Issue2UnitTest: Automatically Generating Unit Tests from Issue Reports

July 2026

Software systems evolve permanently, to align with changing requirements, fix bugs, incorporate additional features, or remove deprecated ones. An essential step in accelerating software evolution consists in automatically performing changes based on the *issue reports* provided by the users. These are usually written in natural language and combine unstructured text with code fragments in potentially different programming languages, a format that poses challenges in automatically extracting the user intent. Consider, for example, the issue report from Figure 1. It represents a fragment of a real-world bug report from Amazon[1], which describes an unsoundness in the Z3 Satisfiability Modulo Theories (SMT) solver's [1] string theory and the commit that introduced it. The input formula that exposes the bug is presented in SMT-LIB format[2], while the instructions performed by the user are expressed as command line operations.

Recent advances in Large Language Models (LLMs) have shown that these are particularly suited for transforming natural language specifications into formal postconditions [2] or for automatically generating issue reproducing tests [3, 4]. While essential for debugging, these state-of-the-art technique are not sufficient for automated program repair (APR), as their test oracles usually do not include any information about the location of the bug. For Figure 1, Issue2Test [4] could in principle generate a test that reproduces the reported bug, i.e., invokes Z3 (the version from commit 93427f1) on the SMT formula provided by the user with the parameter 'model_validate' set to true and asserts that it produces an invalid model. However, such a test treats the solver as a black-box and cannot identify that the length axiomatization from Z3's sequence theory (which is also used to encode strings) is incorrect, i.e., the bug is localized within the method 'axioms::mk_len', which needs to be repaired.

The goal of this project is to automate this step, by generating unit tests from issue reports using LLMs. Assuming that the source code of the system under test is always available, we will extend the approach (and the implementation) of Issue2Test [4] to generate method-level tests that expose the reported issues and guide the developers in fixing them. In particular, we will focus on automatically

[1] https://github.com/Z3Prover/z3/issues/8572
[2] https://smt-lib.org/

On this example, z3 produces an incorrect model:

```
(set-logic QF_S)
(declare-fun a () String)
(assert (= (str.substr (str.++ a a "0") 0 6) (str.++ a "a")))
(check-sat)
(get-model)
```

Here's the behavior

```
> z3 bug.smt2 model_validate=true
sat
(error "line 4 column 10: an invalid model was generated")
(
  (define-fun a () String
    "aAB")
)
```

This is related to issue #8023 (last example).
This bug seems to have been introduced with commit 93427f1.

Figure 1: Fragment of the user bug report for Z3's soundness issue #8572.

constructing unit tests that: (1) reproduce the bugs found by the users and (2) encode the correct, expected behavior.

To achieve this goal, we will perform the following steps:

- We will extend Issue2Test to also generate tests that capture the correct behavior of the system under test.
- We will then design a technique to transform these tests (and the original issue-reproducing tests) into unit tests for the methods that are the most likely to contain the incorrect code.
- We will evaluate our approach on closed issues from Z3.

As a possible extension, we will investigate how our generated unit tests can be used for automated program repair, e.g., by integrating our approach with existing APR tools.

References

[1] Leonardo De Moura and Nikolaj Bjørner. Z3: An Efficient SMT Solver. In *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS'08/ETAPS'08, page 337–340, Berlin, Heidelberg, 2008. Springer-Verlag.

[2] Madeline Endres, Sarah Fakhoury, Saikat Chakraborty, and Shuvendu K. Lahiri. Can Large Language Models Transform Natural Language Intent into Formal Method Postconditions? *Proc. ACM Softw. Eng.*, 1(FSE), July 2024.

[3] Konstantinos Kitsios, Marco Castelluccio, and Alberto Bacchelli. Automated Generation of Issue-Reproducing Tests by Combining LLMs and Search-Based Testing. In *2025 40th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 1982–1994, 2025.

[4] Noor Nashid, Islem Bouzenia, Michael Pradel, and Ali Mesbah. Issue2Test: Generating Reproducing Test Cases from Issue Reports. In *International Conference on Software Engineering*, 2026.