ON THE EXTERNAL CONCURRENCY OF CURRENT BDI FRAMEWORKS FOR MAS

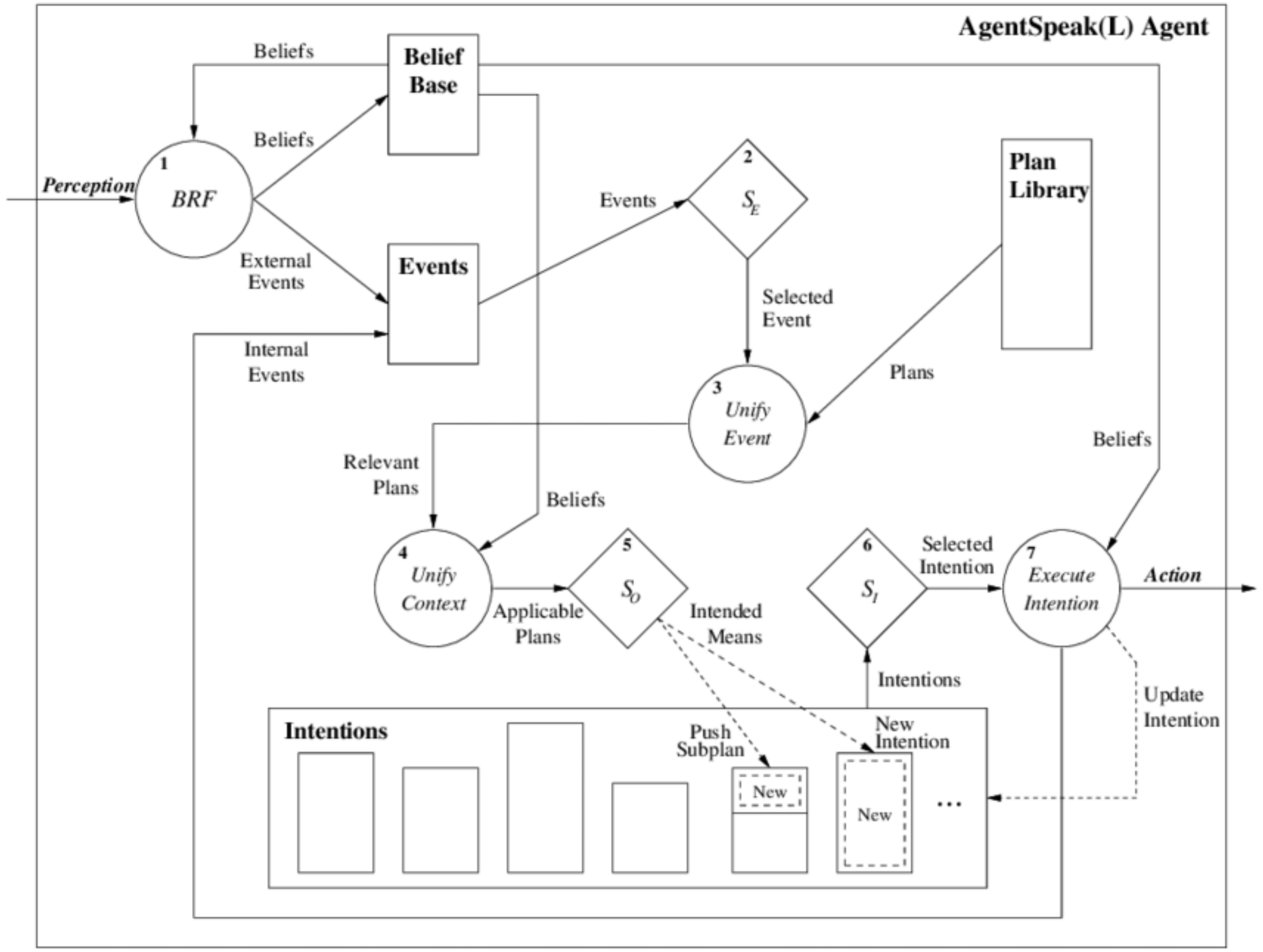
Martina Baiardi, Samuele Burattini, Giovanni Ciatto
Danilo Pianini, Alessandro Ricci, and Andrea Omicini

Department of Computer Science and Engineering (DISI)
Alma Mater Studiorum — Università di Bologna
Via dell'Università 50, 47522 Cesena (FC), Italy

CONTEXT

BDI AGENTS PROGRAMMING

- most famous semantics: AgentSpeak(L)
- most famous architecture (see picture)
- several implementations
  - focus on: Astra, GOAL, Jadex, JaKtA, Jason, PHIDIAS, SPADE-BDI



MOTIVATION

Insight: *the same architecture may be implemented in so many ways* (e.g., w.r.t. **concurrency**)

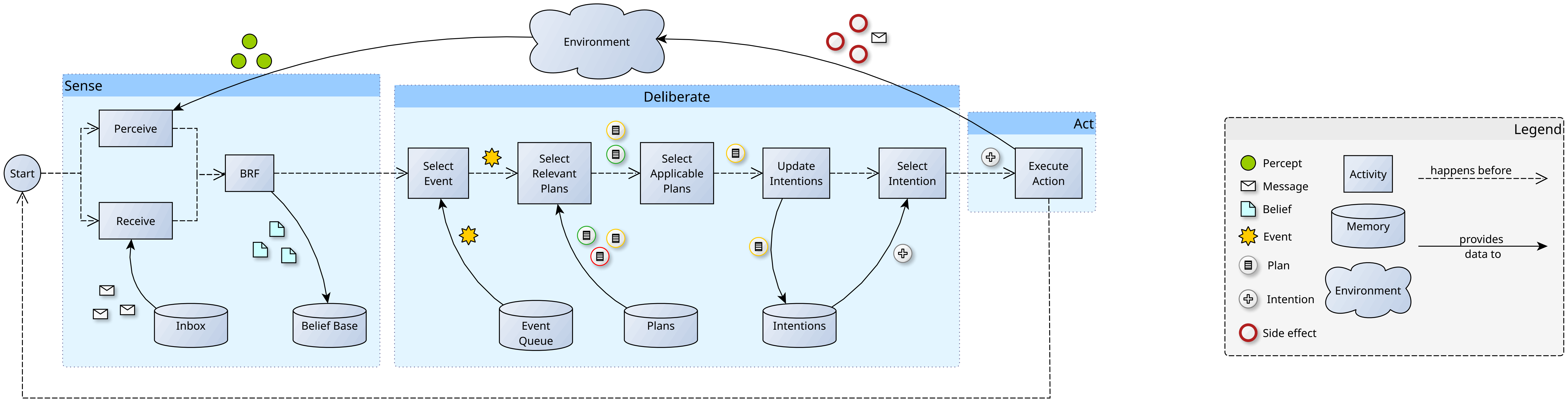
- semantics *unaffected*
- impact on practical properties such as efficiency & reproducibility

GOALS

1. Devise concurrency patterns from the state-of-the-practice
2. Classify BDI technologies accordingly

BACKGROUND

- Agents lifecycle, in general, is a control-loop
    - sense, *then* deliberate, *then* act, repeat
- BDI agents are more *complex*
    - e.g. sense implies collecting percepts, revising beliefs, etc.
    - e.g. deliberate implies selecting plans, updating intentions, etc.



WHICH CONCURRENCY?

We distinguish between internal and external concurrency

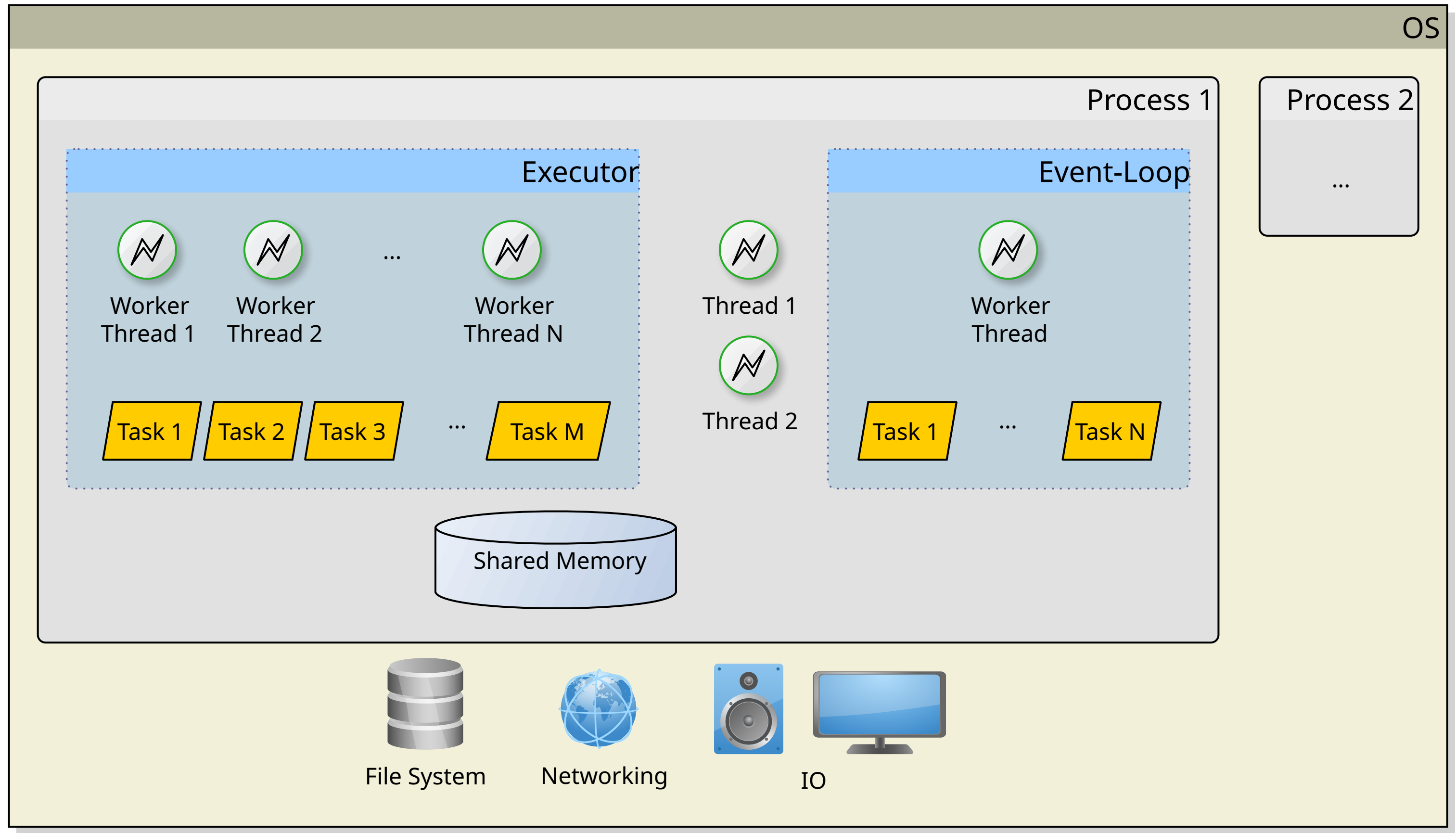
Internal concurrency ≈ how agents schedule intentions internally

External concurrency ≈ how agents' control-loops are scheduled by the underlying platform

WHICH CONCURRENCY ABSTRACTIONS?

IN PRACTICE, TECHNOLOGICAL PLATFORMS SUPPORT:

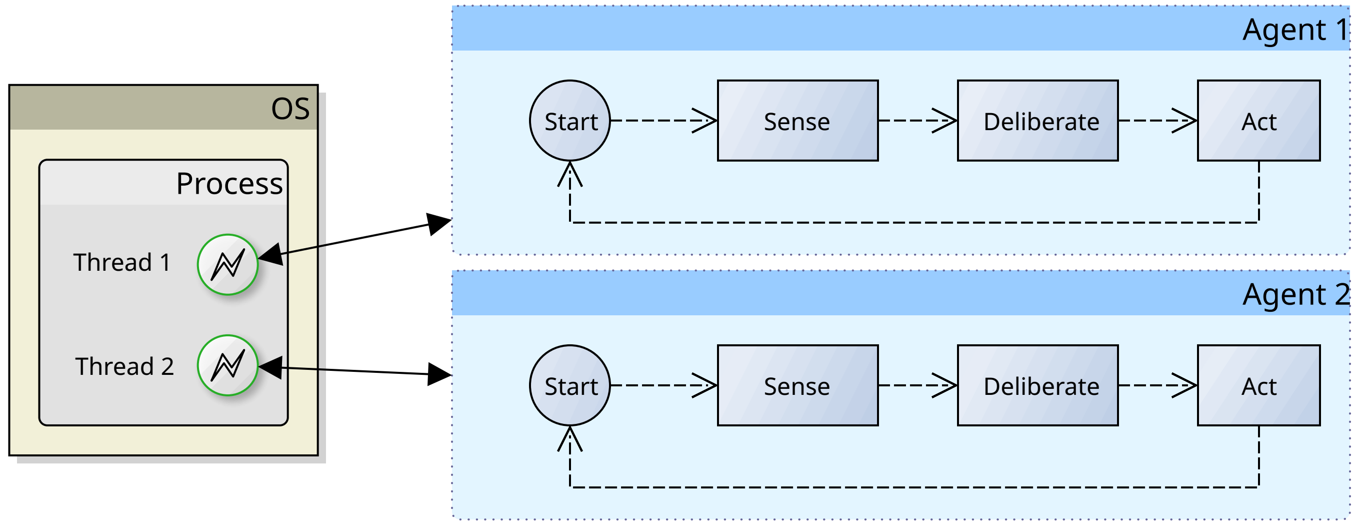
- Processes
- Threads
- Event Loops
- Executors



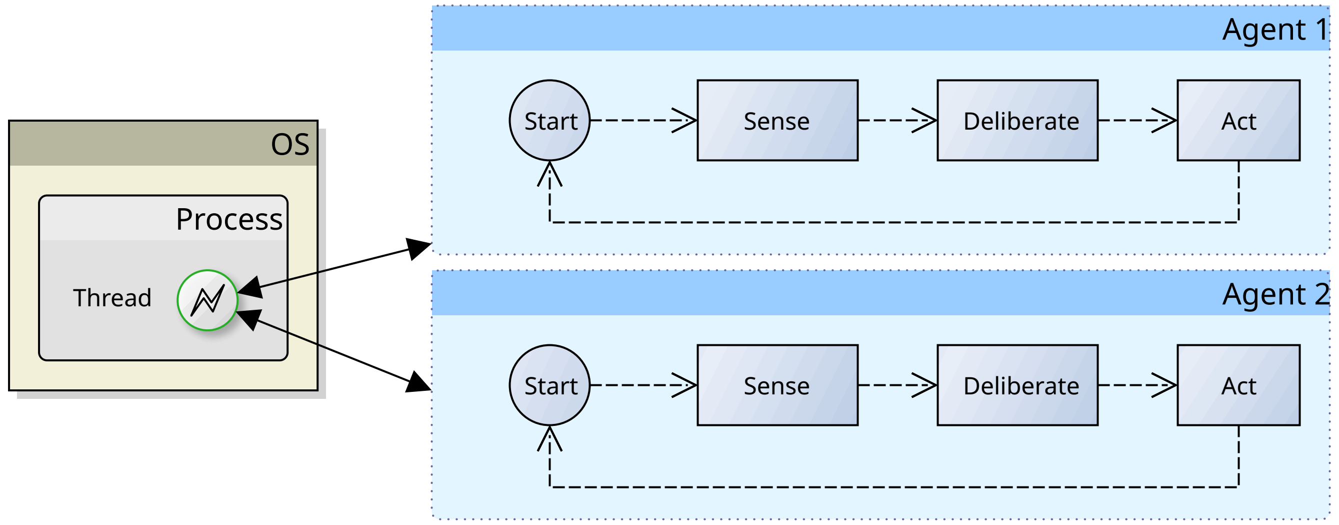
COMMON CONCURRENCY PATTERNS FOR MAS

- One-Agent-One-Thread (`1A1T`)
- All-Agents-One-Thread (`AA1T`)
- All-Agents-One-Event-Loop (`AA1EL`)
- All-Agents-One-Executor (`AA1E`)
    - With a fixed-size thread pool
    - With a variable-size thread pool

ONE-AGENT-ONE-THREAD (1A1T)

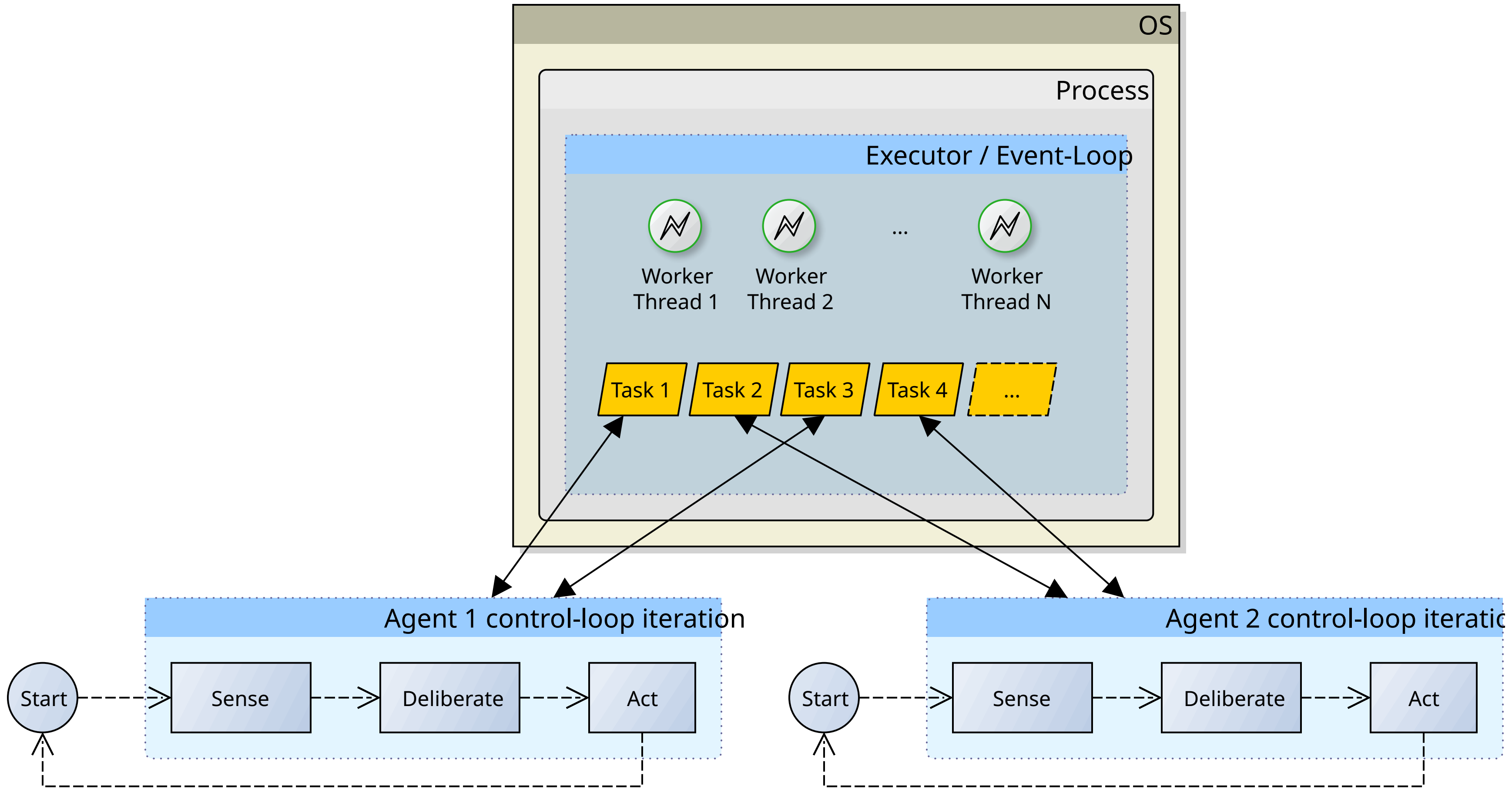


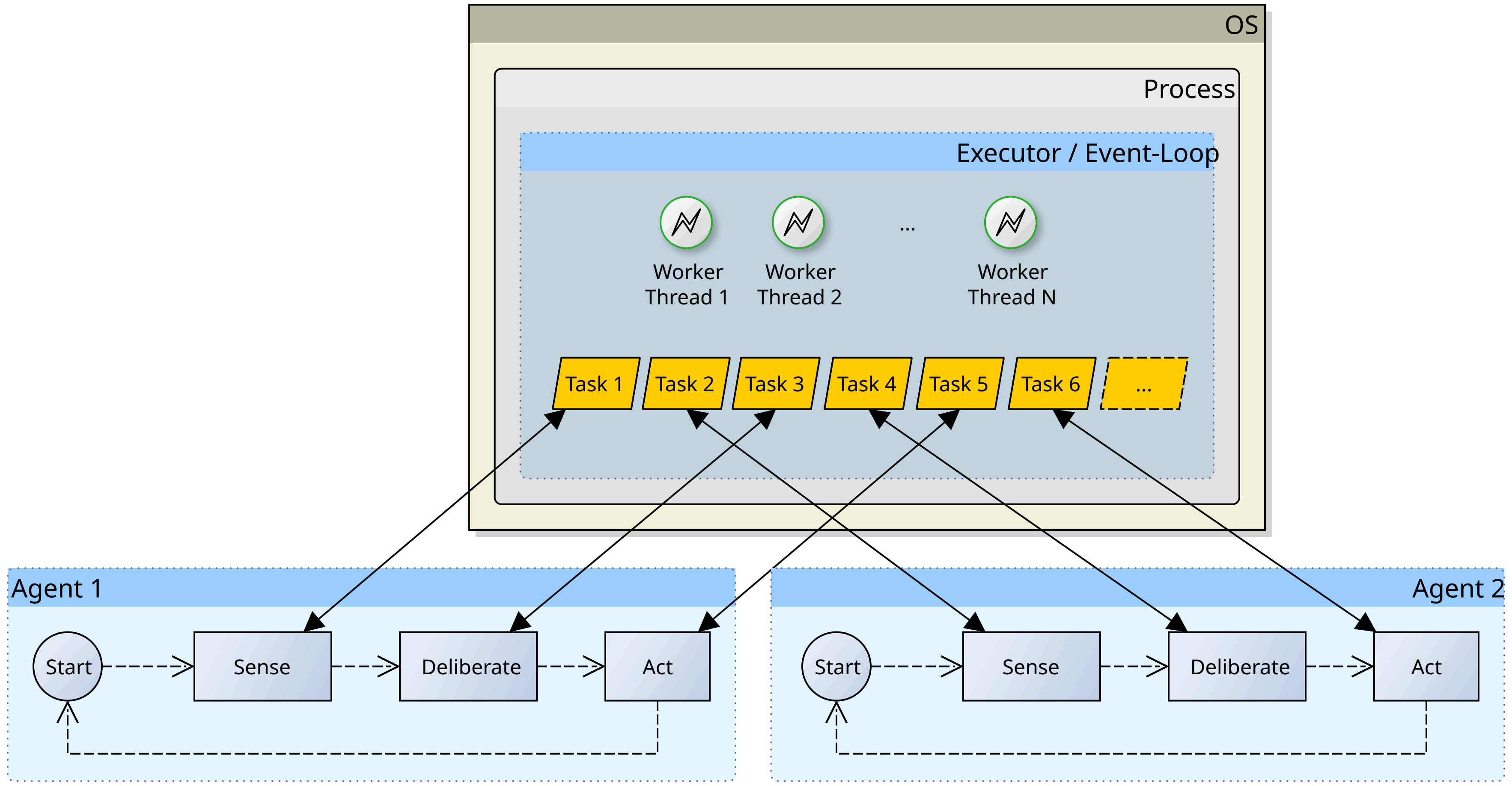
ALL-AGENTS-ONE-THREAD (AA1T)



ALL-AGENTS-ONE-EXECUTOR (AA1E)

Allows for various level of granularity:

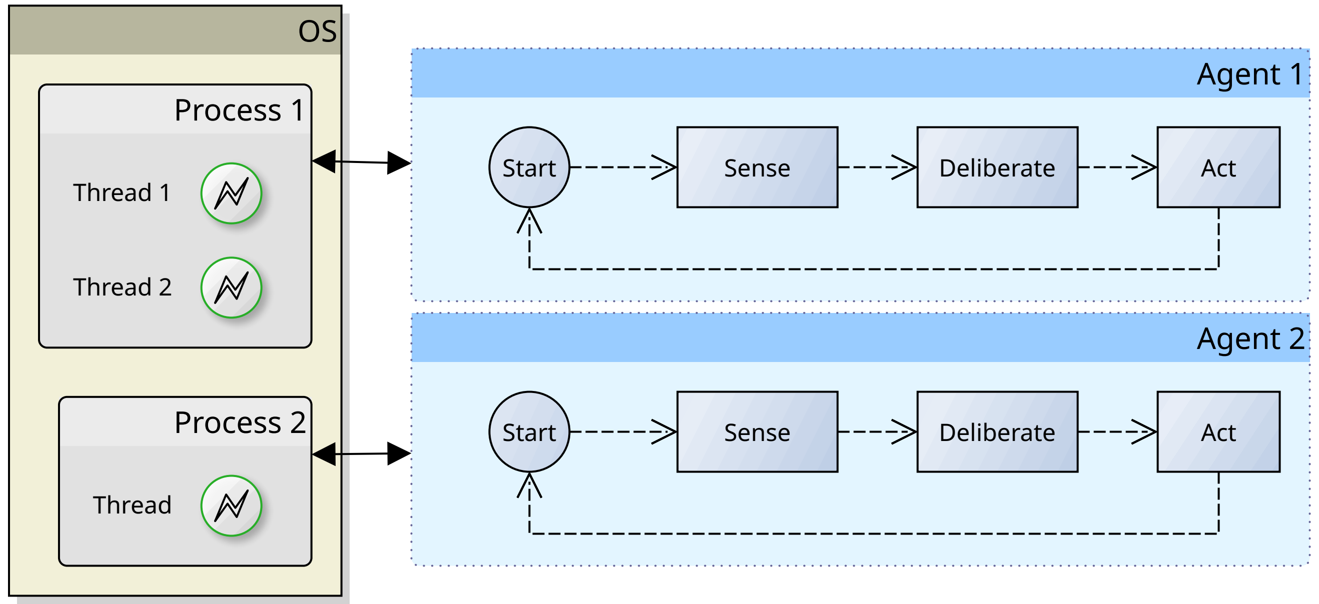




Different properties w.r.t. **fixed** or **variable** amount of worker threads (N)

All-Agents-One-Event-Loop (**AA1EL**) $\equiv$ AA1E with *just one thread*

ONE-AGENT-ONE-PROCESS(1A1P)



WHICH CONCURRENCY ABSTRACTION IS THE MOST APPROPRIATE?

- The selection of an appropriate concurrency model deeply impacts several aspects of the agent programming framework
  - The efficiency of the MAS may improve, but
  - predictability and reproducibility may be affected.
- Capturing and controlling concurrency is crucial,
  and they often are hidden under the framework abstractions

ANALYSIS ON BDI FRAMEWORKS:

METHODOLOGY

We inspected external concurrency in three steps:

1. Empirical Evaluation through a synthetic benchmark
2. Documentation and source code inspection of the selected BDI frameworks
3. Direct contact with maintainers

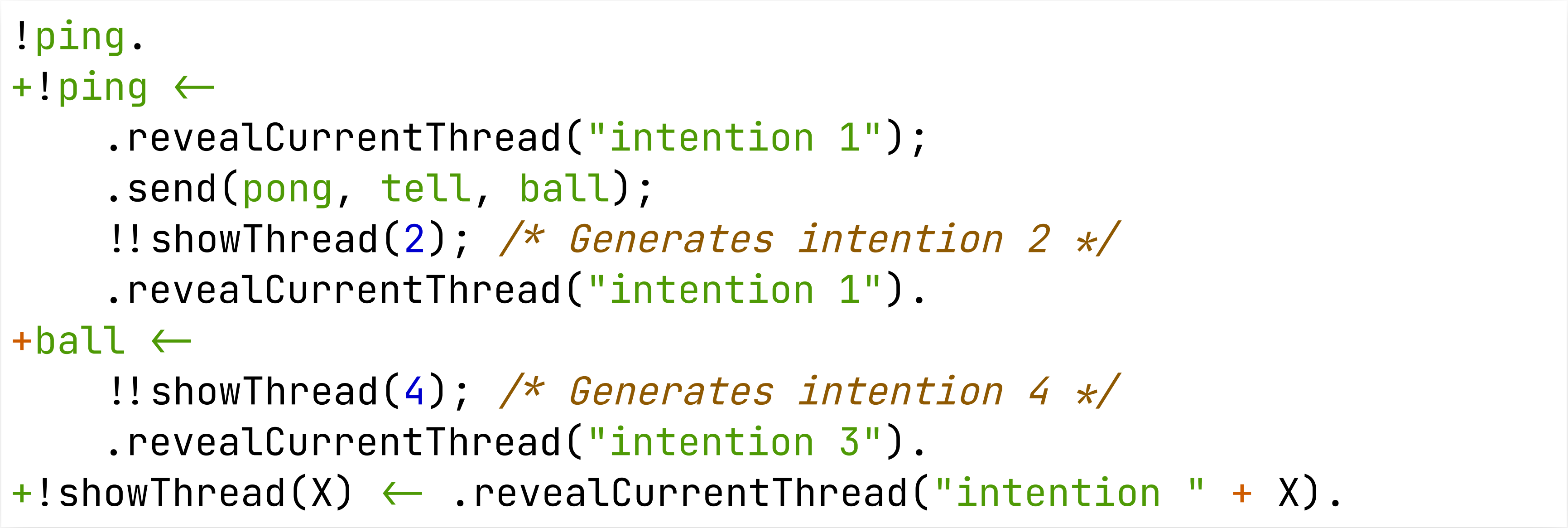
FRAMEWORK SELECTION

We selected actively-maintained and open source BDI programming frameworks:

- Astra
- GOAL
- Jadex
- JaKtA
- Jason
- PHIDIAS
- SPADE-BDI

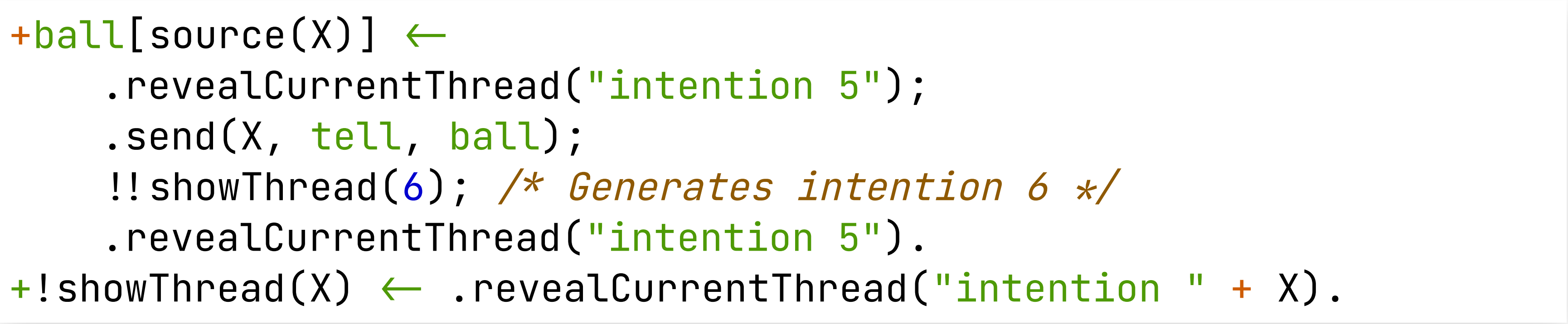
BENCHMARK

AGENT: PINGER



```
!ping.
+!ping ←
    .revealCurrentThread("intention 1");
    .send(pong, tell, ball);
    !!showThread(2); /* Generates intention 2 */
    .revealCurrentThread("intention 1").
+ball ←
    !!showThread(4); /* Generates intention 4 */
    .revealCurrentThread("intention 3").
+!showThread(X) ← .revealCurrentThread("intention " + X).
```

AGENT: PONGER



```
+ball[source(X)] ←
    .revealCurrentThread("intention 5");
    .send(X, tell, ball);
    !!showThread(6); /* Generates intention 6 */
    .revealCurrentThread("intention 5").
+!showThread(X) ← .revealCurrentThread("intention " + X).
```

RESULTS

Model ⇒ Tech. ⇓	1A1T	AA1T	AA1EL	AA1E fixed	AA1E variable	1A1P
Astra	✓	✓	✔	✔	✔	✓
Goal	✔	✘	✘	✘	✘	✘
Jadex	✓	✔	✓	✓	✔	✔
JaKtA	✔	✔	✔	✔	✔	✓
Jason	✔	✓	✔	✔	✓	✔
Phidias	✔	✘	✘	✘	✘	✔
Spade-BDI	✘	✘	✔	✘	✘	✔

LEGEND

- ✔ ≡ supported
- ✘ ≡ not supported
- ✓ ≡ supported in principle, but requires the user to implement it

DISCUSSION

Takeaway 1: better for a BDI framework to support *multiple* concurrency patterns

Takeaway 2: even better for a BDI framework to support concurrency patterns *customisability* on the **user-side**

- supporting e.g. comparing perfomance among different concurrency patterns, for the same MAS
- supporting e.g. *prioritising* determinism over efficiency (AA1T) for testing
- supporting e.g. *prioritising* indipendence of control flows (1A1T) for I/O-bound tasks

CONCLUSIONS

It is necessary to separate BDI architecture from its actual execution

- without impacting the architecture definition
- without necessarily knowing how to program concurrency abstractions
- choosing dynamically which concurrent execution suits the scenario