PhD in Information Technology and Electrical Engineering

Università degli Studi di Napoli Federico II

PhD Student: Marco Castelluccio

XXXI Cycle

Training and Research Activities Report – Third Year

Tutor: Carlo Sansone – co-Tutor: Luisa Verdoliva

1. Information

PhD Student: Marco Castelluccio

MS title: Computer Engineering – University of Naples Federico II

PhD cycle: XXXI – ITEE University of Naples Federico II

Fellowship type: PhD student without grant

Tutor: Carlo Sansone – **co-Tutor:** Luisa Verdoliva

I received my MS degree (cum laude) in Computer Engineering from the Università degli Studi di Napoli Federico II in September 2015.

I have been a Software Engineer at Mozilla since November 2015, after two internships and a period of contracting with the same company during my graduate studies. I am a Senior Software Engineer at Mozilla since August 2017.

2. Study and Training activities

Seminars

1. On the Evolution of NLP, QA, and IE, and Current Research and Commercial Trends – Dan Moldovan (Professor at the University of Texas at Dallas and Founder of Lymba Corporation) – Association for Computing Machinery (ACM)
2. Dynamic Programming for the Masses - Patrick Madden (Associate Professor at SUNY Binghamton) – Association for Computing Machinery (ACM)
3. Deep Learning for Everyone with NVIDIA GPU Cloud - Chris Kawalek (Sr. Product Marketing Manager, NVIDIA GPU Cloud, NVIDIA), Phil Rogers (Chief Software Architect - Compute Server, NVIDIA) – NVIDIA Corporation

4. Reactive Microsystems—The Evolution of Microservices at Scale - Jonas Bonér (Founder and CTO of Lightbend (formerly known as Typesafe), inventor of the Akka project, and ACM Queue Case Study contributor) – Association for Computing Machinery (ACM)
5. Is Continuous Adoption in SE Achievable and Desirable? - Gail Murphy (Professor and Vice-President of Research and Innovation at the University of British Columbia) – Association for Computing Machinery (ACM)
6. HTP-NLP: A New NLP System for High Throughput Phenotyping - Peter Elkin (Professor at the University of Buffalo) – Association for Computing Machinery (ACM)
7. QISKit: A Swiss Army Knife for Quantum Computation - Jay Gambetta (Manager, Theory of Quantum Computing and Information, IBM; Panelist, 50 Years of ACM AM Turing Award Celebration) – Association for Computing Machinery (ACM)
8. Artificial Intelligence with Bayesian Networks - Data Mining, Knowledge Modeling and Causal Analysis - Dr. Lionel Jouffe (Co-founder and CEO of France-based Bayesia S.A.S.) – Association for Computing Machinery (ACM)
9. Hierarchical Adversarially Learned Inference - Negar Rostamzadeh (Research Scientist at Element AI) – Association for Computing Machinery (ACM)
10. Develop and Deploy Deep Learning Services at the Edge with IBM's Edge Solution - Chris Dye (Senior Software Developer, IBM), Amit Goel (Product Line Manager of Autonomous Machines, NVIDIA) - NVIDIA
11. Architecture for Sustainable Systems - David Weiss (Co-founder and Partner of Sustainable Software LLC) – Association for Computing Machinery (ACM)
12. The Programmer's Oath - Robert Martin (Co-founder of cleancoders.com and Founder of Uncle Bob Consulting LLC) – Association for Computing Machinery (ACM)
13. Computing for Disasters - Robin Murphy (Raytheon Professor at Texas A&M University, Director of the Humanitarian Robotics and Artificial Intelligence

Laboratory, and ACM Eugene L. Lawler Award recipient) – Association for Computing Machinery (ACM)

14. Communicating with the New Machine: Human Insight at Machine Scale - Kristian Hammond (Professor of Computer Science at Northwestern University, and Co-founder of Narrative Science) – Association for Computing Machinery (ACM)
15. Software Product Line Analysis and Construction - David Weiss (Co-founder and Partner of Sustainable Software LLC) – Association for Computing Machinery (ACM)
16. What Can Kotlin Do for Me? - Hadi Hariri (VP of Developer Advocacy at JetBrains) – Association for Computing Machinery (ACM)
17. Advances in Socio-Behavioral Computing - Tomek Strzalkowski (Director of the Institute for Informatics, Logics, and Security Studies and Professor at SUNY Albany) – Association for Computing Machinery (ACM)
18. Risky File Management - Audris Mockus (Ericsson-Harlan D. Mills Chair Professor at the University of Tennessee) – Association for Computing Machinery (ACM)
19. Is the Blockchain a Branch of AI? - Toufi Saliba (CEO at PrivacyShell and TodaCorp; and Chair of the ACM Practitioners Board Conference Committee) – Association for Computing Machinery (ACM)
20. The History of Software Engineering - Grady Booch (ACM Fellow and Chief Scientist for Software Engineering at IBM Research) – Association for Computing Machinery (ACM)
21. Explicitly encoded high-dimensional semantic spaces: An encoding model for hypothesis driven research on large heterogeneous data streams - Jussi Karlgren (Adjunct Professor of Language Technology at KTH Royal Institute of Technology, Adjunct Professor of Language Technology at Helsinki University, and a Founding Partner of the text analysis company Gavagai) – Association for Computing Machinery (ACM)

22. Journey to a Real-Time Enterprise - Neha Narkhede (Co-Founder and CTO of Confluent and Co-Creator of Apache Kafka) – Association for Computing Machinery (ACM)
23. Socially Assistive Robotics - Dr. Maja Matarić (Chaired Professor and Vice Dean for Research, University of Southern California; Chief Science Officer, Embodied, Inc.) – Association for Computing Machinery (ACM)
24. Fundamentals of Measurement and Data Analysis for Software Engineers: Part 1 - Basic Concepts of Measurement Theory - Dennis J. Frailey (Principal Fellow of Raytheon Corporation and ACM Fellow) – Association for Computing Machinery (ACM)
25. Fundamentals of Measurement and Data Analysis for Software Engineers - Part 2 - Basic Data Analysis Techniques - Dennis J. Frailey (Principal Fellow of Raytheon Corporation and ACM Fellow) – Association for Computing Machinery (ACM)
26. Fundamentals of Measurement and Data Analysis for Software Engineers - Part 3 - Statistical Techniques for Data Analysis - Dennis J. Frailey (Principal Fellow of Raytheon Corporation and ACM Fellow) – Association for Computing Machinery (ACM)
27. Scala Is for... Distributed Systems? Web Apps? Front-end Development? - Heather Miller (Executive Director and Co-Founder of the Scala Center at EPFL; Carnegie Mellon University) – Association for Computing Machinery (ACM)
28. Fundamentals of Measurement and Data Analysis for Software Engineers - Part 4 - Analysis Techniques Especially Suited to Software - Dennis J. Frailey (Principal Fellow of Raytheon Corporation and ACM Fellow) – Association for Computing Machinery (ACM)
29. Adversarial Machine Learning - Ian Goodfellow (Staff Research Scientist at Google Brain) – Association for Computing Machinery (ACM)
30. The Changing Landscape of Refactoring Research in the Last Decade - Danny Dig (Associate Professor of Computer Science at Oregon State University) – Association for Computing Machinery (ACM)

31. The Bayesian Zig Zag: Developing Probabilistic Models Using Grid Methods and MCMC - Allen Downey (Professor of Computer Science at Olin College) – Association for Computing Machinery (ACM)
32. Combinatorial Testing: Progress in Automating Test Design - George Sherwood (founder and CEO of Testcover.com) – Association for Computing Machinery (ACM)
33. Project Jupyter: From Computational Notebooks to Large Scale Data Science with Sensitive Data - Brian Granger (Associate Professor of Physics and Data Science at Cal Poly State University and co-creator of Project Jupyter) – Association for Computing Machinery (ACM)
34. What Makes Expert Software Designers Successful? - André van der Hoek (Professor and Chair of the Department of Informatics at the University of California, Irvine) – Association for Computing Machinery (ACM)
35. Explainable Machine Learning Models for Healthcare AI - Ankur Teredesai (Co-Founder and CTO, KenSci and Professor at University of Washington - Tacoma; SIGKDD), Dr. Carly Eckert (Medical Director of Clinical Informatics, KenSci), Muhammad Aurangzeb Ahmad (Muhammad Aurangzeb Ahmad, Principal Data Scientist, KenSci), Vikas Kumar (Data Scientist, KenSci) – Association for Computing Machinery (ACM)
36. 56th Crest Open Workshop, “Code Review and Continuous Inspection/Integration” – University College London

External courses

1. Debugging and Profiling C++ Applications for Linux – KDAB Group

Year	Modules	Seminars	Research	Tot.
1	21 (20)	7 (5)	35 (35)	63 (60)

2	16 (9)	12 (6)	45 (42)	73 (60)
3	3 (0)	10 (5)	51 (55)	64 (60)
Tot.	**40 (30-70)**	**29 (10-30)**	**131 (80-140)**	**180 (180)**

3. Research activity

- **Convolutional Neural Networks for Classification of Remote Sensing Images** – Continued work on using convolutional neural networks for classification of remote sensing images started during the MSc thesis [1], analyzing results on additional datasets [2].

  The results of the work have been published: M. Castelluccio, G. Poggi, C. Sansone, L. Verdoliva – Training Convolutional Neural Networks for Semantic Classification of Remote Sensing Imagery – JURSE2017.

- **Automating the Understanding of Groups of Crash Reports** – Many Software Engineering studies have focused on improving the bucketing of crash reports with different techniques (e.g. by using different distance metrics, like Levenshtein [3], custom stack-based metrics [4,5], by using information retrieval techniques [6,7], etc.).

  The focus of my work is instead on how to automatically describe the buckets' properties in the most useful way for developers. Understanding what makes a crash group meaningfully different than other groups is indeed very often useful for debugging (and in some cases even enough for fixing the crash, e.g. by blocklisting a certain graphics card), but it involves a tedious and error-prone manual exploration of the database of crashes.

  We devised an algorithm, inspired by contrast-set mining algorithms such as STUCCO [8,9] and CIGAR [10], to automatically find statistically significant properties (correlations) in crash groups. Developers used to spend a fair amount of time analysing crash groups, which meant that a) they were not spending their time actually developing a fix for the crash; and b) they might have missed something in their exploration of the crash data (there is a large number of attributes in crash reports and it is hard and error-prone to manually analyse everything). Our algorithm helps developers and release managers understand crash reports more easily and in an automated way, helping in pinpointing the root cause of the crash. The tool implementing the algorithm has been deployed on Socorro, Mozilla's crash reporting service.

The results of the work have been published: M. Castelluccio, C. Sansone, L. Verdoliva, and G. Poggi – Automatically analyzing groups of crashes for finding correlations – Proceedings of the 2017 11th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2017).

- **Emprical Study of Uplifts in Mozilla Firefox** – Collaboration with the École Polytechnique de Montréal on studying uplifts (backports) in the Mozilla Firefox software.

  Mozilla Firefox has adopted a rapid-release model [11,12], called the "train model". In rapid release development processes, patches that fix critical issues, or implement high-value features are often promoted directly from the development channel to a stabilization channel, potentially skipping one or more stabilization channels. This practice is called patch uplift. Patch uplift is risky, because patches that are rushed through the stabilization phase can end up introducing regressions in the code. The aim is to understand the properties of these critical changes vs normal changes; understand which uplifts introduced bugs (by using the SZZ algorithm proposed in [13]) and why; with the ultimate goal of building a model to predict the riskiness of an uplift.

  We have examined patch uplift operations at Mozilla, to identify the characteristics of uplifted patches that introduce regressions, both through statistical and manual analyses, and through interviews with Mozilla release managers. Results show that most patches are uplifted because of a wrong functionality or a crash. Uplifted patches that lead to faults tend to change a higher number of lines of code, and most of the faults are due to semantic or memory errors in the patches. Also, release managers are more inclined to accept patch uplift requests that concern certain specific components, and–or that are submitted by certain specific developers. The results of this study could be used to train machine learners able to recommend patches that are safe to uplift for release management teams.

  The results of this work have been published: M. Castelluccio, L. An, and F. Khomh - Is It Safe to Uplift This Patch? An Empirical Study on Mozilla Firefox - In proceedings of the 33rd International Conference on Software Maintenance and Evolution (ICSME 2017). The paper received a IEEE TCSE Distinguished Paper

Award, and we were invited to publish it, with some additions, on the “Empirical Software Engineering” journal (EMSE).

After additions of more than 30% of new content, we have presented the paper to be published on the “Empirical Software Engineering” journal. It has been accepted and it is due to be published soon.

- **Automatically detecting web-compatibility issues** – Web compatibility is a very important concern for browser vendors. While there are standards describing APIs and behaviors of web browsers, there are often corner cases that are not fully specified and thus implemented differently. Given the fast pace in browser development, there are also features which are implemented before being standardized and agreed upon between stakeholders. This leads to problems in the field, as website developers sometimes assume the standards are implemented exactly in the same way or because they simply only test with the most widely used browsers, some websites present compatibility problems, working correctly in some browsers but not in others.

  Currently, spotting websites which present compatibility problems is pretty much a manual effort. Volunteers and QA professionals manually inspect web sites in different browsers and notify issues on bug tracking systems such as Bugzilla [14] or the Web Compatibility issue tracker [15] (an issue tracker which was built specifically for this kind of problems). Once the issues have been identified, browser vendors can then contact the website authors to notify them of the problems or even suggest fixes. Clearly, the manual process is prone to errors (after inspecting many websites, people can start overlooking differences), very slow and above all it is very far from being exhaustive, since the number of web sites is huge and grows continuously. Moreover, it has to be repeated periodically, since websites are frequently modified.

  The purpose of this work is presenting an automatic technique that browser vendors can adopt to quickly spot websites which are not working correctly with their product.

  In order to automatically detect web pages that behave differently in different browsers, we propose to use convolutional neural networks (proposed by Fukushima [16] and later refined by LeCun et al. [17]) to evaluate differences in screenshots of page rendering in different browsers. In particular, we are planning to use siamese networks [18], with different training techniques (e.g. finetuning on network pretrained with ImageNet [19]) and different architectures (e.g. VGG [20]).

  The dataset that we are building is going to be open to all to reproduce and improve our results.

- **What Makes a Code Change Easier to Review - An Empirical Investigation on Code Change Reviewability** – Peer code review is a practice widely adopted in software projects to improve the quality of code. In current code review practices, code changes are manually inspected by developers other than the author before these changes are integrated into a project or put into production. We conducted a study to obtain an empirical understanding of what makes a code change easier to review. To this end, we surveyed published academic literature and sources from gray literature (e.g. blogs and white papers), we interviewed ten professional developers, and we designed and deployed a reviewability evaluation tool that professional developers used to rate the reviewability of 98 changes. We find that reviewability is defined through several factors, such as the change description, size, and coherent commit history. We provide recommendations for practitioners and researchers.

- **Why Did This Reviewed Code Crash? - An Empirical Study of Mozilla Firefox** – Code review, i.e. the practice of having other team members critique changes to a software system, is a pillar of modern software quality assurance approaches. Although this activity aims at improving software quality, some high-impact defects, such as crash-related defects, can elude the inspection of reviewers and escape to the field, affecting user satisfaction and increasing maintenance overhead. In this research, we investigate the characteristics of crash-prone code, observing that such code tends to have high complexity and depend on many other classes. In the code review process, developers often spend a long time on and have long discussions about crash-prone code.

  We manually classify a sample of reviewed crash-prone patches according to their purposes and root causes. We observe that most crash-prone patches aim to improve performance, refactor code, add functionality, or fix previous crashes. Memory and semantic errors are identified as major root causes of the crashes.

  Our results suggest that software organizations should apply more scrutiny to these types of patches, and provide better support for reviewers to focus their inspection effort by using static analysis tools.

- **Understanding Flaky Tests: Relevance, Nature, and Challenges** – Regression testing allows developers to control that new defects are not introduced in a newly committed code change. Unfortunately, even tests might be defective. One of the issues reported by practitioners and researchers concerning tests is *flakiness*, which consists in tests that exhibit a seemingly random passing and failing outcome when run against the same code. While the research community has investigated automated solutions to locate and fix flaky tests, there is still a limited knowledge on (i) how much the problem is relevant in practice, (ii) what is the cause of the flakiness, and (iii) what are the challenges developers perceive when dealing with flaky tests.

With the aim of increasing our scientific knowledge on these aspects, we conduct an empirical investigation that relies on software repository data (pertaining to 391 software systems), a novel dataset of 200 flaky tests classified by practitioners who fixed those tests, and the opinions of 120 developers collected in an online questionnaire. We find that the problem is relevant and characterized by several causes, of which four have never been reported before, despite being costly to fix. We find that, among the challenges faced by developers, test reproduction and the classification of the cause for flakiness are the most pressing.

- **An Empirical Study of DLL Injection Bugs in the Firefox Ecosystem** – DLL injection is a technique used for executing code within the address space of another process by forcing the load of a dynamic-link library. In a software ecosystem, the interactions between the host and third-party software increase the maintenance challenges of the system and may lead to bugs. In this work, we empirically investigate bugs that were caused by third-party DLL injections into the Mozilla Firefox browser. Among the studied DLL injection bugs, we found that 88.8% of the bugs led to crashes and 55.3% of the bugs were caused by antivirus software. Through a survey with third-party software vendors, we observed that some vendors did not perform any QA with pre-release versions nor intend to use a public API (WebExtensions) but insist on using DLL injection. To reduce DLL injection bugs, host software vendors may strengthen the collaboration with third-party vendors, e.g. build a publicly accessible validation test framework. Host software vendors may also use a whitelist approach to only allow vetted DLLs to inject.

[1] M. Castelluccio, G. Poggi, C. Sansone, and L. Verdoliva, "Land Use Classification in Remote Sensing Images by Convolutional Neural Networks", arXiv preprint.

[2] G.S. Xia, W. Yang, J. Delon, Y. Gousseau, H. Sun, and H. Maitre, "Structrual High-Resolution Satellite Image Indexing", in Processings of the ISPRS, TC VII Symposium Part A: 100 Years ISPRS—Advancing Remote Sensing Science, Vienna, Austria, 5–7 July 2010.

[3] Tejinder Dhaliwal, Foutse Khomh, Ying Zou, "Classifying Field Crash Reports for Fixing Bugs: A Case Study of Mozilla Firefox", in Proceedings of the 2011 27th IEEE International Conference on Software Maintenance (ICSM '11), pp. 333-342.

[4] G. Lohman, J. Champlin, and P. Sohn. 2005. Quickly Finding Known Software Problems via Automated Symptom Matching. In Proc. of the Second International Conference on Automatic Computing. 101–110.

[5] Y. Dang, R. Wu, H. Zhang, D. Zhang, and P. Nobel, "ReBucket: a method for clustering duplicate crash reports based on call stack similarity", in Proceedings of the 34th International Conference on Software Engineering (ICSE '12), pp. 1084-1093.

[6] J. Lerch, and M, Mezini, "Finding Duplicates of Your Yet Unwritten Bug Report", in Proceedings of the 2013 17th European Conference on Software Maintenance and Reengineering (CSMR '13). IEEE Computer Society, Washington, DC, USA, 69-78.

[7] J. C. Campbell, E. A. Santos, and A. Hindle, "The unreasonable effectiveness of traditional information retrieval in crash report deduplication", in Proceedings of the 13th International Conference on Mining Software Repositories (MSR '16). ACM, New York, NY, USA, 269-280.

[8] S.D. Bay, and M.J. Pazzani, "Detecting change in categorical data: Mining contrast sets", in Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'99), pp. 302–306, San Diego, U.S.A., August 1999.

[9] S.D. Bay, and M.J. Pazzani, "Detecting group differences: Mining contrast sets", Data Mining and Knowledge Discovery, 5(3):213–246, 2001.

[10] R. J. Hilderman, and T. Peckham, "A Statistically Sound Alternative Approach to Mining Contrast Sets", in Proceedings of the 4th Australasian Data Mining Conference (AusDM), 2005.

[11] https://mozilla.github.io/process-releases/draft/development_overview/.

[12] M. V. Mäntylä, B. Adams, F. Khomh, E. Engström, and K. Petersen, "On rapid releases and software testing: a case study and a semi-systematic literature review", Empirical Softw. Engg. 20, 5 (October 2015), 1384-1425.

[13] J. Śliwerski, T. Zimmermann, and A. Zeller, "When do changes induce fixes?", in Proceedings of the 2005 international workshop on Mining software repositories (MSR '05). ACM, New York, NY, USA, 1-5.

[14] Mozilla, "Bugzilla - https://bugzilla.mozilla.org/".

[15] Mozilla, "Web compatibility issue tracker - https://webcompat.com/".

[16] Kunihiko Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," Biological Cybernetics, vol. 36, no. 4, pp. 193–202, Apr 1980.

[17] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," Neural Comput., vol. 1, no. 4, pp. 541–551, Dec. 1989.

[18] Raia Hadsell, Sumit Chopra, and Yann LeCun, “Dimensionality reduction by learning an invariant mapping,” in Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2, Washington, DC, USA, 2006, CVPR ’06, pp. 1735–1742, IEEE Computer Society.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” in CVPR09, 2009.

[20] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” CoRR, vol. abs/1409.1556, 2014.

4. Products

I Year

M. Castelluccio, G. Poggi, C. Sansone, L. Verdoliva – Land Use Classification in Remote Sensing Images by Convolutional Neural Networks – https://arxiv.org/abs/1508.00092 – 201 citations

II Year

M. Castelluccio, G. Poggi, C. Sansone, L. Verdoliva – Training Convolutional Neural Networks for Semantic Classification of Remote Sensing Imagery – Joint Urban Remote Sensing Event (JURSE2017), Pages 1-4

M. Castelluccio, C. Sansone, L. Verdoliva, and G. Poggi – Automatically analyzing groups of crashes for finding correlations – Proceedings of the 2017 11th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2017), Pages 717-726

M. Castelluccio, L. An, and F. Khomh – Is It Safe to Uplift This Patch? An Empirical Study on Mozilla Firefox – In proceedings of the 33rd International Conference on Software Maintenance and Evolution (ICSME 2017), Pages 411-421. Received IEEE TCSE Distinguished Paper Award. Invited for publication on “Empirical Software Engineering” journal.

III Year

A. Ram, A.A. Sawant, M. Castelluccio, and A. Bacchelli – What Makes A Code Change Easier to Review? An Empirical Investigation On Code Change Reviewability – Proceedings of the 2018 26th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2018), Pages 201-212

L. An, F. Khomh, S. McIntosh, and M. Castelluccio – Why Did This Reviewed Code Crash? An Empirical Study of Mozilla Firefox – The 25th Asia-Pacific Software Engineering Conference (APSEC 2018) [accepted]

M. Castelluccio, L. An, and F. Khomh – An Empirical Study of Patch Uplift in Rapid Release Development Pipelines – Empirical Software Engineering journal (EMSE) [accepted]

M. Eck, F. Palomba, M. Castelluccio, and A. Bacchelli – Understanding Flaky Tests: Relevance, Nature, and Challenges – ICSE2019 [submitted]

L. An, M. Castelluccio, and F. Khomh – An Empirical Study of DLL Injection Bugs in the Firefox Ecosystem – EMSE [submitted]

5. Conferences

Presentation of the paper ‘Automatically analyzing groups of crashes for finding correlations’ at the 11th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, PADERBORN, GERMANY, September 04 – 08.

Attended Free and Open-Source Software Developers' European Meeting (FOSDEM) in 2017 and 2018.

6. Tutorship

- Tutored Google Summer of Code student in 2017 for a project about crash clustering.
- Tutored Outreachy student in 2018 for a project about code coverage and testing.
- Tutored Google Summer of Code student in 2018 for a project about web compatibility issues detection.