Towards Observation Lakehouses: Living, Interactive Archives of Software Behavior

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

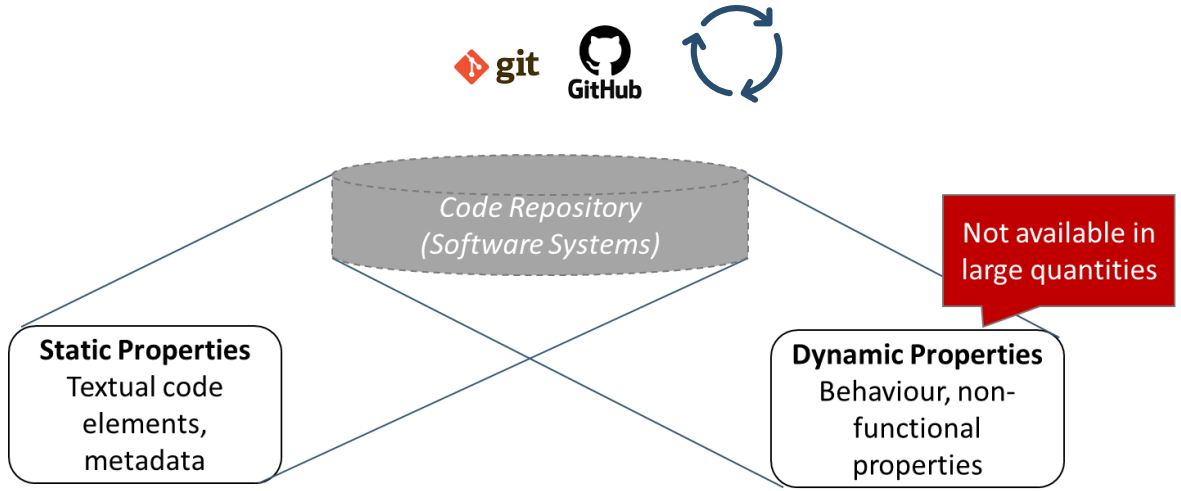
Marcus Kessel marcus.kessel@uni-mannheim.de (University of Mannheim, Software Engineering Group)

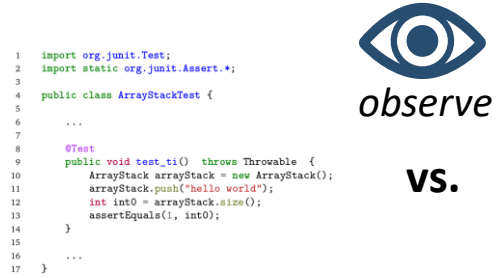
SANER'26 @ Limassol, Cyprus – Tool Demo Track, March 19th, 2026

Motivation – Obtaining Runtime Behavior at Scale

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

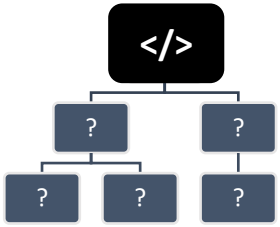
- **Problem**: LLMs (GenAI) typically limited to static artifacts at scale
  → limited availability of runtime behavior
- **Rice's theorem**: semantic (runtime) properties undecidable in general
  → need dynamic observation (software testing)
- **Current Gap**
  - (unit) testing (too little – only pass/fail signal)
  - tracing (too much – expensive)
- **Need**: runtime **behavior at scale, continuously** (!)
  → living, interactive archives of software behavior **complementing** syntactic artifacts





observe

vs.



xUnit

… too little (pass/fail signals)

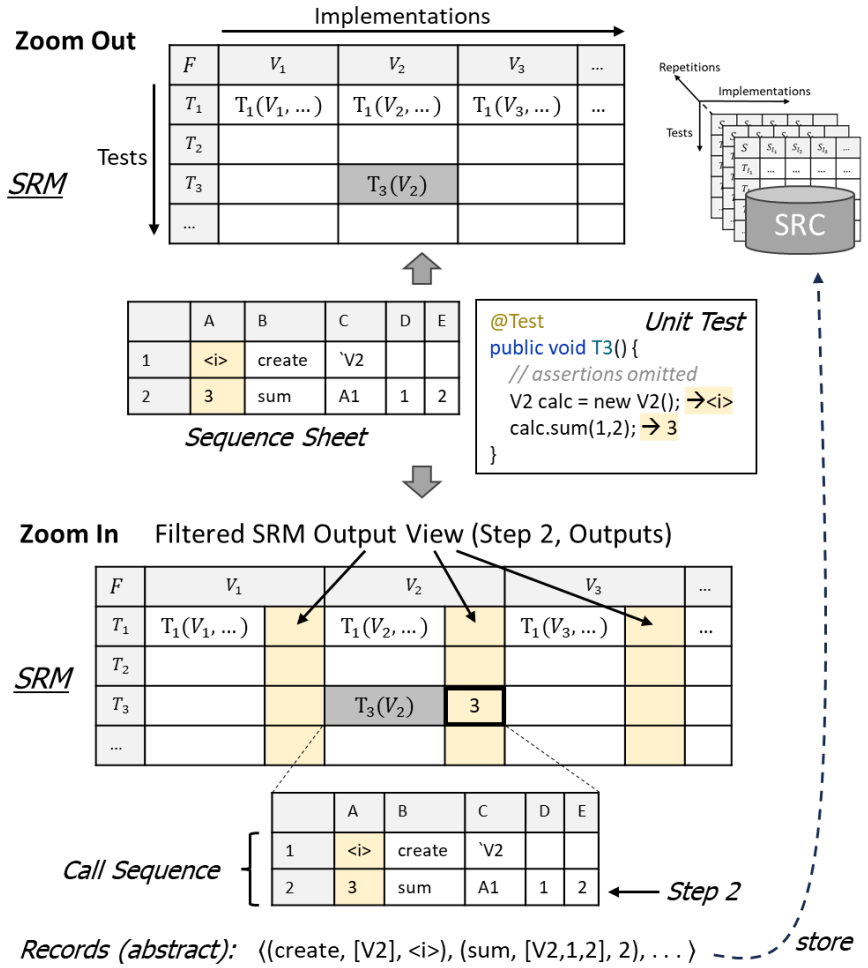
Tracing

… too much (detailed tracing)

Continual SRC – Tabular Representations

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- Sequence Sheets (Test Sequences)
  - stimulus/response encoding
  - invocation-centric: **⟨(op1, in1, out1), (op2, in2, out2), . . .⟩**
- SRMs (Stimulus-Response-Matrix)
  - tests × implementations comparison matrix
- SRCs (Stimulus-Response-Cube)
  - multi-dimensional SRMs across contexts (time, repetitions, envs etc.)
- "Living" (Continual) property
  - add rows/columns dynamically



Kessel, Marcus, and Colin Atkinson. "Morescient GAI for software engineering." ACM Transactions on Software Engineering and Methodology 34.5 (2025): 1-17.

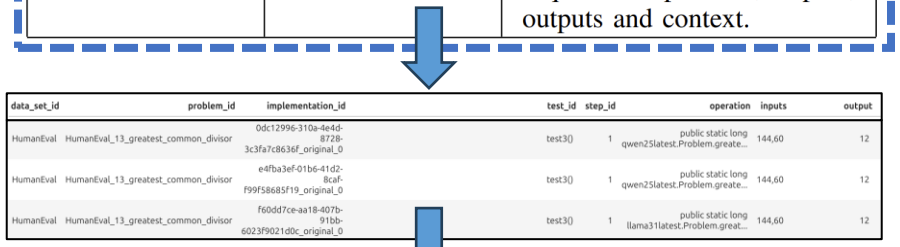
Observation Lakehouse – Architecture

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

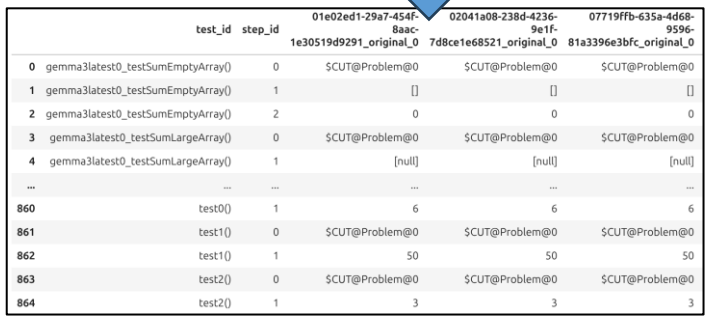
- **Observation Lakehouse:** Local store
  - "lake": "cheap" storage, flexible schema
  - "warehouse": SQL analytics (OLAP), ACID
  - Impl.: *Iceberg+Arrow+DuckDB+Parquet*
- **Invocation-centric: append-only table**
  - Stores invocations of test sequences
    - **⟨(op1, in1, out1), (op2, in2, out2), . . .⟩**
  - Full context awareness (environment, …)
- **Partitioning** avoids full table scans
  - Benchmarks: *data set / problem*
  - Practitioners: *repository / fully-qualified code unit (e.g., class)*

```
|     ├─ observations/
|     |   ├─ data/ # partitioned by data_set_id and problem_id
|     |   |   ├─ data_set_id=HumanEval/
|     |   |   |   └─ problem_id=HumanEval_0_has_close_elements/
|     |   |   |       └─ *.parquet # SRM for problem
|     |   |   └─ ...
```

Table	Contextual Keys	Core Columns & Description
code_ implementations	implementation_id	Stores source code and static metrics (e.g., LOC) for each implementation once.
tests	test_id	Stores the stimulus definition (e.g., a sequence sheet or a mined unit-test method) once.
observations	impl_id, test_id, step_id	The "fact" table storing each invocation step of a call sequence: operation, inputs, outputs and context.



***append-only** table*



*SRM view reconstructed (via **SQL**)*

*SELECT * FROM …*

Evaluation / Performance

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

Dataset

- **~8.6M observations** (= rows)
- **509 coding problems** (from *Cassano et al.'s* MultiPL-E benchmarks)
- **13.384 implementations** (N Java implementations for each problem)
- **95.154 tests** (a combination of human-written and LLM-generated tests)
- obtained with LASSO

Performance (single laptop)

- **Ingestion**
  - **Bulk import (≈ 7secs)**
  - **Rate: ≈ 155K records/second**
  - Denormalization strategy for performance
    - Duplicate data deliberately (principles)
- **Storage**: **51MiB (observations)**
  - high compression via Parquet format - *Run-length encoding (RLE)* etc.
- **Queries** ($\bar{x}$, over all problems): < 100ms
  - **SRM view reconstruction: ≈ 52ms**
  - **Behavioral clustering: ≈ 29ms**
  - **Full join (three tables): ≈ 90ms**

Cassano, Federico, et al. "Multipl-e: A scalable and polyglot approach to benchmarking neural code generation." *IEEE Transactions on Software Engineering* 49.7 (2023): 3675-3691.

Conclusion/Takeaways

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- **Practical realization** of obtaining runtime behavior at scale using a lakehouse stack
  - Initial performance promising (scalability)
  - Data-driven n-version assessment, behavioral clustering, consensus/historical oracles
- **Challenges**
  - Stimulus/response serialization
  - Need for equivalence functions (lossless?)
  - Query-dependent partitioning strategies
- **Ongoing, Future Work**
  - Integration with GenAI, Federation of lakehouses, …
  - *Mining invocations from unit tests*: Test-driver extensions (Java/JUnit5, python/pytest)
  - "*Replaying*" git repositories
  - *Behavioral Co-Versioning*: behavioral archive alongside git (accepted at FSE'26 IVR track)


2026 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)

Towards Observation Lakehouses: Living, Interactive Archives of Software Behavior

Marcus Kessel
University of Mannheim, Germany
marcus.kessel@uni-mannheim.de, 0000-0003-3088-2166

Preprint: https://arxiv.org/abs/2512.02795

Reproduction package: https://github.com/SoftwareObservatorium/observation-lakehouse

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

Thank you!

https://www.wim.uni-mannheim.de/atkinson/team/dr-marcus-kessel/
https://www.linkedin.com/in/marcus-kessel-307609312/