LEVERAGING E2E TEST CONTEXT FOR LLM-ENHANCED TEST CASES

Author: Mattheo de Wit (M.C.A.deWit@student.tudelft.nl)
Supervisor: Amir Deljouyi, Responsible professor: Andy Zaidman

1 - Background

- **Automated unit-test generating frameworks** such as Search-Based Software testing (SBST) optimise for criteria like code coverage and fault detection, but often generate tests that are **hard to understand** due to lack of meaningful test data and comments, as shown in Figure 1.
- **UTGen** aims to improve the understandability of these test cases by conducting **Large Language Models** (LLMs) in several post-processing steps of EvoSuite, to replace test data with human-readable strings and place explanatory comments [1]. This lacks realism and meaningfulness however due to limited context (Figure 3).
- **MicroTestCarver (MTC)** uses **capture/replay** techniques, storing runtime data and objects from end-to-end (E2E) tests as **trace logs**, to be able to generate meaningful and realistic test data. This approach falls short however on the readability of generated tests, not being able to provide explanatory comments [2].
- This research introduces **UTGen+,** combining capture/replay with LLM-enhanced SBST in an attempt to increase the **understandability** and **relevancy** of generated test cases.

```java
@Test
public void testGetDescriptionReturningNonEmptyString() {
    Person person = new Person();
    person.setDescription("Xt1Byl64')^>");
    String description = person.getDescription();
    assertEquals("Xt1Byl64')^>", description);
}
```

Figure 1: Test generated by EvoSuite

2 - Research Question

RQ: How does **integrating contextual data** from the trace logs of end-to-end tests into the **LLM-enhanced SBST approach in UTGen** affect the **understandability** and **relevancy** of generated test cases?

Test elements of interest: Comments, identifiers and test data

Hypothesis: Due to the provided context of how methods are *acutally* used, the generated tests will more closely represent use case scenario's and will therefore be more understandable and relevant as compared to UTGen.

```
1| Now follows a runtime example of the getDescription() method:
2| Suppose this 1st example object:
3| Person person{name: "John Doe", description: "Cool Developer", ...}

4| Example usage on this object:
5| person.getDescription() //Returned: "Cool Developer"
```

Figure 2: Contextual data as included in UTGen+'s prompts

3 - Methodology and Setup

Implementation

UTGen+ consists of the following 5 phases, to enhance the LLM prompts with contextual data from trace logs:

1) Traces are **parsed** into a classmap structure, to allow for easy method-call access during generation.
2) Method statements from the generated test case of EvoSuite are **matched** with examples in traces.
3) **Context preparation**: Arguments and Return Values of matched examples are converted to String representation.
4) Pompt preparation: Context strings and calling objects are enhanced with **human-like text** for prompt inclusion. The result of this is visible in Figure 2, showing how the context will be presented to the LLM.
5) Prompts are **executed**, and their respective responses are **parsed** by the original UTGen implementation.

Steps 2 to 5 are repeated for every generated test case, during the Test Data Refinement and Post-Processing stages of UTGen. In this research, we have exclusively used **GPT-4o** as our LLM of choice.

User evaluation

We conducted a user study with 9 participants, aiming to verify our hypothesis. Over 2 rounds of 3-4 questions, participants were asked to compare test cases of *UTGen+*, *UTGen* and *EvoSuite*, and rate the **naturalness** and **relevancy** of the comments, identifiers and test data using a Likert scale. For the relevancy round, they were first provided with additional background information, allowing them to asses whether the test elements are meaningful in context.

4 - Results

Comments: The analysis of naturalness for comments reveals minimal distinction between UTGen+ and Original UTGen (**-7.8%** in mean), as can be seen in Figure 5. Figure 6 displays a slight decrease but a bit more consistency in relevancy rating (**-3.9%** in mean). From the qualitative answers we get the indication these **decreases** could be explained by perceived formatting differens, although we have not quantified this.

Identifiers: For identifiers, we have observed minimal differences in perceived naturalness (**-1.2%**) and relevancy (**-5.9%**). Although we expected this to increase, the relatively low complexity of selected test cases might have contributed to UTGen being able to match UTGen+ in this regard.

Test data: In terms of test data, UTGen+ demonstrates a marginal improvement in naturalness compared to Original UTGen, although the distributions are nearly identical (**+5.2%**). The relevancy of test data saw **substantial increase** of **+25.0%** in mean rating, highlighting UTGen+'s potential in generating contextually relevant test data.

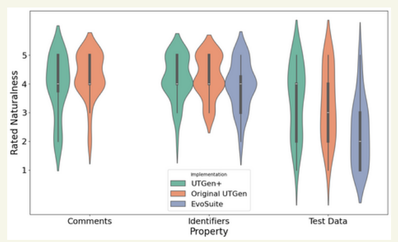

Figure 5: Violin graph from *naturalness* results

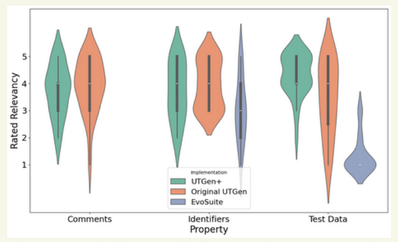

Figure 6: Violin graph from *relevancy* results

5 - Conclusion

- Our findings reveal that while UTGen+ showed a **marginal decrease** in the naturalness and relevancy of **comments** and **identifiers**, it drastically **improved** the quality of the **test data.** More specific, the relevancy of test data generated by UTGen+ was notably higher compared to that produced by Original UTGen.
- This suggests that the inclusion of contextual data from trace logs can indeed enrich the content of generated test cases, making them more applicable and valuable in real-world testing scenarios.
- **This partially confirms our initial hypothesis, that more contextually informed test generation could bridge existing gaps in automated test case development.**

6 - Limitations & Future Work

Despite encouraging results, we identified several limitations:

- The study's scope was **restricted** to a select number of software projects and utilized only a single LLM configuration (GPT-4o with SUA-model).
- Selected test cases in the user evaluation were relatively **simple**, meaning they might not have fully maximized on the capabilities of UTGen+
- The limited sample size and the **similarity** of participant backgrounds, mean that the findings should be interpreted cautiously regarding their **generalizibility**.

We recommend the following areas to conduct further research:

- Exploring **different configurations** of LLMs and extending the approach to include more diverse datasets and software environments
- Investigating the effects of including examples based on **similar method arguments** rather than random choice
- **Extending** the types of **statements** analyzed, also including external leaf methods and constructor calls, and potentially other runtime data.

These enhancement could improve the performance of UTGen+, and provide deeper insights in the opportunities of integrating trace logs in LLM-enhanced SBST.

References:

[1] A. Deljouyi. Understandable test generation through capture/replay and llms (Lisbon, 2024)
[2] A. Deljouyi and A.E. Zaidman. Generating understandable unit tests through end-to-end test scenario carving (SCAM 2023)

```java
@Test
public void testGetDescriptionReturningNonEmptyString() {
    //Given: A new Person instance with a description set
    Person person = new Person();
    person.setDescription("TestDescription123");

    //When: The description is retrieved from the person object
    String description = person.getDescription();

    //Then: The retrieved description should match the one that was set
    assertEquals("TestDescription123", description);
}
```

Figure 3: Test improved by UTGen

```java
@Test
public void testGetDescriptionReturningNonEmptyString() {
    //Given: A new Person with description set to "Handsome Programmer"
    Person person = new Person();
    person.setDescription("Handsome Programmer");

    //When: Getting the description of the person
    String description = person.getDescription();

    //Then: The description should match the expected description
    assertEquals("Handsome Programmer", description);
}
```

Figure 4: Test improved by UTGen+