Course assignment presentation;
Reading the paper ” On effectiveness of manual and automatic unit test generation : Ten Years later”

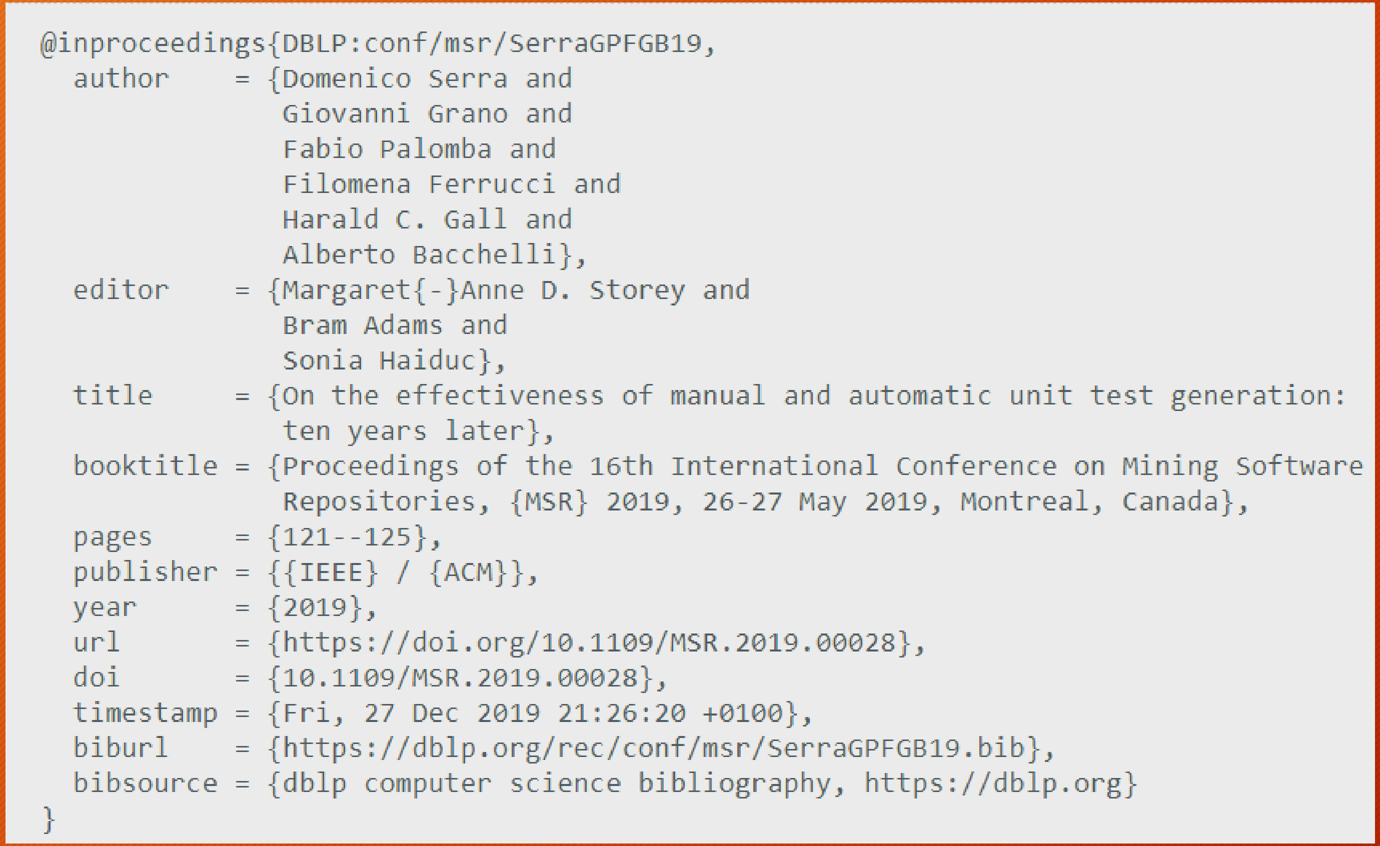
Authors: Domenico Serra, Giovanni Grano, Fabio Palomba, Filomena Ferrucci, Harald C. Gall, Alberto Bacchelli

MSR course 2020/21 at uniko, CS department, Software language team

Shray Sharma

Full Citation

https://dblp.org/rec/conf/msr/SerraGPFGB19



```
@inproceedings{DBLP:conf/msr/SerraGPFGB19,
  author    = {Domenico Serra and
               Giovanni Grano and
               Fabio Palomba and
               Filomena Ferrucci and
               Harald C. Gall and
               Alberto Bacchelli},
  editor    = {Margaret{-}Anne D. Storey and
               Bram Adams and
               Sonia Haiduc},
  title     = {On the effectiveness of manual and automatic unit test generation:
               ten years later},
  booktitle = {Proceedings of the 16th International Conference on Mining Software
               Repositories, {MSR} 2019, 26-27 May 2019, Montreal, Canada},
  pages     = {121--125},
  publisher = {{IEEE} / {ACM}},
  year      = {2019},
  url       = {https://doi.org/10.1109/MSR.2019.00028},
  doi       = {10.1109/MSR.2019.00028},
  timestamp = {Fri, 27 Dec 2019 21:26:20 +0100},
  biburl    = {https://dblp.org/rec/conf/msr/SerraGPFGB19.bib},
  bibsource = {dblp computer science bibliography, https://dblp.org}
}
```

Motivation

Testing plays a very important role in software development life cycle in guaranteeing the quality of a product. Specially in the context of continuous integration where the developers are performing both the tests including integration and unit testing.

This process of testing can be enhanced with the use of automation tests which can then lead to having some positive and negative sides. The paper under discussion Serra et al[1] is a reproduction of Bacchelli et al.[2] comparing the growth of automation testing over the span of 10 years.

This comparison is answered by the below mentioned research questions for Serra et al[1] :

RQ1: What is the code coverage of manual and automated tests?

RQ2: What is the mutation score of manual versus automated tests?(ability of test case to cover variation in code that makes it defective)

RQ3: what is the fault detection ability of manual tests versus automatically generated ones.

Data Collection

The data used by Bacchelli et al.[2] was not made public and the only information given out was that it used FREENET project [3]. As to get a best comparison the data set had to remain the same and the tools for automation of unit testing had to be changed. The data was extracted in three steps:

1. Detecting Manually written regression tests:
   - The authors of Serra et al[1] started by scanning each commit of the version history of FREENET and looking for T^{C}_{i} , namely the commit *i* that firstly introduced the test suite T for the class *c*. Since the test suite followed the naming convention that dictates that each test has the suffix Test after the name of the tested production class, the paper could do an accurate mapping between tests and the corresponding production classes Detecting defective classes.

Data Collection

2. Detecting defective classes.
   - The authors of Serra et al[1] executed a test suite introduced at T^{c}_{i} over the previous commits of c: to see if T fails on the production class in a certain commit c_i, it was assumed that one defect is found. The procedure was repeated until all the defects declared in the reference paper are found.

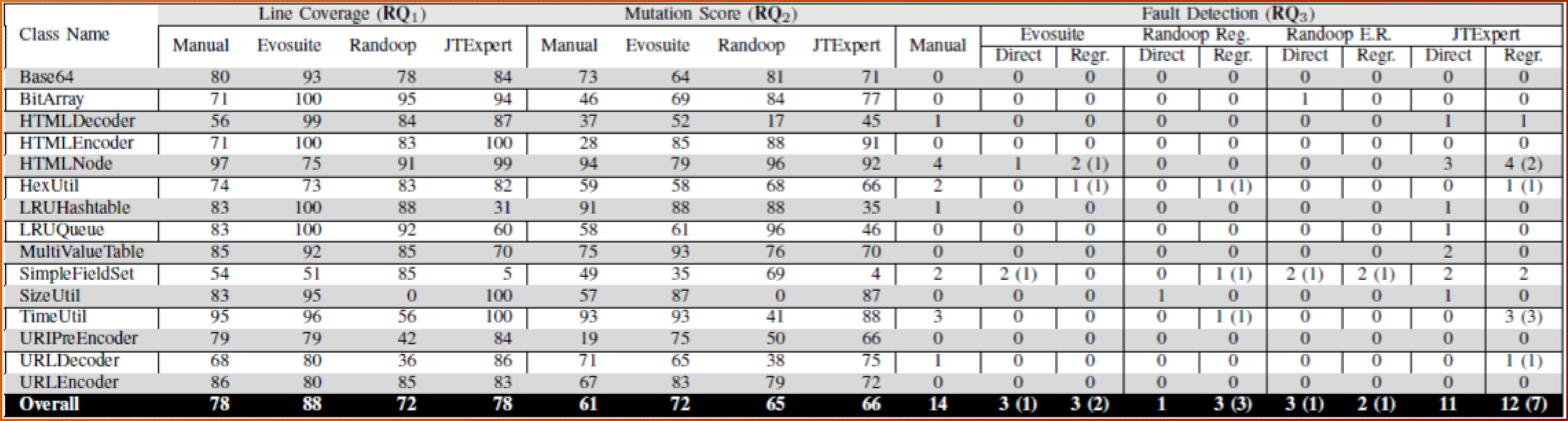
3. Generating automatic regression tests.
   - Once the authors of Serra et al[1] identified the set of defective classes and their fixes, they could generate regression tests automatically. To have an overview of the performance of different existing tools, the paper considered three testing tools, namely EVOSUITE [3], RANDOOP [4],and JTEXPERT [5]

Data Analysis

- The paper Serra et al[1] strictly focuses to address the research questions, the paper first computes line coverage (RQ1) and mutation score (RQ2) for both manually and automatically written test suites. To compute the mutation score the paper relies on PIT [22].

- To answer RQ3, the paper Serra et al[1] computes the number of times manually and automatically generated tests can correctly identify a failure: for the automated tests, it was considered a failure to be detected if the tools identify it in at least one of the runs

Data Analysis



Class Name	Line Coverage (RQ$_1$)				Mutation Score (RQ$_2$)				Fault Detection (RQ$_3$)								
	Manual	Evosuite	Randoop	JTExpert	Manual	Evosuite	Randoop	JTExpert	Manual	Evosuite		Randoop Reg.		Randoop E.R.		JTExpert	
										Direct	Regr.	Direct	Regr.	Direct	Regr.	Direct	Regr.
Base64	80	93	78	84	73	64	81	71	0	0	0	0	0	0	0	0	0
BitArray	71	100	95	94	46	69	84	77	0	0	0	0	0	1	0	0	0
HTMLDecoder	56	99	84	87	37	52	17	45	1	0	0	0	0	0	0	1	1
HTMLEncoder	71	100	83	100	28	85	88	91	0	0	0	0	0	0	0	0	0
HTMLNode	97	75	91	99	94	79	96	92	4	1	2 (1)	0	0	0	0	3	4 (2)
HexUtil	74	73	83	82	59	58	68	66	2	0	1 (1)	0	1 (1)	0	0	0	1 (1)
LRUHashtable	83	100	88	31	91	88	88	35	1	0	0	0	0	0	0	1	0
LRUQueue	83	100	92	60	58	61	96	46	0	0	0	0	0	0	0	1	0
MultiValueTable	85	92	85	70	75	93	76	70	0	0	0	0	0	0	0	2	0
SimpleFieldSet	54	51	85	5	49	35	69	4	2	2 (1)	0	0	1 (1)	2 (1)	2 (1)	2	2
SizeUtil	83	95	0	100	57	87	0	87	0	0	0	1	0	0	0	1	0
TimeUtil	95	96	56	100	93	93	41	88	3	0	0	0	1 (1)	0	0	0	3 (3)
URIPreEncoder	79	79	42	84	19	75	50	66	0	0	0	0	0	0	0	0	0
URLDecoder	68	80	36	86	71	65	38	75	1	0	0	0	0	0	0	0	1 (1)
URLEncoder	86	80	85	83	67	83	79	72	0	0	0	0	0	0	0	0	0
Overall	**78**	**88**	**72**	**78**	**61**	**72**	**65**	**66**	**14**	**3 (1)**	**3 (2)**	**1**	**3 (3)**	**3 (1)**	**2 (1)**	**11**	**12 (7)**

Serra et al[1]

Results

The paper Serra et al[1] comes to a clear conclusion by answering all the research questions.

1. The findings of Serra et al[1] revealed that current automatic test case generation tools can optimize coverage and mutation score more than manually written tests

2. The improvement in the last ten years has not been dramatic. In terms of defect finding, while there has been little improvement over the last years, it has not reached the same level of quality as the other dimensions.

Reproducibility

- The paper Serra et al[1] has made its data set public which can be used as a base to compare the unit testing tools upon and come to a clear conclusion and define if there has been improvements.
- The paper heavily relies on usage and acquiring of test automation tools, to which the data from step 1 can be fed and the results can be compared.
- The problem in reproducing this paper would come in acquiring the licenses for the tools to create a clear comparison of the development of tools over time.

References

1. D. Serra, G. Grano, F. Palomba, F. Ferrucci, H. C. Gall and A. Bacchelli, "On the Effectiveness of Manual and Automatic Unit Test Generation: Ten Years Later," *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, Montreal, QC, Canada, 2019, pp. 121-125, doi: 10.1109/MSR.2019.00028.
2. A. Bacchelli, P. Ciancarini, and D. Rossi, "On the effectiveness of manual and automatic unit test generation," in The Third International Conference on Software Engineering Advances. IEEE, 2008, pp. 252–257.
3. G. Fraser and A. Arcuri, "Evosuite: automatic test suite generation for object-oriented software," in Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering. ACM, 2011, pp. 416–419.
4. C. Pacheco, S. K. Lahiri, M. D. Ernst, and T. Ball, "Feedback-directed random test generation," in Proceedings of the 29th international conference on Software Engineering. IEEE Computer Society, 2007,pp. 75–84.
5. A. Sakti, G. Pesant, and Y.-G. Guéhéneuc, "Instance generator and problem representation to improve object oriented code coverage," IEEE Transactions on Software Engineering, vol. 41, no. 3, pp. 294–313, 2015.