Towards a Multi-Level Explainability Framework for Engineering and Understanding BDI Agent Systems

Elena Yan* Samuele Burattini* Jomi Fred Hübner** Alessandro Ricci*

* Alma Mater Studiorum – University of Bologna
elena.yan@studio.unibo.it samuele.burattini@unibo.it a.ricci@unibo.it

**Federal University of Santa Catarina, 88040-900 Florianópolis (SC), Brasil
jomi.hubner@ufsc.br

24th Workshop "From Objects to Agents" (WOA23)
6th-8th November, Rome, IT

Context

Explainability is becoming a crucial property of any software system.

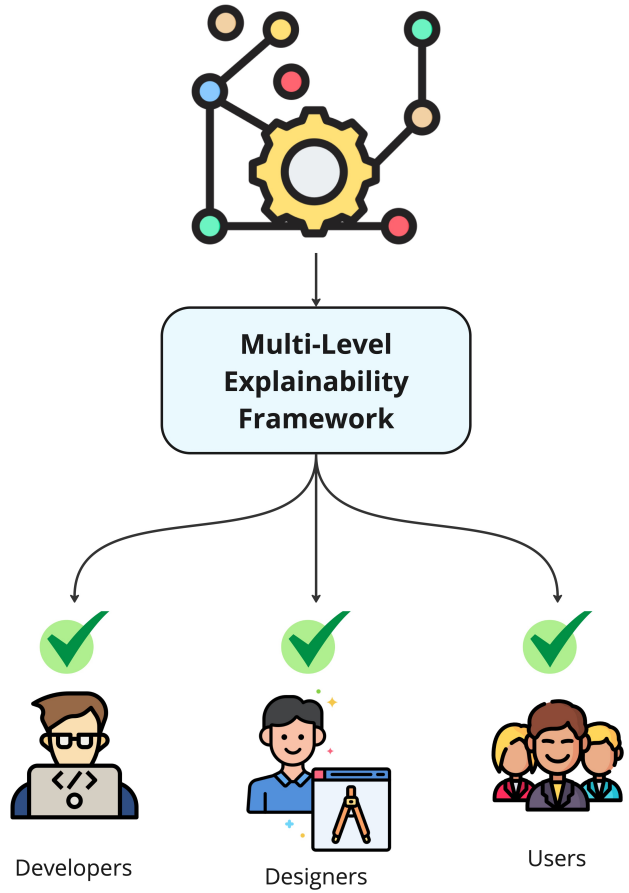
- **User perspective** — to increase the level of understanding [10, 8] and trust [3] in the system,
- **Engineering perspective** — to support activities such as debugging [13, 1, 9, 5], validation [11], and testing [12, 6].

Our focus is on autonomous agents and MAS, particularly for agents based on the **Belief-Desire-Intention (BDI)** abstraction[4].

Objectives

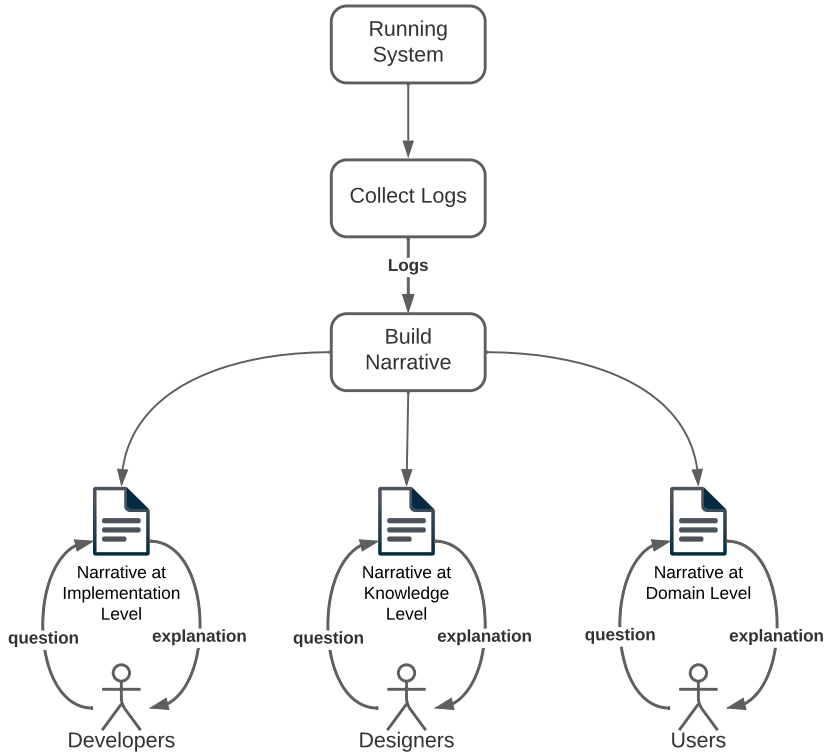
We propose a **multi-level explainability framework**:

- to explain the behaviour of BDI agents at multiple levels of abstraction,
- that can be used by **different classes of users** with different needs:
  - Developers,
  - Designers,
  - Users.



Logs as Narratives to Explain and Understand

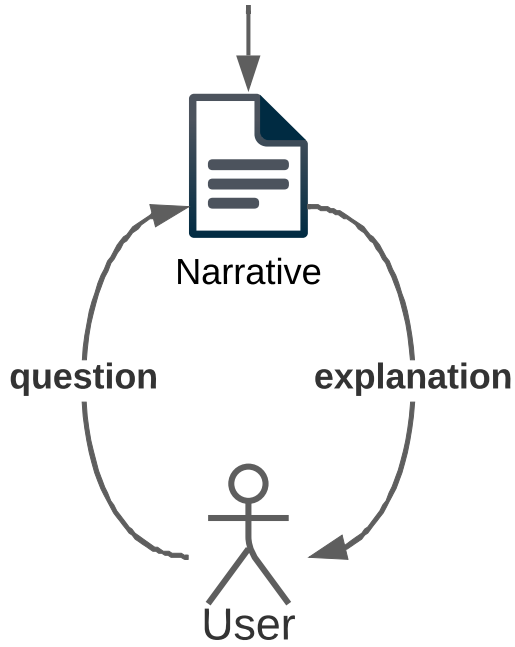
1. **Running the agent system**
2. **Collect Logs:** rely on *implicit logging*
   - the execution trace is recorded without *external* intervention
3. **Build Narratives** at multiple levels:
   - converting logs into a textual series of events that describe the behaviour of the system



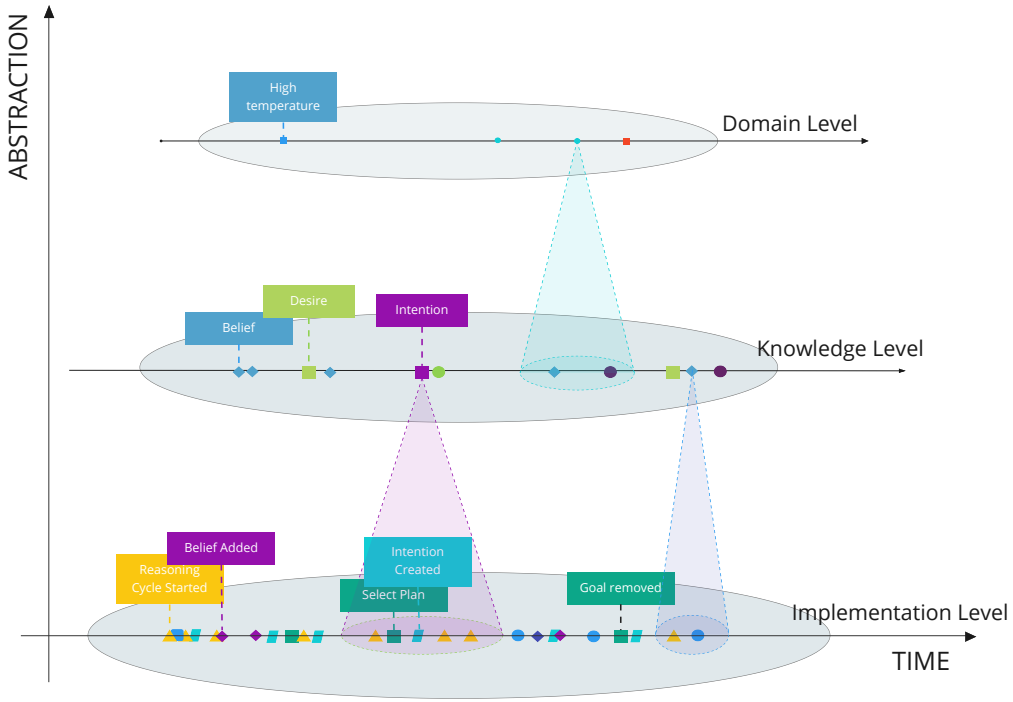
Narrative to Explain User Questions

Users that may have **questions** on the system behaviour:

1. Select the appropriate narrative level,
2. Inspect the narrative to find the main relevant events,
3. Get an **explanation** of the event.

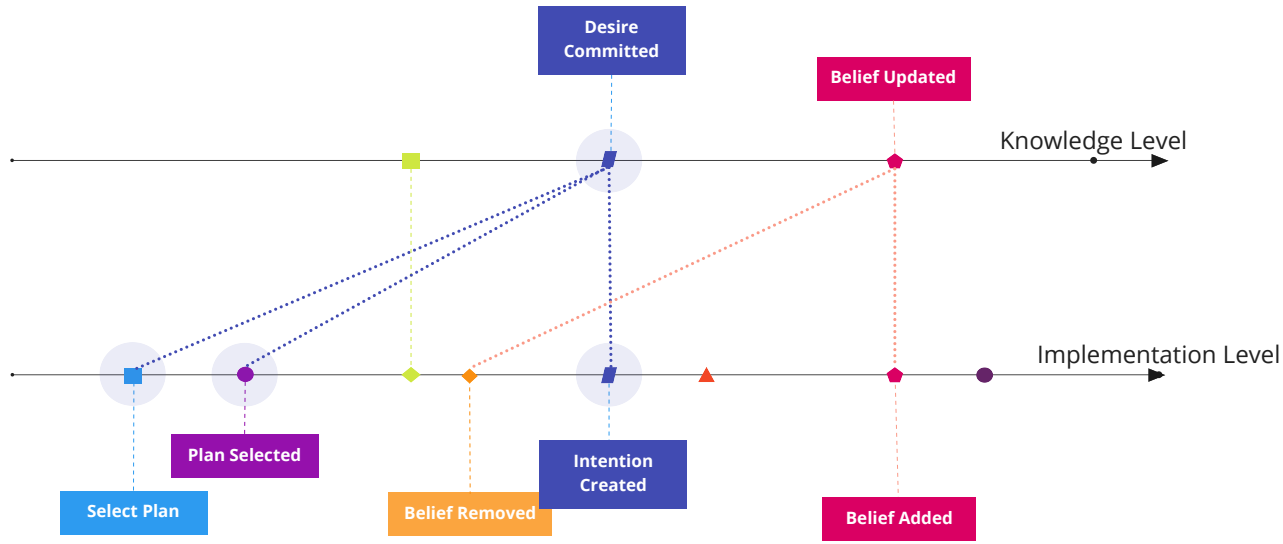


A Multi-Level Perspective



- **Implementation Level** → useful for developers.
- **Knowledge Level** → useful for designers as well as developers.
- **Domain Level** → useful for end-users and software engineers.

Generating narrative process from one level to another



- Identify the events at the new level,
- Identify a mapping from the lower level,
- Generate narrative as text of the new level.

The Implementation Level

- Useful for **developers** in the software engineering phases (e.g. debugging and testing).
- We choose **Jason** as starting BDI technology.
- Detailed and technical level that follows the Jason *operational semantics* and reasoning cycle.

Questions for Implementation Level

(Q1) Why did a plan fail or get an error?

(Q2) Why did the agent execute an action?

(Q1) Why did the robot agent fail the plan `has(owner,beer)`?

1. `Intention 3 has(owner, beer) waiting to execute action get(beer), state: waiting, current step: get(beer); close(fridge); !at(robot, owner); hand in(beer); ?has(owner, beer); .date(YY,MM,DD); .time(HH,NN,SS);`
2. `External action get(beer) triggered`
3. `External action get(beer) failed`
4. `New reasoning cycle started: 14`
5. `Goal has(owner, beer) removed because the action get(beer) failed`

The Knowledge Level

- Useful for **designers** and **developers** who want to focus more on agent behaviour, abstracting from the implementation details.
- We identify the **BDI abstraction** to describe the agent behaviour regardless of its implementation.

Questions for Knowledge Level

(Q3) Why does the agent have this desire?

(Q4) Why did the agent intend to do this?

(Q4) Why did the robot agent not intend to bring me beer?

- `I have a new desire has(owner, beer) created from agent owner by an achieved message`
- `I committed to desire has(owner,beer) because I believe (too much(beer) and limit(beer,L))`, `and it became a new intention 24 has(owner, beer)`
- `I executed internal action .concat(‘‘The Department of Health does not allow me to give you more than 10 beers a day! I am very sorry about that!’’,M) because of intention 24 has(owner, beer)`

The Domain Level

- Useful for **end-users** and **software engineers**.
- The narrative focuses more on the **functionalities** of the system dealing with its requirements and domain-specific insights.

Questions for Domain Level

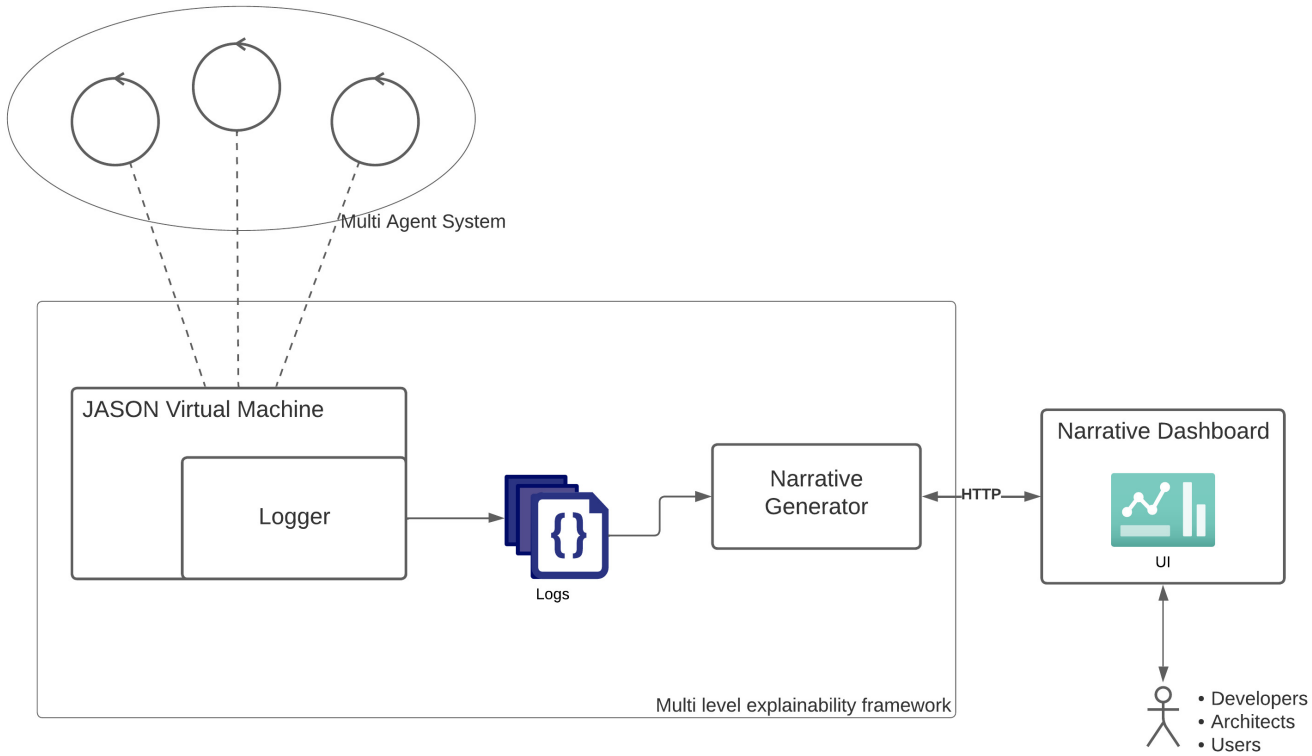
(Q5) Why is this action executed or not executed?

(Q5) Why did the robot inform the user that it could not satisfy her request?

1. `The robot received a request to bring a beer from the owner`
2. `The robot accepted to handle the owner's request`
3. `The robot verified that the owner's request could not be satisfied, due to Health Department laws`
4. `The robot informed the owner that it could not satisfy her request`

→ This level is still under discussion and it is a future direction.

Main Components of the Explainability Framework



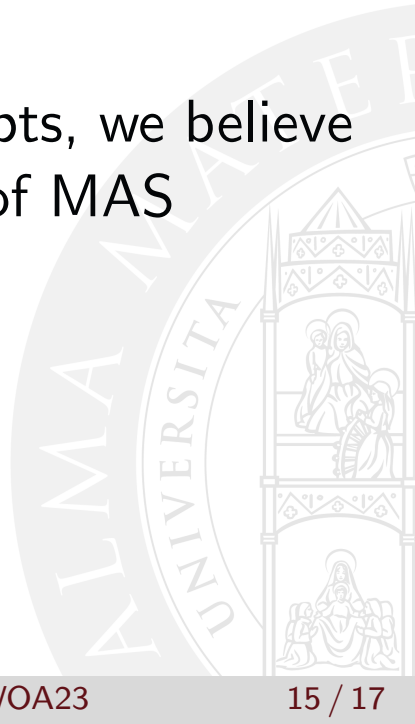
- **Logger**, which generates the log trace for each entity in the system.
- **Narrative Generator**, which processes the logs, builds the narrative at different levels and presents in a Web Dashboard.

Future work

Future directions:

- move towards the **domain level** $\rightarrow$ including formal description of system requirements, use cases and system stories [10],
- integration of **cause-effect** relationships [7].

Although our prototype is based on Jason and its BDI concepts, we believe these levels to be general enough to be applied to any kind of MAS technology.

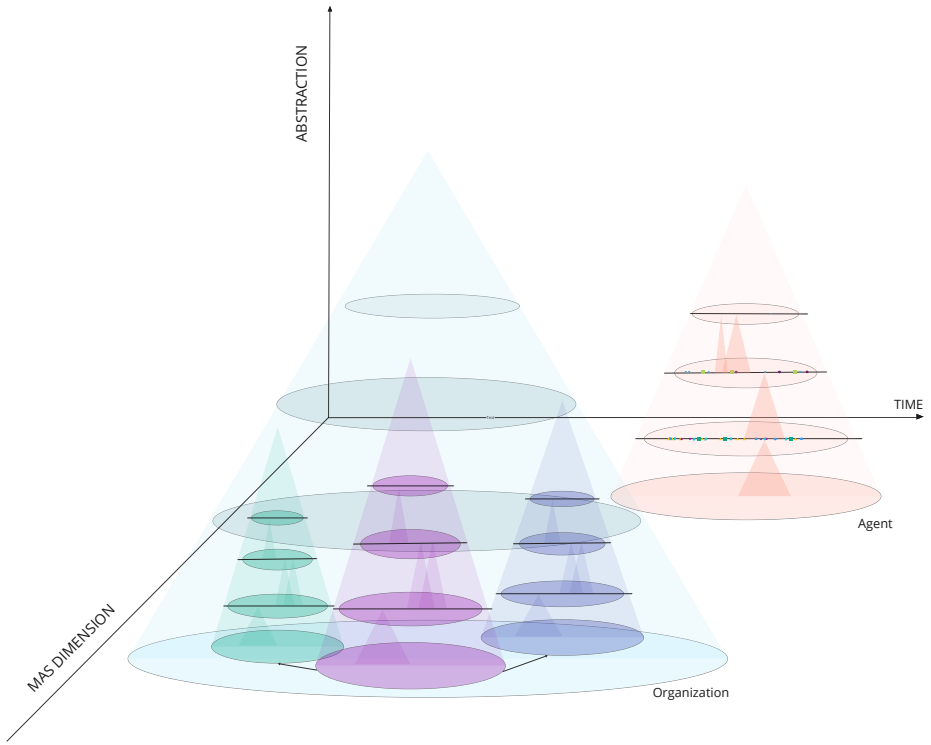


Multiple dimensions and levels of abstraction

In this first exploratory study, we are delimited to the **agent dimension**.

The idea of multi-level explainability could also be extended to **multiple dimensions** of a MAS [2] involving:

- *interaction* dimension,
- *environment* dimension,
- *organisation* dimension.



Towards a Multi-Level Explainability Framework for Engineering and Understanding BDI Agent Systems

Elena Yan* Samuele Burattini* Jomi Fred Hübner** Alessandro Ricci*

* Alma Mater Studiorum – University of Bologna
elena.yan@studio.unibo.it samuele.burattini@unibo.it a.ricci@unibo.it

**Federal University of Santa Catarina, 88040-900 Florianópolis (SC), Brasil
jomi.hubner@ufsc.br

24th Workshop "From Objects to Agents" (WOA23)
6th-8th November, Rome, IT

References I

[1] Tobias Ahlbrecht.
An algorithmic debugging approach for belief-desire-intention agents.
Annals of Mathematics and Artificial Intelligence, pages 1–18, 05 2023.

[2] Olivier Boissier, Rafael H. Bordini, Jomi F. Hübner, and Alessandro Ricci.
Dimensions in programming multi-agent systems.
The Knowledge Engineering Review, 34, 2019.

[3] Dawn Branley-Bell, Rebecca Whitworth, and Lynne Coventry.
User trust and understanding of explainable ai: Exploring algorithm visualisations and user biases.
In Masaaki Kurosu, editor, *Human-Computer Interaction. Human Values and Quality of Life*, pages 382–399, Cham, 2020. Springer International Publishing.

[4] Michael Bratman.
Intention, Plans, and Practical Reason.
Cambridge: Cambridge, MA: Harvard University Press, 1987.

References II

[5] Mehdi Dastani, Jaap Brandsema, Amco Dubel, and John-Jules Ch. Meyer.
Debugging bdi-based multi-agent programs.
In Lars Braubach, Jean-Pierre Briot, and John Thangarajah, editors, *Programming Multi-Agent Systems*, pages 151–169, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[6] Erdem Eser Ekinci, Ali Murat Tiryaki, Övünç Çetin, and Oguz Dikenelli.
Goal-oriented agent testing revisited.
In Michael Luck and Jorge J. Gomez-Sanz, editors, *Agent-Oriented Software Engineering IX*, pages 173–186, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[7] Shakil M Khan and M Rostamigiv.
On explaining agent behaviour via root cause analysis: A formal account grounded in theory of mind.
In *Proceedings of the 26th European Conference on Artificial Intelligence ECAI*, pages 30–09, 2023.

References III

[8] D. N. Lam and K. S. Barber.
Comprehending agent software.
In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '05, page 586–593, New York, NY, USA, 2005. Association for Computing Machinery.

[9] Guillaume Pothier and Éric Tanter.
Back to the future: Omniscient debugging.
IEEE Software, 26:78–85, 11 2009.

[10] Sebastian Rodriguez, John Thangarajah, and Michael Winikoff.
User and system stories: An agile approach for managing requirements in aose.
In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '21, page 1064–1072, Richland, SC, 2021. International Foundation for Autonomous Agents and Multiagent Systems.

References IV

[11] Sebastian Rodriguez, John Thangarajah, and Michael Winikoff.
A behaviour-driven approach for testing requirements via user and system stories in agent systems.
In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pages 1182–1190, 2023.

[12] Sebastian Rodriguez, John Thangarajah, Michael Winikoff, and Dhirendra Singh.
Testing requirements via user and system stories in agent systems.
In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 1119–1127, 2022.

[13] Michael Winikoff.
Debugging agent programs with why? questions.
In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17, page 251–259, Richland, SC, 2017. International Foundation for Autonomous Agents and Multiagent Systems.