the expertise to conduct high-quality
      * research) or caused by artificial barriers for junior researchers
      * (e.g., gate-keeping, paper engineering).
      * This study constitutes a solid empirical foundation upon which
      * other researchers could build, and appropriately evaluate, program
      * repair techniques based on NMT.
      * Our demystification report links the two information
      * sources, which turns “somebody else’s unfortunate mistakes” into
      * a fortune to help developers learn the API usage directives. Our
      * evaluation confirms the accuracy of our open information extraction method, and our user study demonstrates the usefulness of our
      * demystification reports for debugging API-misuse related program
      * errors with respect to the concerned API usage directives
      * In order to address scalability and popularization of the approaches, future
      * research should be invested in tool support and in addressing combined RE traditional adoption strategies. Finally, we conclude our work stating that agile
      * methods assume that it is very hard to elicit all the requirements from the user
      * upfront, at the beginning of a development project.
      * In future Agile methods are fundamented in
      * that i) requirements are not well known at the beginning of the software development project, ii) requirements change, always do, and iii) making changes is
      * not expensive if you have a RE process well-defined.
      * These results have implications for future applications of behavioral code clones, such
      * as enabling robust language migration tools or mastery of a new
      * programming language once one is known.
      * In future side channels are prevalent and the side channel vulnerability detection techniques have to take into account the impact
      * of a program’s runtime during side channels analysis
      * We have made additional analysis, such as the comparison
      * between the subjects’ opinion for each sample. However, the
      * limitation of short papers space restricted our capacity to show
      * them.
      * In future A variety of structural metrics can represent the internal
      * quality attributes considered in this study. Based on our
      * empirical investigation, for metrics that are associated with
      * quality attributes, there are different degrees of improvement
      * and degradation of software quality.
      * Future work will be devoted to further experiment and assess the tool in larger software projects, in particular projects
      * of industrial partners interested in using ADAMS
      * In future, we plan to further improve the detection
      * capability of Watchman and generalize our technique to other
      * Python library ecosystems such as Anaconda to make it accessible
      * to more developer communities.
      * In future these practices
      * in detail, including how to identify and recognize when to build a
      * shared understanding.
      * The results can obviously be improved by finetuning the tools and algorithms based on further case studies. Nevertheless, the results should be encouraging to take
      * advantage of the current situation in model-based development where a lot of redundancy can be avoided by using this
      * kind of approach early on.
      * In future we will perform extrinsic evaluations of the
      * automatic summaries by using them to support various software
      * engineering tasks, such as: concept location, impact analysis,
      * ontology extraction, etc. Tool support will be provided and
      * integrated into IDEs. Displaying the summaries in the IDE will
      * be a research topic in itself.
      * For future work, we plan to use the results of this study to
      * design static and dynamic analysis tools that would simplify
      * how programmers write reliable JavaScript code.
      * .Future work involves investigating how developers from proprietary
      * projects use annotations, as well as an examination of the more
      * structured processes that some teams may use
      * .In future We also intend to
      * examine how tools with a closer coupling to issue tracking systems
      * (e.g., Jazz) impact how programmers perform task management.
      * In future we intend to focus on task lifespan to better
      * understand how developers use annotations for long term task
      * management. Finally, we hope to incorporate some of the tool
      * suggestions that resulted from this work within TagSEA [19], a tool
      * we developed for managing and navigating tagged annotations in
      * source code comments
      * we expect that our results will contribute to research
      * related to the effectiveness of software security. There is a growing
      * community of researchers investigating how the investment in
      * security since the early phases of software development can help
      * in reducing software vulnerabilit
      * .As we found that the majority of the development effort is spent
      * on the collaboration and communication activities, we would like to
      * explore the effect of using models on software design communication. This in order to understand whether or not the use and share
      * of software models could help in communicating and discussing
      * software architectural/design decisions.
      * This involves a return of investment analysis to determine
      * the cost-benefit tradeoff between the increase in development time
      * and the resulting improvement in software quality. Such measures
      * we hope will help project managers make meaningful decisions
      * about the utility of deploying TDD in their organization in future.
      * Using the SciDeBERTa and SciDeBERTa (CS) as a pre-trained LM (PLM) specialized in the science technology domain [19].
      * we can get the sentimental score with greater accuracy.
  + In the future, we will focus on creating automated tools to gather and analyze software requirements efficiently.
  + We aim to explore novel software development methodologies to enhance the effectiveness and efficiency of software development in the future.
  + Creating intelligent software agents to assist in software development will increase software quality in the future.
  + We plan to develop software tools for automated testing and debugging in the future.
  + Our future work will involve designing software systems that can manage and monitor software project risks.
  + We aim to develop software applications for agile project management in the future.
  + Our future research will investigate new software development processes and techniques to enhance software quality.
  + We plan to develop software tools for automated software maintenance and evolution in the future.
  + In the future, we will explore software engineering approaches for developing secure software systems.
  + We aim to create software systems for automated code review and analysis in the future.
  + Our future work will focus on developing software engineering methodologies to support software globalization.
  + We plan to investigate software engineering approaches for developing scalable and distributed software systems in the future.
  + Developing software systems for measuring and improving software engineering team performance will be a focus in the future.
  + We aim to create software engineering solutions for handling big data in software development in the future.
  + Our future research will involve developing software engineering methods for creating adaptive and autonomous software systems.
  + In the future, we will investigate software engineering techniques for improving software documentation.
  + Creating software systems to support continuous integration and deployment is part of our future plans.
  + We plan to develop software engineering methodologies for developing machine learning algorithms in the future.
  + In the future, we will work on creating software engineering solutions for developing blockchain-based systems.
  + Our future work will investigate software engineering approaches for developing software for IoT systems.
  + Developing software engineering solutions for creating virtual and augmented reality applications is part of our future plans.
  + We plan to create software systems for developing cloud-based software solutions in the future.
  + Our future research will involve investigating software engineering approaches for developing real-time and embedded software systems.
  + We aim to develop software systems for automating software engineering processes in the future.
  + Creating software engineering methodologies for developing quantum computing software is part of our future plans.
  + In the future, we will investigate software engineering approaches for developing software for autonomous vehicles.
  + Developing software systems for natural language processing applications is part of our future work.
  + We aim to create software engineering solutions for developing cybersecurity solutions in the future.
  + Our future research will investigate software engineering approaches for developing software for smart cities.
  + Developing software systems for e-commerce and online businesses is part of our future plans.
  + We plan to create software engineering solutions for developing software for educational applications in the future.
  + Our future work will involve investigating software engineering approaches for developing software for healthcare applications.
  + Developing software systems for financial and banking systems is part of our future plans.
  + We aim to create software engineering solutions for developing software for the gaming industry in the future.
  + In the future, we will investigate software engineering approaches for developing software for transportation systems.
  + Developing software systems for the hospitality industry is part of our future work.
  + We plan to create software engineering solutions for developing software for the entertainment industry in the future.
  + Our future research will involve investigating software engineering approaches for developing software for the energy industry.
  + Developing software systems for supply chain management is part of our future plans.
  + We aim to create software engineering solutions for developing software for the retail industry in the future.
  + Our future work will involve investigating software engineering approaches for developing software for the automotive industry.
  + Developing software systems for the aerospace

**Damini lines**

1. In the future, they plan to integrate our approach TechSumBot into an IDE to help developers in searching their needed information from SQA sites more efficiently and accurately.
2. For future work, they outline how and what functionalities may be needed to effectively address the current gaps and user needs whilst embracing the desired balance between automation and human control in the security patch management process.
3. In the future, they plan to continuously improve the proposed approach to address these limitations.
4. Limitations:
5. There are still some limitations in our approach. First, it currently only supports the localization of English texts.
6. Second, for LTS with reversion the acceleration does not work when a specified label text does not exist on the GUI screen.In test automation, to use that LTS version, it is better to first use faster techniques like template matching to determine whether a GUI enters some state and

then apply LTS to locate label texts.

1. Third, LTS yet does not work well for label texts on complex image backgrounds because we only use a very simple text detector. For such cases, it is better to call the original OCR engines when writing test scripts.
2. The future works to optimize the UDG construction to further decrease the overhead.
3. In the future, VirtualBuild can be further optimized by identifying the targets that demand the actual build in advance.
4. In future work, the customized SMT model is desirable to handle the direct translation from text structure to code structure.
5. They found specific bug detectors can still produce a significant number of false alarms which could hinder the adoption into practice and hence should be addressed in future work.
6. For future work toward a more privacy-compliant environment, a framework like a Single Sign- on scheme is desirable where the user is capable of revoking and modifying the previous data consent to a trusted data management entity.
7. In the future work they plan to focus on the program state initialization, support for C++ and leveraging the code information for structure-aware fuzzing.
8. In future work, they plan to incorporate the reinforcement learning techniques (e.g. policy

network) into the framework to adaptively choose the suitable deliberation processes, thereby enhancing the performance.

1. For future work they use more advanced image processing techniques, such as scene graph generation and region captioning to extract more information from the visual datasets to be compared against our automotive domain benchmarks.
2. In future, they will evaluate our method and metric on more DL tasks and MRs. We will also

strengthen our method and metric by considering the features of the other effective DL testing criteria.

1. In future, they plan to include more CI tools and support more programming languages.
2. Future work they will extend our approach to validate the effectiveness of neural networks on intractable industrial instances and design more highly confident and verifiable end-to-end

neural networks for checking LTL satisfiability.

1. In future work other programming languages downstream editing tasks, we studied in this work are using Java. Since CoditT5’s pretraining is on the dataset consisting of six programming

languages, we expect it to also perform well on editing tasks in other programming languages, but we leave empirically to verify.

1. They will explore editing the original class in place instead of creating a renamed template class as future work.
2. In the future, they planned to extend Compressor to support models of more architectures and pre-trained models of code. We also plan to evaluate Compressor using additional datasets and tasks beyond those considered in this paper.
3. In the future, they will enhance our KG construction with more robust information extraction methods and improve our test scenario generation and presentation inspired by our user study.
4. For Future they envision CrystalBLEU to provide a more precise metric for future evaluations of techniques that predict source code.
5. They planned to integrate techniques like Delta Debugging in our future work to minimize the set of crash-causing compiler flags.
6. In future they would like to explore if polarity based data augmentation strategies could improve other related tasks in software engineering, such as sentiment analysis.
7. In future work many of these existing tools detect not only blocking errors but also other kinds of concurrency errors. In principle, Goat can easily be extended to also scan for safety errors, based on the super location graphs it already produced.
8. In future work, they plan to select more distance calculation methods and machine learning algorithms to find a more suitable combination for achieving better detection performance.
9. In future work, they would like to apply different feature selection techniques to figure out the most suitable combination of these features. In this way, Amain may achieve more scalable and effective semantic code clone detection.
10. In future work, they planned to alleviate this limitation by making our procedure of state exploration incremental and on-demand, and the configuration self-adjusted instead of set by users.
11. As a part of future work, they will consider an extension of symmetries and epochs to multi-path programs.
12. For Future work we can allow generated programs to have more than 4 edges per node by

generating mazes of different shapes, such as those on a hexagonal grid. We believe extending Fuzzle to support these features is a promising future work.

1. In future work, they planned to improve CRISCO by providing some features which can guide ADS testers to define trajectory specifications more easily.
2. Future work should be invested in improving the quality of the rules that GLITCH implements.
3. In the future, they planned to consider more information to provide a more effective incident extraction and diagnosis method.
4. In future work it involves evaluating the extent to which the ideas presented here can be applied to other computing domains (e.g., iOS, Web), and expanding Groundhog’s support to additional assistive services and more complex gestures.
5. As future work, they first planned to enlarge the scope of the HyperAST to other languages by integrating more grammars from Tree-Sitter. We will revisit existing approaches discussed in section 4.4 on top of the HyperAST. Finally, we will explore the possibility to offer an engine

capable of running a temporal analysis taken as a query to be executed on the HyperAST. Thus, facilitating its usage by a wider audience.

1. his paper opens various lines for further work. Firstly, our technique is mainly focused on dealing with overfitting, the most pressing concern in automated repair. The readability of ICEBAR’s

patches is inherited from ARepair’s, and clearly calls for improvement. Improving patch

readability may be achieved either by improving ARepair, or by post-processing ICEBAR’s output, e.g., applying some syntactic simplification techniques. We plan to explore both options as

future work. Secondly, the effectiveness of ICEBAR may be affected by the “initial” test suite from which the repair is launched. Although we have not studied this issue in this paper, we plan to evaluate different strategies to generate initial test suites, and how these impact ICEBAR’s

repairability metrics.

1. They left the investigation of how to fit inline tests into different software- and test-design processes as future work.
2. In the future, they planned to generalize the technical framework of Insight to other software ecosystems and involve various programming languages in our empirical study.
3. There are several interesting avenues for future work. First, they were interested in finding more efficient methods to generate test cases based on the given specification. Furthermore, we only considered how the ego vehicle should behave in this work, and are interested in exploring how the traffic flow should be when other traffic participants are autonomous vehicles as well.
4. The observations that they made throughout this work pave the way for the following future work: extend InfraSecure to detect other security vulnerabilities, integrate the tool with

methods for automated patching, and port InfraSecure to other configuration management tools.

1. In particular, the reduction in the number of paths should be directly translated into smaller test suites, but the coverage should still be maximized (as LISSA never prunes feasible paths). This is also an interesting research direction for future work.
2. However, there are other noise handling methods in the research community that could be ported over. We will try more approaches in our future work.
3. This paper outlines directions for future work. Using module bundlers to identify production dependencies may augment current SCA tools to provide better insights on the scope of their dependencies within their project’s context and usage. Consequently, module bundlers or similar tools, may benefit far more than just client-side applications and should be part of the build process of projects that extensively rely on open-source code.
4. In future work, more experiments will be conducted on a variety of models and datasets to alleviate this threat.
5. **Internal validity**: An internal threat comes from the choice of datasets for evaluation. To mitigate this threat, in this study we use the CIFAR-10, CIFAR-100, and SVHN datasets from Pytorch,

which are well organized and widely used.

1. **Construct validity**: In this study, a threat relates to the suitability of our evaluation metrics. We use accuracy and Jaccard distance-based difference as the evaluation metrics. These metrics have also been used in other related work.
2. In the future work, they will consider more rigorous manual labeling and inspection to reduce the noise in the data as much as possible.
3. In future, we will apply and improve our approach to other classification problems in requirement engineering and software engineering.
4. The contribution of their work is to make the first step in providing a lightweight and precise analysis to identify a large part of (un)affected versions while leaving the rest of versions as the future work. Considering that there are a lot of versions to examine, we believe this step is

important and necessary.

1. In the near future, they will study methods of accurately estimating the times of thread creation to reduce the size of CDG.
2. In the future, they will explore ways to enhance and extend our approach and develop novel applications for partial code search and reuse enabled by our model’s unprecedented type

inference capability.

1. It is an interesting future work to study other abstract interpretation-based techniques to reduce the size of integer linear constraints.
2. In future, they planned to explore more on interpretable machine learning and extend our approach to other types of vulnerabilities (e.g., infinite loop vulnerability).
3. For future work, we plan to evaluate the performance of RobustTrainer in various domains of samples.
4. As future work, they identify four main directions. First, they planned to improve the symbolic runtime, making it thread-safe and adding support for floating-point operations. Second, accurate handling of symbolic memory accesses could make SymFusion more effective.
5. Future work will extend dVermin to handle inconsistencies dynamically introduced when the app is running, and more evaluation on more apps and more scale setting.
6. The paper concludes with a discussion of the related research and avenues of future work.
7. In future they wanted to investigate the way to improve our localization accuracy.
8. In future work, they aim to evaluate selection criteria to identify the most suitable CNF transformation for a given formula.
9. For future work, they will develop a notion of fault localization for these programs, develop a

theory of mutation generation, and address issues regarding flakiness present in this computing paradigm.

1. As future work, they aim to work on the main limitations of our Live Refactoring Environment. One of them is the small number of implemented and tested refactoring techniques included in our approach. So far, we have only validated identification, suggestion, and application of the Extract Method refactoring. However, they expect us to verify if the other refactoring we already implemented are correct and working as they should. We also aim to improve the live

mechanisms included in our approach to reduce the time needed to identify and suggest refactoring. Finally, we also wish to include visual methodologies suitable for colorblind people.

1. Also, as future work, they expect to improve our empirical experiment, improve the experimental groups, and test it with more participants. We hope to test the Extract Variable,

Extract Class, Introduce Parameter Object, and Move Method refactoring using different large and complex Java projects.

1. In future work, it would be interesting to utilize the importance weights and incorporate them as one factor of the quality assurance mechanism, specifically the data quality component, of the overall MLOps system, in which the ML solution is eventually deployed.
2. For future work they aim to build a higher-level language (HLL) on top of the core calculus (CC) in which channel actions can be marked as statically/dynamically checked without writing casts explicitly (as usual with gradual typing). Through automated cast insertion, HLL can be translated to CC. To also consolidate fast excution, we aim to develop a technique to optimise away dynamic checks for channel actions that have already passed static checks. We aim to

implement CC and HLL in Clojure (which has a form of gradual typing ) on top of Discourje (which offers dynamic MPST in Clojure).

1. They plan to explore this new tool design space in future work,they have already prototyped a range abstraction, a weak memory model verifier based on sequentialization, and partial order reductions for symbolic explorations. We believe that this can also be used.
2. As future work, we aim to include more refactoring in our tool, such as Extract Class or Move Method to reduce code smells like God Class or Feature Envy. For that, we will also need to

measure more code quality metrics. We also want to learn more about other live programming approaches to improve our tool’s live mechanisms and enhance its visual guidance.

1. They observed that Shibboleth outperforms state-of-the-art ODS in classification and we leave re-implementing these tools and comparing their ranking performance for a future work.
2. In future work, we would extend the model by incorporating more relevant traffic dynamics and thus predict scenario distributions in more places, other than unsignalized crossings. We would also like to employ the model to test real autonomous driving systems and demonstrate how the model can be used for test selection and generation to enforce the scenario relevance in practice.
3. They plan to extend our meta model with more relationships, e.g., a TYPE\_OF relationship between a field and a class. At the same time, we want users to be able to choose the level of detail of the generated graph. For example, users might want to combine methods and fields into a single Class Member node or do not want to include certain relationships, like Method CALLS Method, to keep the graph database simpler.
4. . For future work toward a more privacy-compliant environment, a framework similar to a Single Sign-on scheme is desirable where the user is capable of revoking and modifying the previous data consent to a trusted data management entity.
5. In future they planned to extend the data init file to include global variables, types and deeper parameter dependencies. We also plan to extend AoT with an ability to initialize program state by using memory dumps performed on target with our tool kflat [10]. Although this approach would be device-specific, we hope to be able to automatically retrieve and use precise program state data in the OTs.
6. . In future work, we plan to incorporate the reinforcement learning techniques (e.g. policy

network) into the framework to adaptively choose the suitable deliberation processes, thereby enhancing the performance.

1. The downstream editing tasks we studied in this work are using Java. Since CoditT5’s pretraining is on the dataset consisting of six programming languages, we expect it to also perform well on editing tasks in other programming languages, but we leave empirically verifying this as future work.
2. Data Contamination. CoditT5 is pretrained on data collected from open-source projects. It is possible that similar examples in pretraining data exist in downstream tasks’ test set. While prior work has shown that data contamination may have little impact on the performance of

pretrained models in natural language processing tasks, future work can investigate this problem for pretrained models for software engineering.

1. JAttack introduces test templating, a way to define a set of programs used for testing compilers.

Templates written for JAttack could be useful for testing other program analysis tools, but we leave such studies for future work.

1. We currently support capturing static fields with primitive and array types as the global state; future work could also capture reference.
2. In this section, we contrast JAttack’s execution-based generation to static generation, describe limitations of JAttack, and provide directions for future work.
3. As future work Execution-based generation knows what exactly would be executed in a

generated program after being compiled, i.e., which parts are dead code or which parts are executable.

1. For future work, e.g., using Evo Suite to create receiver objects and inputs for instance methods. Second, we use a different name for the extracted template class from the original class, which sometimes made the template not pass Java type-checking due to circular dependencies

between the template class and other classes. We will explore editing the original class in place instead of creating a renamed template class.

1. For future work thus the ideas of template based testing and execution-based generation, can be also applied to other languages and compilers, e.g., Scala, C#, etc., or even software systems in general.
2. As our scenario-level test scenario generation method simply connects the scenarios from the seed bug and the relevant bug, the generated test scenarios often have redundant or

unnecessary steps affecting the testing experience. Based on this feedback, strengthening the coherence of the S2Rs of the generated test scenarios and simplifying the test scenarios is an essential part of our future work.

1. We plan to integrate techniques like Delta Debugging in our future work to minimize the set of crash-causing compiler flags.
2. In future they would like to explore if polarity-based data augmentation strategies could improve other related tasks in software engineering, such as sentiment analysis.
3. We leave such evaluations to future work but point again to the fact that leakage is firmly established as problematic in educational material.
4. In the future, they will explore the further usage of fault signatures for fault localization and automated patch generation/verification. We will also experiment our approach for grouping fuzzed crashes from different program versions and from different fuzzers.
5. Some limitations of Griffin and evaluation, and the plans to address them in future work.
6. Powered by Groundhog, they conducted an empirical study on a large set of real-world apps and found new classes of critical accessibility issues that should be the focus of future work in this area.
7. For future work accessibility testing and Android fundamentals and explains the details of our approach, and describes the optimizations over our technique. The evaluation of Groundhog on real-world apps with a discussion of the related research.
8. Our future work involves evaluating the extent to which the ideas presented here can be applied to other computing domains (e.g., iOS, Web), and expanding Groundhog’s support to additional assistive services and more complex gestures.
9. In future work further studies on active thread interleaving are required to better discover this type of vulnerabilities and we leave it as future work.
10. A better choice is to adjust the size K and length L based on the available computing resources adaptively, and we leave it as future work.
11. We leave the investigation of how to fit inline tests into different software- and test-design processes as future work.
12. The observations that we made throughout this work pave the way for the following future work: extend InfraSecure to detect other security vulnerabilities, integrate the tool with methods for automated patching, and port InfraSecure to other configuration management tools.
13. Since these adaptations themselves can be a research problem and have technical challenges, we do not consider them in our evaluation but plan to explore their effectiveness in our future work.
14. The interval analysis adopted in this work is an application of abstract interpretation with interval as its abstract domain. It is an interesting future work to study other abstract

interpretation-based techniques to reduce the size of integer linear constraints.

1. For future work, it would be interesting to investigate formal verification of QNNs that have more complicated architectures, activation functions, and a larger number of neurons.
2. For Linux, two configs make up 73% of the nodes and removing just four configs would leave the BDD forest with less than 100,000 nodes. These hot spots give clear indications of where to look to improve the translation and the building of the BDDs in future work.
3. As future work, we identify four main directions. First, we plan to improve the symbolic runtime, making it thread-safe and adding support for floating-point operations. Second, accurate

handling of symbolic memory accesses could make SymFusion more effective.

1. The view has “getEllipsize” attribute, and dVermin determines it can be with ellipsis (i.e., not treated as scaling issue), but its appearance is cropped. Future work would employ more

strategies to handle these corner cases to further improve the performance.

1. Future work will extend dVermin to handle inconsistencies dynamically introduced when the app is running, and more evaluation on more apps and more scale settings.
2. In future the evaluation of OverSight on real-world apps is presented. The paper concludes with a discussion of the related research and avenues.
3. In future want to investigate the way to improve our localization accuracy.
4. In future work, we aim to evaluate selection criteria to identify the most suitable CNF transformation for a given formula. To this end, we want to investigate further what makes a constraint difficult to transform. We also aim to apply our taxonomy of transformation

properties on other feature-model transformations, such as slicing.

1. An avenue for future work would be to investigate how team characteristics (e.g. size, workflow) or roles (manager, contributor) affect productivity perceptions.
2. . We use K-means clustering as our approach to clustering because it is simple and broadly used, and leave exploration of other clustering solutions as future work.
3. . Among these representative reviews, we selected examples from different categories to illustrate that privacy concerns are rarely category specific. A more detailed look into each specific privacy theme is left as future work.
4. As the examples below show, users sentiment on this topic can range from annoyed to angry. Since our methods can extract reviews on this issue, it permits future work on sentiment analysis and perhaps a deeper exploration into which types of data are more sensitive.
5. To address this problem, in future work, we will investigate ways to simplify the automatically generated query pairs and make them more representative of queries that can be encountered in practice.
6. In future work, we plan to apply AMOEBA to additional DBMSs and to improve our approach based on our current and future findings. Our current results, for instance, highlight relevant query patterns that DBMS may have difficulties processing efficiently.
7. In future work, we would like to provide guarantees for our approach by evaluating

measures such as the residual risk and reliability of a SUT after we terminate a fuzzing campaign.

1. Encouraged by the significant advantages of BugListener as shown in Sec.6, we believe that our approach could facilitate the bug discovering process and software quality

improvement.we propose potential usage scenarios as well as improvement opportunities for future work.

1. To alleviate the threat, we thoroughly analyzed where our approach performs unsatisfactorily and planned future work.
2. Instead, the workflows’ structure, which is known at design time, and whether a new workflow instance will be started suffice. If necessary, all future workflow instances are blocked to enforce Quiescence.
3. Although the current weights improve the predictions, this only sets the minimum bar. Future work can exploit automatic learning of these weights.
4. In the future, we plan to incorporate more domain specific knowledge on aspects of

training and evaluating a training ML models. For instance, one can limit the context fed to the model based on the programming language to better incorporate related information of

functions and nested scopes in a piece of code. We also plan to further investigate statement completion, including better metrics for its evaluation.

1. While our implementation is limited to a single language (Java) and a single greybox fuzzer (Zest), we believe that our results are compelling enough to have a significant impact on the field of software engineering, warranting future work.
2. In practice, we use the mature Z3 theorem prover , which worked well in our experiments. Future work might consider other constraint solvers, perhaps using newer Java APIs like JavaSMT3.
3. We also believe that future work could improve the efficacy of the generator for Rhino.
4. We suspect that tuning the rate at which hints are selected could increase the likelihood that Confetti detects bugs like this, but leave such investigation for future work.
5. We believe that this could be interesting future work, but feel that such an evaluation is outside of the scope of this paper.
6. There have also been numerous advancements in fuzzer seed selection and scheduling, most of which are complementary to Confetti, and combinations of the approaches could be

studied in future work.

1. Future work might combine Confetti with heuristics for selectively propagating taint tags through implicit flows.
2. For future works in decomposing the deep learning model into modules.
3. For the future to study how an interface could be built around the modules to take

different types of inputs. Also, we will also look into how decomposing model into modules help to debug, localize, and fix bugs [40, 41]. Another direction could be looking into other properties of the DL model e.g., fairness [5, 6], in light of the modularity.

1. The rest of the paper is organized as follows: §2 describes the motivation of our approach. §3 describes our dynamic failure symptoms detection algorithm. §4 describes the evaluation of our approach compared with prior works. §5 discusses the threats to validity. §6 discusses related works, and §7 concludes and discusses future work.
2. l. In our future work, we plan to investigate more hyperparameters such as the batch size, epoch, and dropout rate to handle the above models.
3. Ffuture work we would like to extend our approach to support additional model types, failure symptoms, and automatic repair.
4. For future work we would like to conduct studies on how to improve DNN bug repair on nonfunctional bugs such as fairness bugs . Third, we would like to extend our approach to support additional types of bugs in different stages of the ML pipeline
5. For future work we would like to explore how to leverage our findings to improve the performance of AutoML models.
6. We plan to continuously contribute to and improve this database in future work.
7. In the future, we plan to continue to grow DeepStability and also implement a web portal to encourage open-source style contributions to it.
8. In addition, much of the analysis is conducted manually which is hard to scale.

Nevertheless, we plan to expand the analysis effort as future work to improve kernel fuzzing.

1. we leave computing recall as future work. However, we observed R-CPatMiner detects 16% more change-graphs (i.e., mapped code blocks) and 15% more patterns than the

PyCPatMiner, which strongly indicates that the PyRMiner has a satisfactory recall.

1. In future we consider developing means to deal with the case of an unrealizable dynamic update, i.e., where switching cannot be forced. Currently, our technique is limited to detecting unrealizable dynamic updates, but not to further deal with them. When the system cannot force a strategy update, it is still possible that an update can be done with the

cooperation of the environment. That is, there may exist a sequence of inputs by the

environment that enables a strategy switch. This calls for synthesizing a cooperative bridge- controller.

1. In future we hope to complete the case study of deploying our implementation in Nokia Bell Labs system and report on our experience in a follow-up paper.
2. As part of future work, we plan to further investigate and explain the safety violations detected by our approach; for example, we can derive association rules between the test scenario attribute values and the resulting safety violations using association rule mining algorithms.
3. In future transferring ReDoS vulnerabilities Software engineers solve similar problems in similar ways [68]. A general version of the Consistent Sanitization Assumption is possible: that web services validate similar content in similar ways, so sanitization logic revealed by one service may be transferred to another. For example, suppose two web services use an accessibility

feature like ARIA labels [107] to label a form field as an email. If one service provides client-side sanitization logic, similar logic might be in use by the other.

1. In future work we would like to work on why are API practices more dangerous? In answering each research question, there were marked differences between ReDoS risks in traditional HTML forms as compared to the emerging approach of API specification. We

conjectured two causes (§6): providing detailed API specifications to ease the development of input validation logic, and inadvertent exposure resulting from API extraction from server-side code. We believe this finding bears further investigation.

1. In future work Improved tooling although we chose RESTler to help us reach endpoints in complex APIs, we eventually performed manual intervention for most of the 32 APIs we probed. In practice, API specifications underspecify valid interactions. When we intervened, we consulted API documentation as well as the service error messages. Incorporating NLP

techniques into automated API interactions is a natural direction for improved black-box web service testing.

1. In future work Regex dataset previous researchers have collected regex datasets from open-source software repositories with applications including improved regex usability tools and safer regex engines . To complement this effort, we contribute a dataset of web input

sanitization regexes. This dataset contains the ∼ 1850 unique regexes identified during our experiments.