August 15, 2015

Dear Bram and Alex,

Please find attached an extended version of our SANER’15 paper: “JCHARMING: A Bug Reproduction Approach Using Crash Traces and Directed Model Checking,” by Mathieu Nayrolles, Abdelwahab Hamou-Lhadj, Sofiène Tahar, and Alf Larsson.

The extended version is titled: “A Bug Reproduction Approach Based on Directed Model Checking and Crash Traces” by the same authors.

The new contributions of the paper consist of what follows:

* We addressed all the comments raised by SANER’s reviewers (please see the annexed document with our responses to the reviewers’ comments)
* We validated JCHARMING on three additional systems, Hadoop, Mahoot, and ActiveMQ, bringing the total number of subject systems to 10. We would like to note that the new systems are larger than those used in the conference version of the paper. This addresses the comments of Reviewer 2.
* We added the analysis of three additional bugs, as pointed out by Reviewer 1.
* We added a section on JUnit Test Case Generation.
* We updated the related work section
* We updated the threats to validity section
* We updated the conclusion and future work section

We would like to thank you and SANER anonymous reviewers for the constructive comments. We hope that you will find everything to your satisfaction.

We look forward to hearing from you.

Best regards,

Abdelwahab Hamou-Lhadj, PhD, PEng,

OCUP, OCEB

Associate Professor, Undergraduate Program Director

Software Behaviour Analysis (SBA) Research Lab

Electrical and Computer Engineering

Concordia University

1455 de Maisonneuve West

Montreal, Quebec H3G 1M8 Canada

Office: S-EV5.213

Tel: +1.514.848.2424 ext. 7949

Email: [abdelw@ece.concordia.ca](mailto:abdelw@ece.concordia.ca)

**This document explains how we have addressed each of the comments made by SANER’s reviewers of the paper “JCHARMING: A Bug Reproduction Approach Using Crash Traces and Directed Model Checking”.**

**We have numbered each comment. Original comments from reviewers are in bold; our responses are in non-bold text.**

Reviewer 1:

**R1.1 Section IV, B - just before C - I understand why you do not look at the Java GUI and JDK, but I wonder what if the crash is an interaction issue? These types of issues are not detectable by your method?**

Wahab: I guess here it is a matter of scope. I will write something.

**R1.2 Section IV, C, just under figure 4 - If all the frames are corrupt, you perform full up model checking? How often does this happen?**

Wahab: This is a good point. Please add one or two lines what would happen if all the frames are corrupt.

**R1.3. Section IV, E - This reader wonders if the user is not able to provide some information (which is why we want to do this in an automated fashion), can we trust a threshold that they offer? Will they be informed enough to know what 80% of the frames means?**

**R1.4. Section V, A, just below Table I - I understand that you used a random technique and it selected 7 bugs from one system, and one from others. It would have been nice to have at least 2 from each system though, for comparison.**

We tried in earlier version to describe two bugs for each system but it turned out to take a huge amount of space. In this extended version, we preferred to describe three new bugs from the newly added systems, namely Hadoop, Mahout, and ActiveMQ. We believe this gives a good coverage of the various bugs. I hope this addresses the reviewer’s concern.

**R1.5. able II - I wonder how long it took for the no scenarios to run. Did it take a very long time to realize that it did not work?**

Wahab: Please let me know how long.

**R1.6. Just before Section VI, discussion of Struts - could this type of thing not happen often? How can you address it?**

Wahab: Please add a paragraph to discuss this point.

**R1.7. Just before Section VI - Why not apply your approach to the "proper crash trace" and see if it did reproduce it?**

For the bugs that were successfully reproduced, we applied the approach to the proper crash trace. However, in practice, unfortunately, we do not always have proper crash traces, so we preferred to show an example where we have corrupt frames and how partial traces can be dealt with.

--------------

Reviewer 2:

**R2.1. Page 2, 1. col  
- coverage of automated test -> coverage of automated tests  
- 'pass' and 'fail' path which -> 'pass' and 'fail' path, which   
Page 2, 2nd col  
- on-field bug -> on-field bugs  
- core dump and try -> core dump and tries  
- an in-memory which -> an in-memory, which (read up on the use of 'which'  
and the difference between w and w/o comma, and check the paper )  
- own JVM allows to capture -> own JVM allows one to capture  
Page 3, 1st col  
- Except for STAR, the majority: rephrase (leave out either 'except for'  
or 'majority')  
- when there exists a sequence of state transition: whether or not it is  
a "sequence" depends on the logic formula  
Page 3, 2nd col  
- superior -> larger**

Fixed. We would like to thank the reviewer for proof reading the paper. It is much appreciated. We fixed all the typos raised by the reviewer. We also proof read again and again.

**R2.2. Page 4, 2nd col  
- address the problem of nested exceptions: namely?**

We added a sentence explaining what we mean by this. Basically, ….

**R2.3. Page 5, 1st col.  
- for our directed: missing noun  
- does not necessary -> does not necessarily  
- allows to reach -> allows one to reach**

Fixed.

Page 5, 2nd col  
- Bytecode -> byte code  
- in Fig. 4, what is the entry?  
- did you take WALA's slicer as-is or did you have to extend it?

Page 6, 1st col  
- Fig.5: do you call WALA's slicer?  
- Fig.5: the two arrows in line 7 are fairly non-standard  
- Fig.5: typo in 'slide'  
- states transitions -> state transitions (everywhere)  
Page 6, 2nd col  
- why does serialization help with optimization?  
Page 7, 1st col  
- Can we use crash traces ... to reproduce on-field bugs: why should  
the answer be anything other than 'yes'? Be more specific.  
Page 8, 1st col  
- fonts in line 29   
Page 9, 1st col  
- lines and the replaced -> lines and then replaced  
Page 9, 2nd col  
- We believe that JCHARMING ..: what prevents you from checking it?

>>>- when there exists a sequence of state transition: whether or not it is

a "sequence" depends on the logic formula

Wahab: This comment is not clear to me. Let's not address it.

Page 4, 2nd col

>>- address the problem of nested exceptions: namely?

Wahab: Please add a sentence here.

>>- in Fig. 4, what is the entry?

Wahab: - I think the reviewer means Fig 2 of SANER paper, which is Fig. 4 in the journal version.

- My suggestion is to change the figure to have the crash trace as the entry, but also add the system as an entry to the model checker.

- There are two typos in the figure: "Fillaure" --> "Failure" and "Cleaned Crash Trace" --> "Clean Crash Trace"

>>- did you take WALA's slicer as-is or did you have to extend it?

Wahab: Please add a sentence clarifying this.

Page 6, 1st col

>>- Fig.5: do you call WALA's slicer?

Wahab: Can you please address this?

>>- Fig.5: the two arrows in line 7 are fairly non-standard

Wahab: I don't understand what the reviewers means here. I think he is referring to Fig. 4 of SANER paper, but the arrows are OK.

>>- Fig.5: typo in 'slide'

Wahab: Please check this. Again, you need to figure out which figure the reviewer is referring to. He mixed up the numbers.

Page 6, 2nd col

>>- why does serialization help with optimization?

Wahab: Can you comment on this. Otherwise, we need to remove the term optimization and replace it with something else.

Page 7, 1st col

>>- Can we use crash traces ... to reproduce on-field bugs: why should

the answer be anything other than 'yes'? Be more specific.

Wahab: I will add something later.

-Reviewer 3:

In section IV.D, it would be good to have a bit more description (maybe an example) of how the backwards slice information is used to guide the model checker, since this is the central idea. The 7 lines quite formal description does not seem adequate (and is IMO not so easy to follow).  
  
One thing that needs to be fixed is "Figure 5", the algorithm for computing the union of slices (btw., it's not really a figure). There are some inconsistencies (or bugs) in there:  
- CurrentFrames = frames[i] when i has not been initialized  
- the "for" loop is a bit strange. It's neither Pascal nor C/Java like, its semantics are not completely clear  
- shouldn't the slice go from CurrentFrame to Frame[i+offset]?  
- "Slide" should be "Slice"  
- I wonder if the description cannot be simplified. I mean, the branch/offset is just for ignoring corrupted entries, right?  
  
Minor issues:  
- p. 2: "try (with evolutionary algorithms) to..." => tries  
- p. 3: "there exists a sequence of states transition x leading..." => sequence of states or sequence of transitions, or both?  
- p. 5: "for our directed, ..." => a noun missing?  
- p. 5, introduction of bslice: my impression was that sometimes m and n are exchanged  
- p. 6: "that need to be" => needs  
- p. 6: "current sequence of states transitions x" => see above  
- p. 6: "satisfies a set a property" => ?  
- p. 7: Case studies: The description of subject systems could be shortened.  
- p. 8: the example stack trace for bug #311 could be shortened to the first half, since the second half is thrown away anyway. It does not seem adequate to spend 1/3 page for that.  
  
In summary, the paper is well written and contains an interesting and effective contribution to bug reproduction.