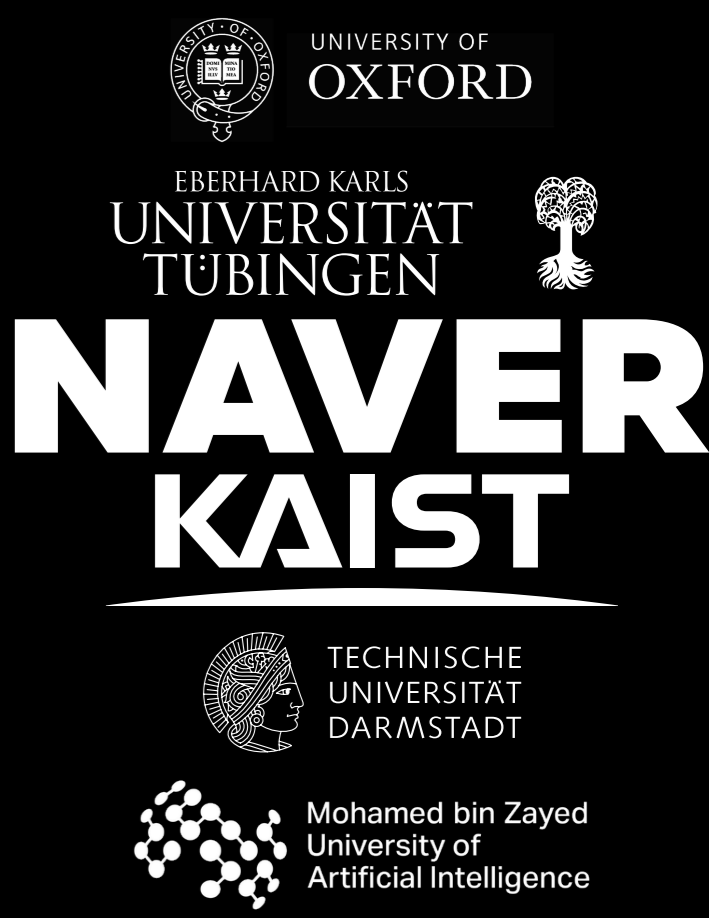


MASEval: Extending Multi-Agent Evaluation from Models to Systems

Cornelius Emde^{1,2,*} Alexander Rubinstein^{3,§} Anmol Goel^{1,4,§} Ahmed Heakl^{1,5,§}
Sangdoon Yun⁶ Seong Joon Oh^{1,3,7} Martin Gubri¹

§ Equal contribution. ¹ Parameter Lab ² University of Oxford ³ University of Tübingen ⁴ TU Darmstadt
⁵ MBZUAI ⁶ NAVER AI Lab ⁷ KAIST



MASEval is a Python library for evaluating whole multi-agent systems, not just models. It plugs into your frameworks, tools, and models with lightweight wrappers. You stay in control, MASEval does the boilerplate.

Problem: Evals Fix the System, Compare Models

- Framework and system impact on results is ignored.
- Different papers use different frameworks and systems, making results hard to compare.

Framework Choice Matters Much

Final scores variation across frameworks nearly matches variation across models when system is fixed.

14.2 pp mean range across models
12.4 pp mean range across systems

MASEval: From Models to Systems

Your own stack

Agent framework: smolagents, LangGraph, LlamaIndex
Models: OpenAI, Anthropic, Google
Multi-Agent System: roles, topology, tool wiring
Tools: search, code, APIs, databases
Environment: retail sandbox, web app, simulator
Benchmark tasks: MACS Travel, τ^2 -Bench, ConVerse
Evaluators: LLM judge, rules, task metrics

↓ thin adapters

setup → execute → evaluate → log → teardown

↓ structured reports

logs • metrics • per-agent traces • usage • costs

MASEval

What MASEval Provides

Trace-first evaluation

Metrics read full execution traces, not only final answers.

Separation of concerns

Task, environment, agent, and evaluator vary independently.

Efficient evaluation

Predict final score within ~2 pp saving 99% of tokens.

Bring your own stack

Use existing frameworks, models, tools, logging backends.

Multi-agent native

Per-agent traces and partial observability, not a flat harness.

Reproducible runs

Seeded provisioning for deterministic, repeatable evaluation.

Less Glue Code

83–91% less interface code to adopt a benchmark
35–57% less total benchmark implementation code

τ^2 -Bench: 5,011 fewer lines overall.
Total implementation drops from 8,804 to 3,793 lines of code.

Who Benefits

Researchers

Isolate system, model, topology, and orchestration effects.

Practitioners

Test real systems you ship, not a proxy.

Benchmark users

Run many benchmarks through one interface.

Benchmark builders

Less boilerplate; reusable infrastructure.

Try It

```
from maseval.benchmark import tau2
from maseval.interface.inference import LiteLLMModelAdapter
MODEL = "openrouter/qwen/qwen3.5-122b-a10b"

class MyTau2Benchmark(tau2.DefaultAgentTau2Benchmark):
    def get_model_adapter(self, model_id, **kwargs):
        return LiteLLMModelAdapter(model_id=model_id,
                                    seed=kwargs.get("seed"))

tau2.ensure_data_exists(domain="retail")
tasks = tau2.load_tasks("retail", limit=1) # single task
tau2.configure_model_ids(tasks, evaluator_model_id=MODEL)

benchmark = MyTau2Benchmark()
results = benchmark.run(tasks, agent_data={"model_id": MODEL})
print(tau2.compute_benchmark_metrics(results))
```

Comparison to Prior Work

Work	Compare Systems	Framework-agnostic	Per-agent traces	Stack-agnostic
MASEval	✓	✓	✓	✓
AnyAgent	×	△	×/△	✓
Inspect-AI/HAL	×	×/△	×	△
MARBLE	△	×	△	×/△
MLflow/Phoenix/TruLens	×	×	△	△
OpenCompass/GAIA/AgentBench	×	×	×	△

✓ yes △ partial × no