# exLong: Generating Exceptional Behavior Tests with Large Language Models

Jiyang Zhang[1], Yu Liu[1], **Pengyu Nie**[2], Junyi Jessy Li[1], and Milos Gligoric[1]

[1] The University of Texas at Austin

[2] University of Waterloo

ICSE 2025

partially supported by National Science Foundation, SIMONS FOUNDATION, CISCO

# Exceptional Behavior Tests (EBTs)

- Exceptions: thrown during program execution if an unwanted event happens
- Exceptional behavior tests: check that the code detects unwanted events and throws appropriate exceptions

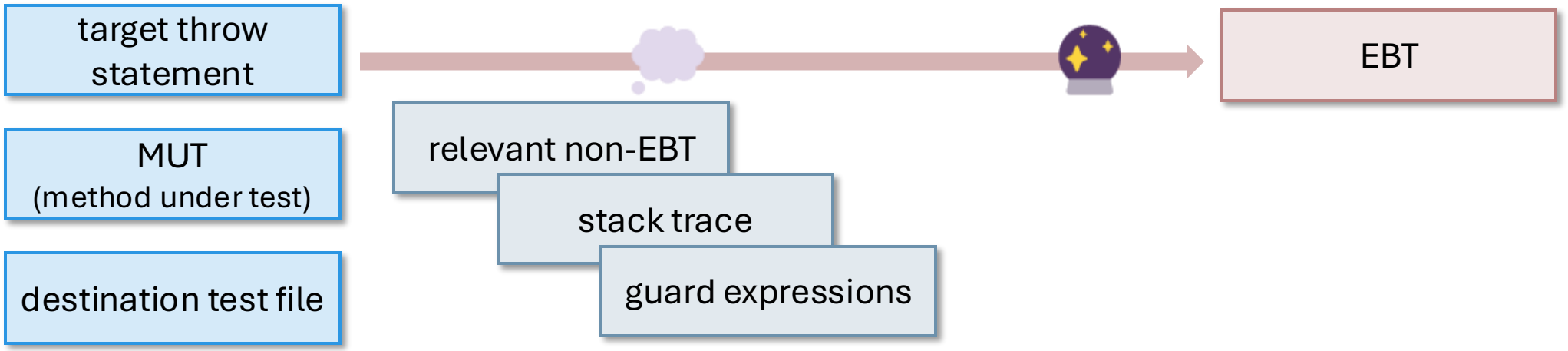
# Exceptions Deserve More Test Coverage!

- EBTs check critical error-handling code
- Developers are not writing sufficient EBTs
  - ~40% throw statements covered
  - Developers usually focus on “happy paths” when writing tests [1][2]
- No existing tools specifically help developers write EBTs
- LLMs are struggling to generate correct EBTs

[1] Dalton, Francisco, et al. "Is exceptional behavior testing an exception? an empirical assessment using java automated tests." Proceedings of the 24th International Conference on Evaluation and Assessment in Software Engineering. 2020.
[2] Di Bernardo, Rafael, et al. "Agile testing of exceptional behavior." 2011 25th Brazilian Symposium on Software Engineering. IEEE, 2011.

# Our Solution: LLM + Program Analyses

- exLong
  - base LLM: Code Llama
  - instruction-tuned to reason about context collected via program analyses



# Relevant Non-EBT

- Non exception behavior tests that
  (1) invoke the given MUT, or
  (2) are in the same destination test file
- Provides example on how to prepare the test inputs & the expected coding style

MUT

```
public static int[] qualityScores(
  final Fastq fastq,
  final int[] qualityScores
) {
  ...
}
```

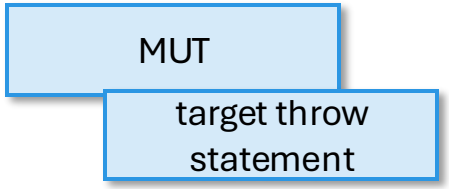
relevant non-EBT

```
@Test
public void testQualityScoresIntArray() {
  int[] qualityScores = new int[4];
  FastqTools.qualityScores(builder.build(), qualityScores);
  for (int i = 0; i < 4; i++) {
    Assert.assertTrue(qualityScores[i] != 0);
  }
}
```

# Stack Trace

- The sequence of method invocations that start from the given MUT and lead to the target throw statement
- Provides insights on what other methods may be involved during execution

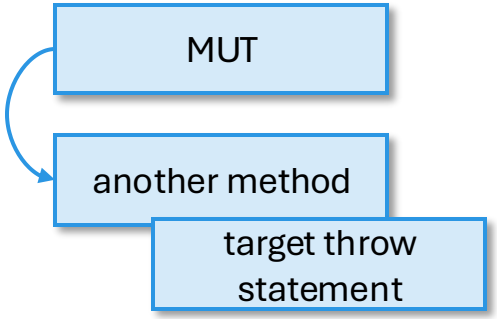
*target throw statement is in the MUT*




stack trace

```
qualityScores(FastqTools.java)
```

*target throw statement is in another method transitively invoked by the MUT*




```
searchTerm(SearchCommandParser.java)
parse(SearchCommandParserTest.java)
...
```

# Guard Expressions

- The logical formula representing the constraints on the symbolic variables that must be true to follow the stack trace
- Provides direct guidance on how to trigger the exception

MUT

```
public static int[] qualityScores(final Fastq fastq, final int[] qualityScores) {
  if (fastq == null) {
    throw new IllegalArgumentException("fastq must not be null");
  }
  if (qualityScores == null) {
    throw new IllegalArgumentException("qualityScores must not be null");
  }
  int size = fastq.getQuality().length();
  if (qualityScores.length != size) {
    throw new IllegalArgumentException(
      "qualityScores must be the same length as the FASTQ sequence quality");
    }
  FastqVariant variant = fastq.getVariant();
  for (int i = 0; i < size; i++) {
    char c = fastq.getQuality().charAt(i);
    qualityScores[i] = variant.qualityScore(c);
  }
  return qualityScores;
}
```

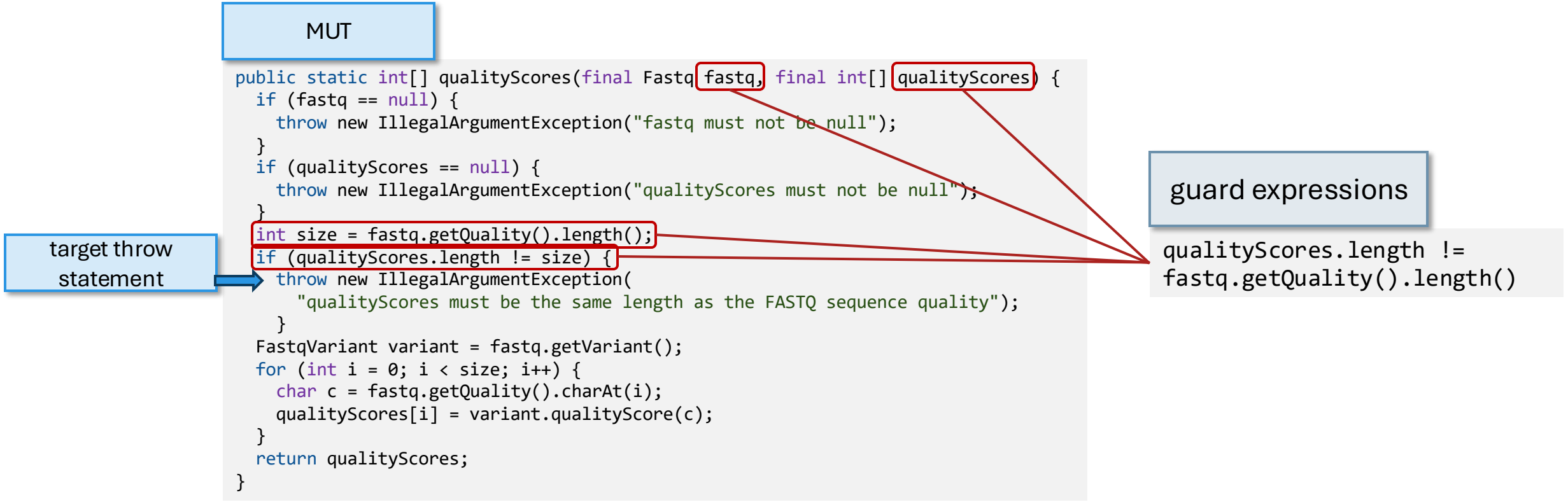
target throw statement

guard expressions

```
qualityScores.length != fastq.getQuality().length()
```

# Guard Expressions

- The logical formula representing the constraints on the symbolic variables that must be true to follow the stack trace
- Provides direct guidance on how to trigger the exception



# Use Case: Developer-Oriented

Goal: Generate an EBT for one specific target throw statement

Generate a test in ${destination test file} that covers ${target throw statement} from ${MUT}

performing program analyses

```
@Test
public void testQualityScoresIntArrayNullFastq() {
  try {
    FastqTools.qualityScores(null, new int[0]);
    Assert.fail(
      "qualityScores(null, int[]) expected IllegalArgumentException");
  } catch (IllegalArgumentException e) { }
}
```

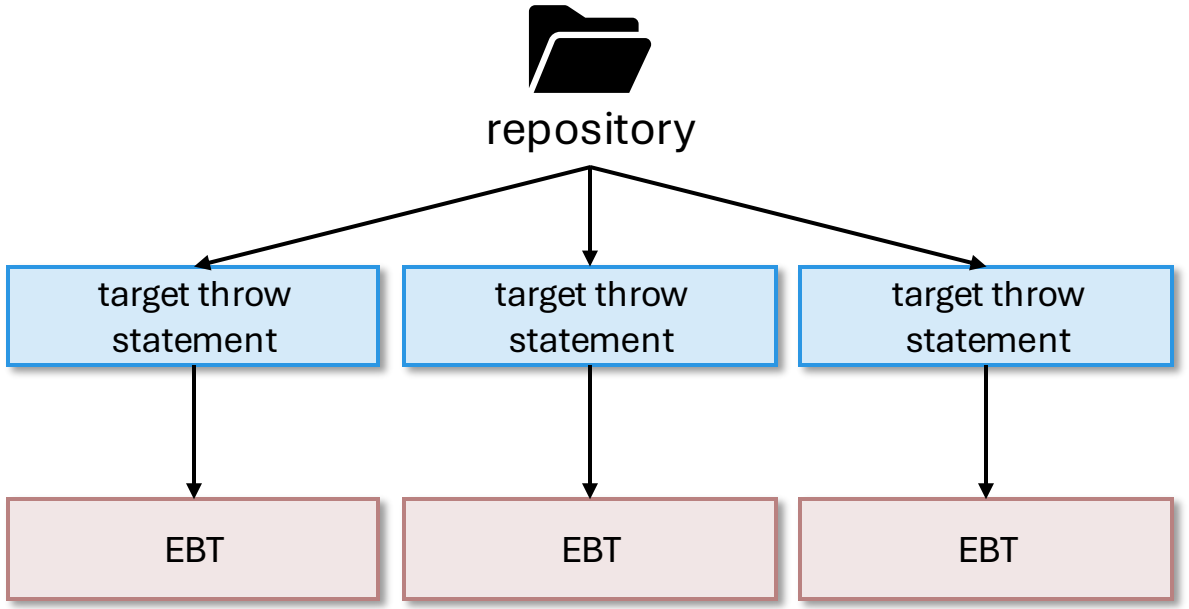
# Use Case: Machine-Oriented

Goal: Generate EBTs to cover as many throw statements as possible

searching for throw statements

performing program analyses

generating EBT



# Evaluation Dataset

- Open-source Java repositories on GitHub
  - based on CodeSearchNet's list of repositories and cross-project split
  - compile successfully and do not contain test failures
  - have at least one EBT

| | #Repos | #Tests | #EBTs |
|---|---|---|---|
| All | 562 | 111,230 | 12,574 |
| Train | 501 | 100,030 | 11,182 |
| Valid | 29 | 5,298 | 550 |
| Eval | 32 | 5,902 | 842 |

instruction-tuning exLong

evaluation on the two use cases

# Results: Developer-Oriented

- 434 EBTs, 278 throw statements (covered by at least one non-EBT), 41 exception types

similarity metrics
(comparing against deverloper-written ones)

functional-correctness metrics

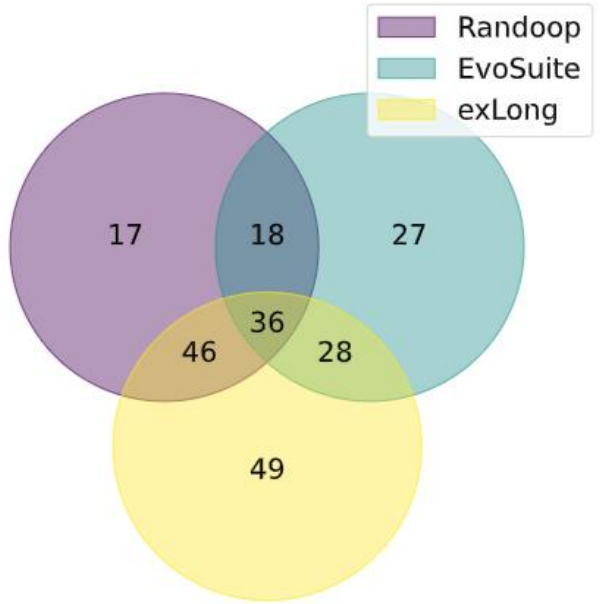
| | xMatch | CodeBLEU | Compilable% | Runnable% | ThrowCov% |
|---|---|---|---|---|---|
| GPT-3.5 | 14.98 | 64.28 | 75.12 | 61.29 | 48.39 |
| CAT-LM | 9.83 | 59.79 | 71.83 | 36.64 | 30.03 |
| exLong w/o context | 12.06 | 60.62 | 61.52 | 46.70 | 38.25 |
| exLong | **19.05** | **67.49** | **82.10** | **67.63** | **59.45** |

# Results: Machine-Oriented

- 649 throw statements, 81 exception types

random / search-based test generation tools

| | ThrowCov% |
|---|---|
| Randoop | 21.87 |
| EvoSuite | 20.37 |
| exLong | **29.72** |



# Conclusions

- exLong: LLM + program analyses for generating exceptional behavior tests
- Sent 7 PRs with 35 EBTs generated by exLong; 4 accepted, 3 pending
- Tool paper at FSE'25

- Code: https://github.com/EngineeringSoftware/exlong
- Data: https://huggingface.co/datasets/EngineeringSoftware/exLong-dataset
- Model: https://huggingface.co/EngineeringSoftware/exLong

Jiyang Zhang, Yu Liu, **Pengyu Nie**, Junyi Jessy Li, and Milos Gligoric

<jiyang.zhang@utexas.edu> <pynie@uwaterloo.ca>

# Backup Slides

# Results: exLong based on Proprietary LLM

| | xMatch | CodeBLEU | Compilable% | Runnable% | ThrowCov% |
|---|---|---|---|---|---|
| GPT-4o | 16.82 | 65.56 | 81.87 | 71.52 | 55.53 |
| exLong-GPT-4o | **17.74** | **66.77** | **82.49** | **75.35** | **64.75** |

# Case Study / Real-World Impact

- Prepare PRs with the EBTs generated by exLong for 9 actively-maintained repositories in the eval set
- For 2 repositories, the same EBTs were added by developer in a later commit (after we collected data)
- Submitted PRs to 7 repositories with 35 EBTs
  - 4 accepted; 1 of them merged within 30min
  - 3 pending

# Exceptions

- Exceptions: thrown during program execution if an unwanted event happens
  - e.g., invalid input, illegal state
  - “throw” in Java/C++, “raise” in Python

```
public static int[] qualityScores(final Fastq fastq, final int[] qualityScores) {
  if (fastq == null) {
    throw new IllegalArgumentException("fastq must not be null");
  }
  if (qualityScores == null) {
    throw new IllegalArgumentException("qualityScores must not be null");
  }
  int size = fastq.getQuality().length();
  if (qualityScores.length != size) {
    throw new IllegalArgumentException(
      "qualityScores must be the same length as the FASTQ sequence quality");
    }
  FastqVariant variant = fastq.getVariant();
  for (int i = 0; i < size; i++) {
    char c = fastq.getQuality().charAt(i);
    qualityScores[i] = variant.qualityScore(c);
  }
  return qualityScores;
}
```