Requirements-based Explainability for Multi-Agent Systems

Extended Abstract

Sebastian Rodriguez
RMIT University
Melbourne, Australia
sebastian.rodriguez@rmit.edu.au

John Thangarajah
RMIT University
Melbourne, Australia
john.thangarajah@rmit.edu.au

Michael Winikoff
Victoria University of Wellington
Welington, New Zealand
michael.winikoff@vuw.ac.nz

ABSTRACT

Explainability is essential for building trust in intelligent and autonomous systems. However, existing techniques for explainability focus on the behaviour of the system, but do not go back to the system's requirements. We provide fully traceable explanations that link back to requirements, expressed as User and System stories, by extending previous work on explainable agents (XAg) that uses an agent design pattern. Our implementation leverages industry-grade mainstream monitoring tools.

KEYWORDS

Explainable Agents (XAg); requirements; Verification; Monitoring

ACM Reference Format:
Sebastian Rodriguez, John Thangarajah, and Michael Winikoff. 2025. Requirements-based Explainability for Multi-Agent Systems: Extended Abstract. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 3 pages.

1 INTRODUCTION

Explainability is critical to the development of appropriate levels of trust [2, 5–7, 10, 14], and plays an important role in enabling transparency of intelligent and autonomous systems [3, 4]. However, existing approaches to generating explanations such as "why did you do X?" only provide an explanation in terms of the behaviour of the implemented system. What they do not do is go further back to the system's *requirements* and use those for explanation. This is a significant limitation since clearly a system's behaviour may not match its requirements perfectly.

In this paper we propose a combination of an approach for explanation (using the TriQPAN XAg design pattern [11]) with an approach for specifying requirements in order to give end-to-end explanations, from behaviour to requirements.

To represent requirements we build on recent work [12] which presented an agile approach to capturing requirements in agent systems via *user and system stories* (USS). These present the behavioural requirements from the user's and system's perspective, respectively. A *user story* [1] is an informal, natural language template-based description of software requirements, often used in *Agile* development to represent end-user needs (e.g. Scenario 1, lines 1-3). A *system story* refines user stories by considering the system's perspective [12]. Each story is accompanied by a set of *acceptance criteria* which identify the scenarios under which the system behaviour is acceptable by the users or stakeholders (e.g. Scenario 1, lines 5-16).

Scenario 1: System Story for the top level "GetCoffee" goal

```
1  Feature: GetCoffee
2    As a researcher, I want to GetCoffee
3    so I can finish my paper's experiments
4
5    Scenario: plan-kitchencoffee
6      Given I believe staffCardAvailable is true
7      When I adopt the GetCoffee goal
8      Then plan KitchenCoffee is applicable
9    Scenario: plan-officecoffee
10     Given I believe annInOffice is true
11     When I adopt the GetCoffee goal
12     Then plan OfficeCoffee is applicable
13   Scenario: plan-shopcoffee
14     Given I believe haveMoney is true
15     When I adopt the GetCoffee goal
16     Then plan ShopCoffee is applicable
```

Our approach has been implemented[1]. It uses a range of industry-grade state-of-the-art tools and approaches. Specifically, we used OpenTelemetry[2], also known as OTel, a vendor-neutral open source Observability framework for instrumenting, generating, collecting, and exporting telemetry data ("signals") such as traces, metrics, and logs. It is the *de facto* standard to instrument, generate, collect, and export telemetry data.

This provides a number of benefits. Firstly, we are able to provide a high-quality solution to developers that is robust, efficient, usable, well documented, and scalable to handle (e.g.) many events. The tools also have a rich set of features and a large user community, and hence good support. Secondly, because we use mainstream software engineering approaches and tools, we facilitate the adoption of agent technology by mainstream developers. Finally, using these tools allows for integration with other systems and tools.

We implemented an agent system with SARL [9], a popular open-source agent-oriented programming language designed for building multi-agent systems [8], using the recent extension with goal-oriented behaviours [12, 13] to implement the get-coffee example as used in [11, 15]. We evaluated the system, showing that it is able to detect a range of seeded errors, and that it generates explanations that link back to requirements.

[1] https://github.com/srodriguez-research/xag-monitoring-prototype
[2] https://opentelemetry.io

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

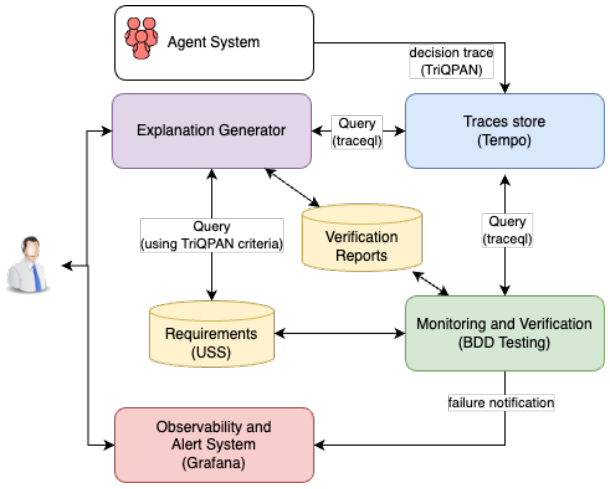

Figure 1: System architecture overview

2 EXPLAINING DECISIONS VIA REQUIREMENTS

In order to explain the agent's decisions based on the original requirements captured, a number of tools and techniques and more importantly, a set of practices and development standards are required. The main steps are as follows:

(1) Requirements are captured using *User and System Stories* by domain experts and developers working together.
(2) At the development stage, a USS will be selected to be implemented by a developer.
(3) The implementation needs to ensure that explainability is built into the system design. We use the TriQPAN XAg design pattern, where an `XAg Process Event` captures all the information required to understand a decision at that moment in time. This will include the *trigger* of the decision; the mental state *queried*; the *process* and *criteria* used to make the decision and finally the *actions* taken.
(4) The agent system is then deployed. During execution, each decision process (e.g a plan selection) generates an `XAg Process Event` that is sent to a *Traces Store*.
(5) Developers and Domain experts will then utilise *Explainability and Monitoring Systems* to understand and verify the correct behaviours of the agent system.

The key modules of the architecture that enables our explanation generation are presented in Figure 1. Once the agent system is executing, it will use OpenTelemetry libraries to send traces with TriQPAN information to the *Traces Store*[3]. This requires a set of semantic conventions, which we have developed based on the TriQPAN design pattern.

The observability module includes, in addition to the TriQPAN information, additional information gathered by OpenTelemetry, including CPU and memory usage, thread executions, and more; giving the user a comprehensive view of the system's performance.

[3]For the Traces Store we chose to use Grafana Tempo (https://grafana.com/oss/tempo/) for their powerful querying language TraceQL, which other modules can use to query for traces that meet specific conditions

Why did you select KitchenCoffee?
In that moment, I believed that *haveMoney was False*; *staffCardAvailable was True*; and *annInOffice was false* And I applied the following criteria *(see Scenario 1)*

Figure 2: Explanations generated from USS specifications

In addition to considering explanations that are generated on demand from the user, we also continuously monitor the system's execution to proactively detect when the system behaves inappropriately. This is done by the *Monitoring and Verification* module.

The *Explanation Generator* module queries the traces store to retrieve information to explain agent decisions to the user. We now briefly describe how this is done.

As discussed, each agent decision process will be recorded as a trace containing TriQPAN information in the traces store. Among the TriQPAN information, agents have to record the criteria used during the process to make the decision. In our approach that criteria is specified using a *System Story*. To link the two elements (requirement and decision), we include in the captured information an attribute *criteria* that points to the *requirement-id*.

To illustrate the process, assume the user asks the *Explanation generator*, *"Why did you select KitchenCoffee?"* The module will then follow the steps below (refer to Figure 1):

(1) Query the *Traces Store* using TraceQL to retrieve the trace of the decision at that point in time.
(2) Parse the trace to find the requirement-id defined in the criteria attribute used to make the decision (i.e., "getcoffee/plan-rating" - stored as a trace attribute)
(3) Query the *Requirements* database to find the *User and System Story* matching this requirement-id.
(4) Query the *Verification Reports* DB to confirm this decision complies with the requirements.
(5) Generate an explanation using all the above information. Figure 2 shows the answer to the question *"Why did you select KitchenCoffee?"*. It gives its reasoning and explicitly refers to Scenario 1.

The explanation generation depends on the question being asked, with a range of question types being possible [11, 15], e.g.: Why did the agent have a certain belief at a certain point in time? Why it chose a particular way to achieve a goal? Why did it update a belief? (e.g. concluding from *battery_charge* = 20% that *batter_level* is LOW) Why did it choose to suspend or abandon a goal?

These processes may be chained together (e.g., a belief update triggers a goal adoption and then a plan selection); or structured as a sub-process of a larger decision (e.g., for the plan selection to complete it must first trigger the applicability decisions for each of the relevant plans.)

ACKNOWLEDGMENTS

This research is partially supported by the C2IMPRESS project funded by the EU.

REFERENCES

[1] Mike Cohn. 2004. *User Stories Applied: For Agile Software Development.* Addison Wesley Longman Publishing Co., Inc., USA.

[2] Virginia Dignum. 2019. *Responsible Artificial Intelligence: How to Develop and Use AI in a Responsible Way.* Springer. https://doi.org/10.1007/978-3-030-30371-6

[3] High-Level Expert Group on AI. 2019. Ethics guidelines for trustworthy AI. https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai

[4] IEEE. 2022. IEEE Standard for Transparency of Autonomous Systems. IEEE Std 7001-2021. , 54 pages. https://doi.org/10.1109/IEEESTD.2022.9726144

[5] Pat Langley, Ben Meadows, Mohan Sridharan, and Dongkyu Choi. 2017. Explainable Agency for Intelligent Autonomous Systems. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, Satinder Singh and Shaul Markovitch (Eds.). AAAI Press, 4762–4764. https://doi.org/10.1609/aaai.v31i2.19108

[6] P Jonathon Phillips, Carina A Hahn, Peter C Fontana, Amy N Yates, Kristen Greene, David A Broniatowski, and Mark A Przybocki. 2021. *Four Principles of Explainable Artificial Intelligence.* Technical Report NIST IR 8312. National Institute of Standards and Technology (U.S.), Gaithersburg, MD. NIST IR 8312 pages. https://doi.org/10.6028/NIST.IR.8312

[7] Paul Robinette, Wenchen Li, Robert Allen, Ayanna M. Howard, and Alan R. Wagner. 2016. Overtrust of Robots in Emergency Evacuation Scenarios. In *The Eleventh ACM/IEEE International Conference on Human Robot Interation, HRI 2016, Christchurch, New Zealand, March 7-10, 2016*, Christoph Bartneck, Yukie Nagai, Ana Paiva, and Selma Sabanovic (Eds.). IEEE/ACM, 101–108. https://doi.org/10.1109/HRI.2016.7451740

[8] Sebastian Rodriguez, Stephane Galland, and Nicolas Gaud. 2024. Ten Years of SARL: What's Next?. In *The 21st European Conference on Multi-Agent Systems*. Dublin, Ireland.

[9] Sebastian Rodriguez, Nicolas Gaud, and Stéphane Galland. 2014. SARL: A General-Purpose Agent-Oriented Programming Language. In *The 2014 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, Vol. 3. IEEE Computer Society Press, Warsaw, Poland, 103–110. https://doi.org/10.1109/WI-IAT.2014.156

[10] Sebastian Rodriguez and John Thangarajah. 2024. Explainable Agents (XAg) by Design. In *Proceedings of the 2024 International Conference on Autonomous Agents and Multiagent Systems (Blue Sky) (AAMAS '24)*. Auckland, New Zeland, 2712–2716. https://dl.acm.org/doi/10.5555/3635637.3663263

[11] Sebastian Rodriguez, John Thangarajah, and Andrew Davey. 2024. Design Patterns for Explainable Agents (XAg). In *Proceedings of the 2024 International Conference on Autonomous Agents and Multiagent Systems (AAMAS '24)*. Auckland, New Zeland, 1621–1629. https://dl.acm.org/doi/10.5555/3635637.3663023

[12] Sebastian Rodriguez, John Thangarajah, and Michael Winikoff. 2021. 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)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1064–1072. https://doi.org/10.5555/3461017.3461136

[13] Sebastian Rodriguez, John Thangarajah, and Michael Winikoff. 2023. 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 (AAMAS '23)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1182–1190. https://doi.org/10.5555/3545946.3598761

[14] Michael Winikoff. 2017. Towards Trusting Autonomous Systems. In *Engineering Multi-Agent Systems - 5th International Workshop, EMAS 2017, Sao Paulo, Brazil, May 8-9, 2017, Revised Selected Papers (LNCS, Vol. 10738)*, Amal El Fallah Seghrouchni, Alessandro Ricci, and Tran Cao Son (Eds.). Springer, 3–20. https://doi.org/10.1007/978-3-319-91899-0_1

[15] Michael Winikoff, Galina Sidorenko, Virginia Dignum, and Frank Dignum. 2021. Why Bad Coffee? Explaining BDI Agent Behaviour with Valuings. *Artificial Intelligence* 300 (Nov. 2021), 103554. https://doi.org/10.1016/j.artint.2021.103554