Massimiliano Di Penta

List of Publications by Year
in descending order

Source: https://exaly.com/author-pdf/2557832/publications.pdf

Version: 2024-02-01

225	10,265	109264 35	114418 63
papers	citations	h-index	g-index
226	226	226	3628
all docs	docs citations	times ranked	citing authors

#	Article	IF	Citations
1	An approach for QoS-aware service composition based on genetic algorithms. , 2005, , .		668
2	An exploratory study of the impact of antipatterns on class change- and fault-proneness. Empirical Software Engineering, 2012, 17, 243-275.	3.0	277
3	Is it a bug or an enhancement?. , 2008, , .		259
4	A framework for QoS-aware binding and re-binding of composite web services. Journal of Systems and Software, 2008, 81, 1754-1769.	3.3	255
5	An Exploratory Study of the Impact of Code Smells on Software Change-proneness. , 2009, , .		194
6	Mining Version Histories for Detecting Code Smells. IEEE Transactions on Software Engineering, 2015, 41, 462-489.	4.3	192
7	On the diffuseness and the impact on maintainability of code smells: a large scale empirical investigation. Empirical Software Engineering, 2018, 23, 1188-1221.	3.0	183
8	Mining StackOverflow to turn the IDE into a self-confident programming prompter. , 2014, , .		181
9	API change and fault proneness: a threat to the success of Android apps. , 2013, , .		180
10	Release planning of mobile apps based on user reviews. , 2016, , .		172
11	An experimental investigation on the innate relationship between quality and refactoring. Journal of Systems and Software, 2015, 107, 1-14.	3.3	165
12	Mining energy-greedy API usage patterns in Android apps: an empirical study. , 2014, , .		160
13	Detecting bad smells in source code using change history information. , 2013, , .		156
14	When and Why Your Code Starts to Smell Bad (and Whether the Smells Go Away). IEEE Transactions on Software Engineering, 2017, 43, 1063-1088.	4.3	156
15	Do They Really Smell Bad? A Study on Developers' Perception of Bad Code Smells. , 2014, , .		151
16	An Empirical Study on Learning Bug-Fixing Patches in the Wild via Neural Machine Translation. ACM Transactions on Software Engineering and Methodology, 2019, 28, 1-29.	4.8	151
17	An empirical study on the maintenance of source code clones. Empirical Software Engineering, 2010, 15, 1-34.	3.0	144
18	How Clones are Maintained: An Empirical Study. , 2007, , .		141

#	Article	IF	Citations
19	The Impact of API Change- and Fault-Proneness on the User Ratings of Android Apps. IEEE Transactions on Software Engineering, 2015, 41, 384-407.	4.3	139
20	Multi-objective Cross-Project Defect Prediction. , 2013, , .		126
21	How do API changes trigger stack overflow discussions? a study on the Android SDK. , 2014, , .		118
22	User reviews matter! Tracking crowdsourced reviews to support evolution of successful apps. , 2015, , .		118
23	When and Why Your Code Starts to Smell Bad. , 2015, , .		109
24	Analyzing cloning evolution in the Linux kernel. Information and Software Technology, 2002, 44, 755-765.	3.0	106
25	When Does a Refactoring Induce Bugs? An Empirical Study. , 2012, , .		106
26	Who is going to mentor newcomers in open source projects?. , 2012, , .		105
27	Assessing and improving state-based class testing: a series of experiments. IEEE Transactions on Software Engineering, 2004, 30, 770-783.	4.3	101
28	Service-Oriented Architectures Testing: A Survey. Lecture Notes in Computer Science, 2009, , 78-105.	1.0	101
29	Improving Multi-Objective Test Case Selection by Injecting Diversity in Genetic Algorithms. IEEE Transactions on Software Engineering, 2015, 41, 358-383.	4.3	100
30	Achievements and challenges in software reverse engineering. Communications of the ACM, 2011, 54, 142-151.	3.3	98
31	An empirical investigation into the nature of test smells. , 2016, , .		98
32	Detecting missing information in bug descriptions. , 2017, , .		96
33	New Frontiers of Reverse Engineering. , 2007, , .		93
34	How the Apache community upgrades dependencies: an evolutionary study. Empirical Software Engineering, 2015, 20, 1275-1317.	3.0	93
35	An experimental investigation of formality in UML-based development. IEEE Transactions on Software Engineering, 2005, 31, 833-849.	4.3	91
36	Linguistic antipatterns: what they are and how developers perceive them. Empirical Software Engineering, 2016, 21, 104-158.	3.0	85

#	Article	IF	Citations
37	How Open Source Projects Use Static Code Analysis Tools in Continuous Integration Pipelines. , 2017, , .		82
38	Automatic generation of release notes. , 2014, , .		73
39	Object-oriented design patterns recovery. Journal of Systems and Software, 2001, 59, 181-196.	3.3	72
40	REPENT: Analyzing the Nature of Identifier Renamings. IEEE Transactions on Software Engineering, 2014, 40, 502-532.	4.3	71
41	Code siblings: Technical and legal implications of copying code between applications. , 2009, , .		69
42	ARENA: An Approach for the Automated Generation of Release Notes. IEEE Transactions on Software Engineering, 2017, 43, 106-127.	4.3	69
43	FOCUS: A Recommender System for Mining API Function Calls and Usage Patterns. , 2019, , .		69
44	Identifying Changed Source Code Lines from Version Repositories. , 2007, , .		68
45	Mining source code descriptions from developer communications. , 2012, , .		65
46	The Evolution of Project Inter-dependencies in a Software Ecosystem: The Case of Apache. , 2013, , .		65
47	Crowdsourcing user reviews to support the evolution of mobile apps. Journal of Systems and Software, 2018, 137, 143-162.	3.3	65
48	Development Emails Content Analyzer: Intention Mining in Developer Discussions (T). , 2015, , .		64
49	A large-scale empirical study on the lifecycle of code smell co-occurrences. Information and Software Technology, 2018, 99, 1-10.	3.0	64
50	An empirical study on the evolution of design patterns. , 2007, , .		61
51	An empirical comparison of methods to support QoS-aware service selection. , 2010, , .		61
52	Assessing, Comparing, and Combining State Machine-Based Testing and Structural Testing: A Series of Experiments. IEEE Transactions on Software Engineering, 2011, 37, 161-187.	4.3	61
53	Optimizing energy consumption of GUIs in Android apps: a multi-objective approach. , 2015, , .		59
54	Defect prediction as a multiobjective optimization problem. Software Testing Verification and Reliability, 2015, 25, 426-459.	1.7	59

#	Article	IF	Citations
55	An exploratory study of the evolution of software licensing. , 2010, , .		57
56	Enabling mutation testing for Android apps. , 2017, , .		57
57	A New Family of Software Anti-patterns: Linguistic Anti-patterns. , 2013, , .		56
58	Assessing staffing needs for a software maintenance project through queuing simulation. IEEE Transactions on Software Engineering, 2004, 30, 43-58.	4.3	55
59	Using acceptance tests as a support for clarifying requirements: A series of experiments. Information and Software Technology, 2009, 51, 270-283.	3.0	53
60	CODES: mining source code descriptions from developers discussions. , 2014, , .		53
61	TransientMeter: A Distributed Measurement System for Power Quality Monitoring. IEEE Transactions on Power Delivery, 2004, 19, 456-463.	2.9	50
62	The use of searchâ€based optimization techniques to schedule and staff software projects: an approach and an empirical study. Software - Practice and Experience, 2011, 41, 495-519.	2.5	50
63	How changes affect software entropy: an empirical study. Empirical Software Engineering, 2014, 19, 1-38.	3.0	50
64	Understanding and Auditing the Licensing of Open Source Software Distributions. , 2010, , .		49
65	On the Impact of Refactoring Operations on Code Quality Metrics. , 2014, , .		49
66	Would static analysis tools help developers with code reviews?. , 2015, , .		49
67	Listening to the Crowd for the Release Planning of Mobile Apps. IEEE Transactions on Software Engineering, 2019, 45, 68-86.	4.3	48
68	The effectiveness of source code obfuscation: An experimental assessment. , 2009, , .		47
69	Improving Source Code Lexicon via Traceability and Information Retrieval. IEEE Transactions on Software Engineering, 2011, 37, 205-227.	4.3	47
70	The Role of Experience and Ability in Comprehension Tasks Supported by UML Stereotypes. , 2007, , .		46
71	Too long; didn't watch!. , 2016, , .		46
72	Search-based testing of service level agreements. , 2007, , .		44

#	Article	IF	Citations
73	Putting the Developer in-the-Loop: An Interactive GA for Software Re-modularization. Lecture Notes in Computer Science, 2012, , 75-89.	1.0	44
74	A family of experiments to assess the effectiveness and efficiency of source code obfuscation techniques. Empirical Software Engineering, 2014, 19, 1040.	3.0	42
75	An empirical characterization of bad practices in continuous integration. Empirical Software Engineering, 2020, 25, 1095-1135.	3.0	42
76	Using multivariate time series and association rules to detect logical change coupling: An empirical study. , 2010, , .		41
77	How Can I Use This Method?. , 2015, , .		40
78	TIDIER: an identifier splitting approach using speech recognition techniques. Journal of Software: Evolution and Process, 2013, 25, 575-599.	1.2	39
79	Automated Reporting of Anti-Patterns and Decay in Continuous Integration. , 2019, , .		39
80	Ldiff: An enhanced line differencing tool. , 2009, , .		38
81	Tracking Your Changes: A Language-Independent Approach. IEEE Software, 2009, 26, 50-57.	2.1	38
82	Was self-admitted technical debt removal a real removal?. , 2018, , .		38
83	A Tale of CI Build Failures: An Open Source and a Financial Organization Perspective. , 2017, , .		37
84	CrossRec: Supporting software developers by recommending third-party libraries. Journal of Systems and Software, 2020, 161, 110460.	3.3	37
85	A heuristic-based approach for detecting SQL-injection vulnerabilities in web applications. , 2010, , .		36
86	Service Composition (re)Binding Driven by Applicationâ€"Specific QoS. Lecture Notes in Computer Science, 2006, , 141-152.	1.0	36
87	Improving IR-based Traceability Recovery Using Smoothing Filters. , 2011, , .		35
88	Using Test Cases as Contract to Ensure Service Compliance Across Releases. Lecture Notes in Computer Science, 2005, , 87-100.	1.0	35
89	Query-based configuration of text retrieval solutions for software engineering tasks. , 2015, , .		34
90	Supporting Software Developers with a Holistic Recommender System. , 2017, , .		34

#	Article	IF	Citations
91	What kind of questions do developers ask on Stack Overflow? A comparison of automated approaches to classify posts into question categories. Empirical Software Engineering, 2020, 25, 2258-2301.	3.0	34
92	An empirical study of the relationships between design pattern roles and class change proneness. , 2008, , .		33
93	Web Services Regression Testing. , 2007, , 205-234.		33
94	How Developers' Collaborations Identified from Different Sources Tell Us about Code Changes. , 2014, , .		32
95	Labeling source code with information retrieval methods: an empirical study. Empirical Software Engineering, 2014, 19, 1383-1420.	3.0	32
96	Assessing Test Case Prioritization on Real Faults and Mutants. , 2018, , .		32
97	Automatic Identification and Classification of Software Development Video Tutorial Fragments. IEEE Transactions on Software Engineering, 2019, 45, 464-488.	4.3	32
98	On the Use of Line Co-change for Identifying Crosscutting Concern Code. , 2006, , .		31
99	Trend Analysis and Issue Prediction in Large-Scale Open Source Systems. , 2008, , .		31
100	The life and death of statically detected vulnerabilities: An empirical study. Information and Software Technology, 2009, 51, 1469-1484.	3.0	31
101	Social interactions around cross-system bug fixings. , 2011, , .		31
102	A Method for Open Source License Compliance of Java Applications. IEEE Software, 2012, 29, 58-63.	2.1	31
103	LHDiff: A Language-Independent Hybrid Approach for Tracking Source Code Lines. , 2013, , .		31
104	Prompter. Empirical Software Engineering, 2016, 21, 2190-2231.	3.0	31
105	Automatically classifying posts into question categories on stack overflow. , 2018, , .		31
106	An eclectic approach for change impact analysis. , 2010, , .		29
107	Cooperative Co-evolutionary Optimization of Software Project Staff Assignments and Job Scheduling. Lecture Notes in Computer Science, 2011, , 127-141.	1.0	29
108	Why Developers Refactor Source Code. ACM Transactions on Software Engineering and Methodology, 2020, 29, 1-30.	4.8	29

#	Article	IF	Citations
109	A Heuristic-Based Approach to Identify Concepts in Execution Traces. , 2010, , .		28
110	Continuous Delivery Practices in a Large Financial Organization. , 2016, , .		28
111	Improving network applications security. , 2005, , .		27
112	How Long Does a Bug Survive? An Empirical Study. , 2011, , .		27
113	CodeTopics. , 2011, , .		27
114	Prompter: A Self-Confident Recommender System. , 2014, , .		27
115	Are fit tables really talking?. , 2008, , .		26
116	When and why developers adopt and change software licenses. , 2015, , .		26
117	Estimating the number of remaining links in traceability recovery. Empirical Software Engineering, 2017, 22, 996-1027.	3.0	26
118	Designing your Next Empirical Study on Program Comprehension. , 2007, , .		25
119	License usage and changes: a large-scale study on gitHub. Empirical Software Engineering, 2017, 22, 1537-1577.	3.0	25
120	Identifying licensing of jar archives using a code-search approach. , 2010, , .		24
121	Parameterizing and Assembling IR-Based Solutions for SE Tasks Using Genetic Algorithms. , 2016, , .		24
122	MDroid+. , 2018, , .		24
123	On the diffuseness and the impact on maintainability of code smells. , 2018, , .		23
124	An Empirical Study on the Usage of Transformer Models for Code Completion. IEEE Transactions on Software Engineering, 2021, , 1-1.	4.3	23
125	A language-independent software renovation framework. Journal of Systems and Software, 2005, 77, 225-240.	3.3	22
126	The Effect of Communication Overhead on Software Maintenance Project Staffing: a Search-Based Approach. , 2007, , .		22

#	Article	IF	Citations
127	License Usage and Changes: A Large-Scale Study of Java Projects on GitHub. , 2015, , .		21
128	A Quantitative and Qualitative Investigation of Performance-Related Commits in Android Apps. , 2016, , .		21
129	Automatically Learning Patterns for Self-Admitted Technical Debt Removal. , 2020, , .		21
130	On the relationship between refactoring actions and bugs: a differentiated replication. , 2020, , .		21
131	Applying a smoothing filter to improve IR-based traceability recovery processes: An empirical investigation. Information and Software Technology, 2013, 55, 741-754.	3.0	20
132	DECA. , 2016, , .		20
133	Self-Admitted Technical Debt Removal and Refactoring Actions: Co-Occurrence or More?. , 2019, , .		20
134	An empirical study on the co-occurrence between refactoring actions and Self-Admitted Technical Debt removal. Journal of Systems and Software, 2021, 178, 110976.	3.3	20
135	Migration of information systems in the Italian industry: A state of the practice survey. Information and Software Technology, 2011, 53, 71-86.	3.0	19
136	Do Developers Introduce Bugs When They Do Not Communicate? The Case of Eclipse and Mozilla. , 2012, , .		19
137	How the evolution of emerging collaborations relates to code changes: an empirical study. , 2014, , .		19
138	Patterns of developers behaviour: A 1000-hour industrial study. Journal of Systems and Software, 2017, 132, 85-97.	3.3	19
139	Self-admitted technical debt practices: a comparison between industry and open-source. Empirical Software Engineering, 2021, 26, 1.	3.0	19
140	An Approach for Search Based Testing of Null Pointer Exceptions. , 2011, , .		18
141	Configuration smells in continuous delivery pipelines: a linter and a six-month study on GitLab. , 2020, , .		17
142	Towards experimental evaluation of code obfuscation techniques. , 2008, , .		16
143	An exploratory study of identifier renamings. , 2011, , .		16
144	On the role of diversity measures for multi-objective test case selection. , 2012, , .		16

#	Article	IF	Citations
145	Towards the Integration of Versioning Systems, Bug Reports and Source Code Meta-Models. Electronic Notes in Theoretical Computer Science, 2005, 127, 87-99.	0.9	15
146	CI/CD Pipelines Evolution and Restructuring: A Qualitative and Quantitative Study. , 2021, , .		15
147	An approach for mining services in database oriented applications. , 2007, , .		14
148	TRIS: A Fast and Accurate Identifiers Splitting and Expansion Algorithm. , 2012, , .		14
149	Exploiting Natural Language Structures in Software Informal Documentation. IEEE Transactions on Software Engineering, 2019, , 1-1.	4.3	14
150	Smart Formatter: Learning Coding Style from Existing Source Code. , 2007, , .		13
151	An Exploratory Study of Factors Influencing Change Entropy. , 2010, , .		13
152	Recommending refactorings based on team co-maintenance patterns. , 2014, , .		12
153	The market for open source: An intelligent virtual open source marketplace. , 2014, , .		12
154	Recommending API Function Calls and Code Snippets to Support Software Development. IEEE Transactions on Software Engineering, 2022, 48, 2417-2438.	4.3	12
155	Identifying and locating interference issues in PHP applications: the case of WordPress. , 2014, , .		11
156	An empirical characterization of software bugs in open-source Cyberâ€"Physical Systems. Journal of Systems and Software, 2022, 192, 111425.	3.3	11
157	Who are Source Code Contributors and How do they Change?. , 2009, , .		10
158	MoMS: Multi-objective miniaturization of software. , 2011, , .		10
159	An experimental investigation on the effects of context on source code identifiers splitting and expansion. Empirical Software Engineering, 2014, 19, 1706-1753.	3.0	10
160	A Survey on Online Learning Preferences for Computer Science and Programming. , 2019, , .		10
161	Characterizing the evolution of statically-detectable performance issues of Android apps. Empirical Software Engineering, 2020, 25, 2748-2808.	3.0	10
162	Demystifying the adoption of behavior-driven development in open source projects. Information and Software Technology, 2020, 123, 106311.	3.0	10

#	Article	IF	Citations
163	A Hidden Markov Model to detect coded information islands in free text. , 2013, , .		9
164	The relation between developersâ€™ communication and fix-Inducing changes: An empirical study. Journal of Systems and Software, 2018, 140, 111-125.	3.3	9
165	A Fast Algorithm to Locate Concepts in Execution Traces. Lecture Notes in Computer Science, 2011, , 252-266.	1.0	9
166	An NLP-based Tool for Software Artifacts Analysis. , 2021, , .		9
167	Using code reviews to automatically configure static analysis tools. Empirical Software Engineering, 2022, 27, 1.	3.0	9
168	"Talking tests": a Preliminary Experimental Study on Fit User Acceptance Tests. , 2007, , .		8
169	Reuse or rewrite: Combining textual, static, and dynamic analyses to assess the cost of keeping a system up-to-date. , 2008, , .		8
170	Software migration projects in Italian industry: Preliminary results from a state of the practice survey. , 2008, , .		8
171	Frontiers of reverse engineering: A conceptual model. , 2008, , .		8
172	Guidelines on the use of Fit tables in software maintenance tasks: Lessons learned from 8 experiments. , 2008, , .		8
173	An Empirical Investigation on Documentation Usage Patterns in Maintenance Tasks. , 2013, , .		8
174	To distribute or not to distribute?. , 2018, , .		8
175	Compiler Hacking for Source Code Analysis. Software Quality Journal, 2004, 12, 383-406.	1.4	7
176	The Evolution and Decay of Statically Detected Source Code Vulnerabilities. , 2008, , .		7
177	A Study on the Relation between Antipatterns and the Cost of Class Unit Testing. , 2013, , .		7
178	SCAN: an approach to label and relate execution trace segments. Journal of Software: Evolution and Process, 2014, 26, 962-995.	1.2	7
179	An empirical study on the usefulness of Conallen's stereotypes inWeb application comprehension. , 2006, , .		6
180	A search-based approach for dynamically re-packaging downloadable applications. Proceedings of CASCON, 2007, , .	0.0	6

#	Article	IF	Citations
181	Relating the Evolution of Design Patterns and Crosscutting Concerns. , 2007, , .		6
182	Search-based inference of dialect grammars. Soft Computing, 2008, 12, 51-66.	2.1	6
183	Dynamic composition of web applications in human-centered processes. , 2009, , .		6
184	Lawful software engineering. , 2010, , .		6
185	Enabling Mutant Generation for Open- and Closed-Source Android Apps. IEEE Transactions on Software Engineering, 2022, 48, 186-208.	4.3	6
186	COCONUT: COde COmprehension Nurturant Using Traceability. , 2006, , .		5
187	SCAN: An Approach to Label and Relate Execution Trace Segments. , 2012, , .		5
188	YODA: Young and newcOmer Developer Assistant. , 2013, , .		5
189	Adversarial Attacks to API Recommender Systems: Time to Wake Up and Smell the Coffee?. , 2021, , .		5
190	Discovery of SOA patterns via model checking. , 2007, , .		4
191	What topics do Firefox and Chrome contributors discuss?. , 2011, , .		4
192	Managing and assessing the risk of component upgrades. , 2012, , .		4
193	Irish: A Hidden Markov Model to detect coded information islands in free text. Science of Computer Programming, 2015, 105, 26-43.	1.5	4
194	The C-Cube framework. , 2005, , .		3
195	Estimating the evolution direction of populations to improve genetic algorithms. , 2012, , .		3
196	Mining developers' communication to assess software quality: Promises, challenges, perils. , 2012, , .		3
197	Adversarial Machine Learning: On the Resilience of Third-party Library Recommender Systems. , 2021, , .		3
198	"Talking tests": a Preliminary Experimental Study on Fit User Acceptance Tests. First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007), 2007, , .	0.0	2

#	Article	IF	Citations
199	METAMORPHOS: MEthods and Tools for migrAting software systeMs towards web and service Oriented aRchitectures: exPerimental evaluation, usability, and tecHnOlogy tranSfer. , 2009, , .		2
200	Introduction to the special issue on search based software engineering. Empirical Software Engineering, 2011, 16, 1-4.	3.0	2
201	Nothing else matters. , 2011, , .		2
202	Sixth international workshop on traceability in emerging forms of software engineering (TEFSE 2011). , 2011, , .		2
203	How design notations affect the comprehension of Web applications. Journal of Software: Evolution and Process, 2007, 19, 339-359.	1.1	1
204	LHDiff: Tracking Source Code Lines to Support Software Maintenance Activities. , 2013, , .		1
205	Guest editorial: special section on mining software repositories. Empirical Software Engineering, 2015, 20, 291-293.	3.0	1
206	Estimating the number of remaining links in traceability recovery (journal-first abstract). , 2018, , .		1
207	Why Do Developers Reject Refactorings in Open-Source Projects?. ACM Transactions on Software Engineering and Methodology, 2022, 31, 1-23.	4.8	1
208	IWPSE 2007. , 2007, , .		0
209	Special issue on source code analysis and manipulation (SCAM 2006). Journal of Software: Evolution and Process, 2007, 19, 203-204.	1.1	0
210	Guest editorsâ€™ introduction: special issue from the 13th working conference on reverse engineering (WCRE 2006). Empirical Software Engineering, 2008, 13, 597-600.	3.0	0
211	Special Issue on Searchâ€Based Software Maintenance. Journal of Software: Evolution and Process, 2008, 20, 317-319.	1.1	0
212	Introduction to the WCRE 2007 special issue. Software Quality Journal, 2009, 17, 305-307.	1.4	0
213	Introduction to the special issue on reverse engineering (WCRE 2008). Journal of Software: Evolution and Process, 2009, 22, n/a-n/a.	1.1	0
214	Welcome from the Workshop Chair. , 2009, , .		0
215	Workshop on emerging trends in software metrics (WETSoM 2011). , 2011, , .		0
216	Five days of empirical software engineering: The PASED experience. , 2012, , .		0

#	Article	IF	Citations
217	Empirical Studies in Reverse Engineering and Maintenance: Employing Developers to Evaluate Your Approach and Tool. , 2012, , .		0
218	Message from the PROMISE 2013 Chairs. , 2013, , .		0
219	Guest editorial: special section on software maintenance and evolution. Empirical Software Engineering, 2015, 20, 410-412.	3.0	0
220	Guest editorial: Special section on mining software repositories. Empirical Software Engineering, 2016, 21, 301-302.	3.0	0
221	Guest editorial: special section on software reverse engineering. Empirical Software Engineering, 2016, 21, 749-752.	3.0	0
222	Guest editorial: special section on software analysis, evolution, and reengineering. Empirical Software Engineering, 2020, 25, 1379-1381.	3.0	0
223	How Empirical Research Supports Tool Development. , 2021, , .		0
224	3rd International Workshop on Designing Empirical Studies: Assessing the Effectiveness of Agile Methods (IWDES 2009). Lecture Notes in Business Information Processing, 2009, , 234-235.	0.8	0
225	Understanding and Improving Continuous Integration and Delivery Practice using Data from the Wild. , 2020, , .		0