Global-Aware Recommendations for Repairing Violations in Exception Handling*

Eiji Adachi Barbosa
Digital Metropolis Institute, UFRN
Natal, RN, Brazil
eijiadachi@imd.ufrn.br

Alessandro Garcia
Informatics Department, PUC-Rio
Rio de Janeiro, RJ, Brazil
afgarcia@inf.puc-rio.br

ABSTRACT

This paper presents an extended abstract incorporated as a journal-first paper into the ICSE'18 program.

CCS CONCEPTS

• **Software and its engineering → Software maintenance tools**;

KEYWORDS

Exception Handling, Recommender system, Software maintenance

ACM Reference Format:
Eiji Adachi Barbosa and Alessandro Garcia. 2018. Global-Aware Recommendations for Repairing Violations in Exception Handling. In *ICSE '18: ICSE '18: 40th International Conference on Software Engineering , May 27-June 3, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 1 page. https://doi.org/10.1145/3180155.3182539

1 EXTENDED ABSTRACT

Exception handling mechanisms allow developers to structure their programs to deal with abnormal or unexpected conditions that occur at runtime [5]. Despite exception handling being aimed at improving software robustness, recurring failures in software systems occur due to faults in exception handling code [2, 4].

In previous studies [2, 3], we observed most faults in exception handling stem from violations in the propagation path of exceptions that traverse different methods, the so-called global exceptions. We observed from 85% to 90% [3] of exception handling-related failures were caused by violations related to global exceptions.

Despite of effort in detecting exception handling violations [1], there is still no solution aimed at assisting developers in repairing these violations. In fact, repairing violations in exception handling is a difficult and error-prone task. It requires understanding how global exceptions are intended to be used and how the implementation of their propagation path violates the intended exception handling design. It also requires knowing how to change the source code so that the implemented path adheres to the intended design. Developers are required to know the correct place where the exception should be signaled, the correct type that should be used to instantiate the exception and the correct places where the exception should be handled. And this type of information is not available in the source code of the systems.

In this context, we proposed RAVEN, a recommender heuristic strategy that complements our previous work on EPL by producing recommendations of how exception handling violations may be repaired. More specifically, a recommendation produced by RAVEN is a sequence of modifications to be performed in the call-chain of a method in order to repair a given violation. The RAVEN strategy comprises three steps. In the first step, the solution space where RAVEN searches its recommendations is built. This solution space is built with information extracted from the source code of the system and from its exception handling policy specification. It is worth noting that the availability of a policy specification is not mandatory for the proper functioning of the RAVEN strategy. In the second step, the solution space is traversed for constructing valid recommendations. Finally, in the third step, the valid recommendations are ranked and presented for developers.

We implemented RAVEN for Java programs and assessed its effectiveness in the context of three open-source projects. Effectiveness was assessed in terms of RAVEN's ability to produce recommendations able to actually repair violations in exception handling. The results show RAVEN is effective even when no explicit policy specification is available: it provides recommendations able to repair violations in 69% of the cases. Results also show RAVEN's effectiveness is significantly improved when explicit policy specifications are used: it produces recommendations able to repair violations in 97% of the cases. Moreover, results show it is possible to improve RAVEN's effectiveness with non-exhaustive and not overly detailed policy specifications. As a general conclusion, the results indicate development teams may promptly benefit from RAVEN even if they do not explicitly specify their exception handling policies.

REFERENCES

[1] E. Barbosa, A. Garcia, M. Robillard, and B. Jakobus. 2015. Enforcing Exception Handling Policies with a Domain-Specific Language. *IEEE Transactions on Software Engineering* PP, 99 (12 2015), 1–1.
[2] Eiji Adachi Barbosa, Alessandro Garcia, and Simone D. J. Barbosa. 2014. Categorizing Faults in Exception Handling: A Study of Open Source Projects. In *Proceedings of the XXVIII Brazilian Symposium on Software Engineering (SBES'14).*
[3] N Cacho, Eiji Adachi Barbosa, Juliana Araújo, Frederico Pranto, Alessandro Garcia, Thiago César, Artur Cassio, Eliezio Soares, Thomas Filipe, and Israel Garcia. 2014. How Does Exception Handling Behavior Evolve? An Exploratory Study in Java and C# Applications. In *Proceedings of the 30th International Conference on Software Maintenance and Evolution.*
[4] Felipe Ebert, Fernando Castor, and Alexander Serebrenik. 2015. An exploratory study on exception handling bugs in Java programs. *Journal of Systems and Software* 106 (2015), 82–101.
[5] John B. Goodenough. 1975. Exception handling: issues and a proposed notation. *Commun. ACM* 18, 12 (1975), 683.

*This paper was published in the IEEE Transactions on Software Engineering in June 2017. DOI: http://doi.org/10.1109/TSE.2017.2716925

ICSE '18, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5638-1/18/05.
https://doi.org/10.1145/3180155.3182539