# Explainable Agents (XAg) by Design

Blue Sky Ideas Track

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

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

## ABSTRACT

The likes of ChatGPT has propelled the use of AI techniques beyond our community's expectations. Along with this, the fear of AI has also risen, in particular around the ability, or lack thereof, of the AI system to explain its behaviours. Explainability is a key element of building trust and an important issue for our community. In this paper we advocate for agents that are explainable-by-design, that is, explainability is built into the development of agents rather than an afterthought. We propose key features of an explainable agent (XAg) system and propose a general framework that enables explainability. We advocate the use of design patterns to develop XAgs and propose a general design pattern that can be used for any agent architecture. We instantiate our framework for goal-based agents and implement the framework for the SARL agent programming language coupled with a state-of-the-art event management system. We make a call to the developers of other agent programming languages (APLs) in our community to follow suit by instantiating the general framework we propose into their APL, perhaps even enhancing the framework we present. We also propose an open repository of design patterns and examples for agent systems. If nothing else, we hope this paper will inspire further work on XAg from the design perspective as it is critical that multi agent systems are explainable by design!

## KEYWORDS

Explainable AI; AOSE; EMAS

**ACM Reference Format:**
Sebastian Rodriguez and John Thangarajah. 2024. Explainable Agents (XAg) by Design: Blue Sky Ideas Track. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 5 pages.

## 1 INTRODUCTION

ChatGPT and other generative AI tools have brought AI into the spotlight beyond expectations. Every company is now frantically seeking to integrate these technologies into their products and industries. This rapid adoption however, has also raised questions on whether these systems can be trusted.

As Prof. Virginia Dignum states in her book "Responsible AI" [9]: *"Responsible AI is more than the ticking of some ethical 'boxes' or the development of some add-on features in AI system"*. Ensuring the system is *designed responsibly* is crucial to trusting its behaviour. Explainability, therefore, should not be considered as an add-on technique of an AI system, rather it must be embedded into the system's architectural foundations. Whilst, embedding explainability into agents will increase trust and adoption [24], it is not the only benefit. Explainable agents (XAg) can also increase productivity as they can more quickly uncover errors and/or areas of improvement, and it can also help mitigate risks as it provides insights on whether the AI system could potentially violate norms, rules or standards.

Traditional software engineering has promoted strong design conventions to improve a large number of quality attributes of software systems. These quality features (sometimes referred as "-ity" attributes) include modularity, usability, and so on. We argue that "explainability" should be included into the list of quality attributes for AI systems, ensuring that it is considered as a central aspect of the system's design. A well established means of ensuring quality attributes is through the use of *Design Patterns* [14]. Tores et al. called for the use of design patterns in MAS over a decade ago [5] to increase the broader acceptance of the technology. There has been some work on the use of design patterns in MAS [7, 22], for example, [7] provided design patterns for agent-oriented programming, providing templates for realising some agent-oriented concepts and abstractions. There has also been recent work by Washizaki et al. [35] exploring design patterns for machine learning systems, also highlighting the importance of this approach in gaining wider acceptance by the broader software engineering community. However, to the best of our knowledge, no patterns have been proposed for explaining agent behaviours. In this paper, *we advocate the use of design patterns for XAg.*

In this work, we make a call to the AAMAS community to: (i) make explainability a central element of any agent architectural design; (ii) contribute design patterns that ensure explainability is embedded into the agent's architecture; and (iii) develop new or enhance existing agent programming frameworks that implement XAg design patterns, so that the agents created are explainable by design. We propose an open repository that hosts a collection of design patterns for agents systems, including for XAg, and example applications that implement those design patterns.

As a first step towards this, we have developed a general architecture for XAg based on the event-driven architecture which is inherent to agent systems. Within this framework, we have developed a design pattern (TriQPAN ) for XAg. We have instantiated our design pattern for goal-based agents, incorporating it in to the SARL agent programming language [27] coupled with a state-of-the-art event management system. We add our design patterns and three application examples into the open repository as initial contributions to kickstart community contributions.

*Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand.* © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

We hope these preliminary steps inspire further work on XAg from the design perspective as it is critical that multi agent systems are explainable by design.

## 2 BACKGROUND: XAG

While there has been a significant focus on the importance of explainable AI, there is a limited amount of research in Multi-Agent Systems (MAS) in the development of explainable agents. Notable exceptions to this trend include recent work by Winikoff et al. [36] and earlier contributions by Habers et al. [17, 18]. These works leverage goal-plan trees (GPT) [33, 34] as a foundation for providing explanations. GPTs are akin to decision trees and facilitates traceability. Hence, essentially, explaining the rationale behind an action involves tracing the GPT. Habers et al. [18] necessitate agents to record any decisions requiring an explanation through an explicit logging mechanism. For instance, when an agent adopts a goal G at time t, it logs this decision. The explanation log can encompass beliefs, goals, actions, etc. They emphasize that the decision of what to log should depend on the information desired for an explanation. Despite providing templates for specific case studies, their approach lacks a general formalism. Winikoff et al.'s [36] approach extends the work of Habers et al., introducing the concept of "valuings". Their work show that employing these valuings leads to more effective explanations. Further, a significant contribution of [36] lies in their mechanisms for generating explanations in a human-friendly natural language, surpassing a simple trace of a GPT. They provide formal definitions and detailed algorithms for their approach. To the best of our knowledge, the work of Winikoff et al. has not been implemented in an agent programming language but rather exists as prototypes in Haskell and Python, serving to evaluate the presented formalisms and algorithms.

## 3 GENERAL APPROACH FOR XAG

In this section we propose a general approach that enables XAg by design. Figure 1, presents an overview of the general architecture and we describe the components below.

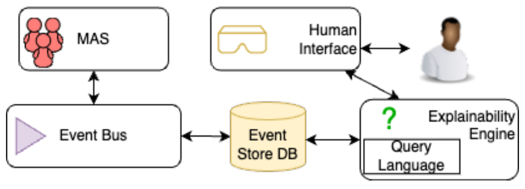

**Figure 1: Architecture Overview**

### Event driven architecture

The *Sense-Deliberate-Act* is the standard reasoning model for any agent architecture. This general agent model is inherently an event-driven architecture [13] which can benefit from decades of software engineering research in this area. An event-driven architecture (EDA) uses events to transmit information that triggers loosely coupled microservices. A wide range of software architecture patterns have emerged to support this new way of building software [19]. In an agent architecture, these microservices translate to modules at different levels of abstraction such as sensing, goal selection, plan selection, intention scheduling, and so on. These events can be stored and queried to generate explanations. For the purpose of storing and querying events, there are numerous well-established mainstream patterns and tools, that can be used in the context of agent systems rather than having to re-invent the wheel, a wheel that has seen decades of best practice in industry.

### Explainable decision-making processes

At the core of our *explainable-by-design* principle is that each decision-making process in the agent's architecture must surface enough information to explain the reasons for the decision. For instance, a goal-oriented agent should be able to explain *why* it is pursuing a particular goal, or a particular plan, or *why not* and so on. The AAMAS community has already started some efforts towards this though they are more "add-on" features than a "built-in" feature of the agent architecture (see Section 2).

We acknowledge that agent technologies are still evolving (and will continue to do so with the rapid adoption of AI), and there is no "one-size-fit-all" solution for agents. So, instead of trying to propose a "silver-bullet" architecture for XAg, we propose to focus on defining a set of design patterns that enable *explainable-by-design* decision-making processes. Similar to the seminal works like the *Gang of Four* [15] for mainstream software engineering, we advocate creating design patterns that can be adapted to different use cases, while ensuring the quality feature of explainability. We have made a contribution towards this and have proposed the TriQPAN pattern which we present in Section 4. We see this as a catalyst to other design patterns for XAgs and indeed other agent reasoning processes.

### Query languages and explanation engines

In the architecture we propose, events generated are stored in a data lake, that is a collection of events from disparate sources. In a large complex and long-running agent system, this will result in potentially very large amounts of events containing information not only of *what* did the agent(s) do, but the information that enables understanding *why* they did it. To understand the why, we need specialized query languages to find these reasons in a potentially large ocean of events. Once again, fortunately, we can borrow and adapt tools from mainstream software engineering for this purpose. Indeed, more often than not, event storage systems provide, well formed sophisticated query languages that can be used to interrogate the event-store.

The explanation engine needs to formulate the appropriate queries, gather and process the results to generate the explanations. Big-data applications such as banking or fintech deal with enormous amounts of streaming data events, and these days utilise machine-learning techniques in managing data. Our community should explore these state of the art approaches for XAgs.

The format the events are stored in, and the query languages are generally not the most intuitive to non-technical human users, who are often the end users of the AI systems. Hence, there is a need for a human interface module to allow queries and present responses in an intuitive human-readable form. Various AI techniques, such as natural language processing, large language models, dialog systems, and many others could be applied for this purpose. We propose this human interface to be decoupled from the other components which

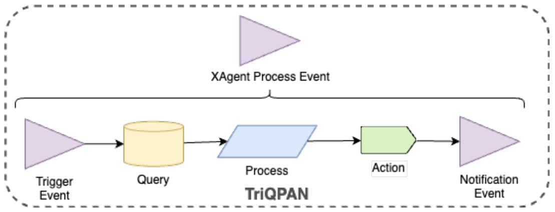

**Figure 2: TriQPAN pattern for agents**

allows the query/explanation process to be more easily tailored to different kinds of human users.

## 4 TRIQPAN : XAG DESIGN PATTERN

Aligned with the approach we advocate for, we propose a design pattern TriQPAN (Trigger, Query, Process, Action and Notify) [28], which is designed to ensure that decision processes are recorded to explain the agent's behaviours. TriQPAN structures the steps of a decision-making process following the *Sense-Deliberate-Act* loop (see Figure 2). Once the decision process has been *triggered* (e.g. by a perception), it will *query* its state or known information (e.g., its belief sets), compute or *process* this information to select the *actions* to perform, and finally, *notify* of its actions to other modules of the agent. All the components of the decision-making process are documented in the XAgentProcess event that is emitted when the process completes.

TriQPAN enables the use of well-known event-driven patterns (i.e., *Event Sourcing* [11] and *CQRS* [12]) in agent architectures for explainability. This allows us to leverage years of industry practice in event-driven architectures and integrate state-of-the-art technologies such as Event Store DB.

In particular, Event Sourcing (ES) is commonly used to track changes in the state of the system for auditing and traceability, although many other benefits can be obtained. When correctly implemented, ES allows to replay events to recreate the state of the system at any given time; create "what if" scenarios by injecting events; and more.

Usually closely associated with ES, Command Query Responsibly Separation (CQRS) pattern is widely used in industry applications to separate the interfaces that modify the state and those that query the state. Furthermore, CQRS enables the creation of multiple query models from the event store to allow more efficient use cases. For XAgs, this allows creating multiple *explainability query models* that can be tailored to different needs (e.g., granularity and depth; temporal traces; specific modules or reasoning aspects; etc.). Coupled with continuous event streams, it opens the possibility of *continuous live explanations* of the agent's decisions.

We instantiated our framework for goal-based agents using the SARL agent programming language [27]. As the engine itself is designed using the TriQPAN pattern, developers always create XAgs that emit XAgentProcess event documenting there goal-based reasoning processes. Detailed discussion can be found in [28]

We hope this is only a first step to a collaborative and interoperable XAg framework. We make a call to the developers of other agent programming languages (APLs) in our community to follow suit by instantiating the general framework we propose into their APL, perhaps even enhancing the framework we present.

## 5 RESEARCH AGENDA

Explainaibility has proven to be a complex topic of AI in general. XAgs are no exception to this. In this section, we highlight some of the most critical areas we need to focus on as a community to enable XAgs that are *explainable-by-design* .

*Standardization.* Most large endeavours require the community to collaborate using a shared understanding. We will follow the natural path of agreeing on the key agent processes and components that will later find its path into *standards*. We have seen efforts towards this goal in proposals like FIPA [20]. Unfortunately, this standardization efforts have been stalled for several years.

To overcome the complexity of a "comprehensive" agent standard, we advocate for a "loosely coupled" approach where standards are focused on events format, syntax and semantics. Focusing on the events (i.e., information exchange units) will allow different decision-making techniques to be interchangeable.

Implicit in this description is the need for standard communication exchange patterns. Messaging architectures can take a wide-range of formats, and each one requires its own standardization [19]. For instance, if we follow a *publisher-subscriber* pattern, the community will have to agree on topics (and subtopics), acceptable events on each topics, and so on. This is similar to the popular robotic architecture *ROS* [37].

We expect that the XAgentProcess event will attract the most attention and debate. As the event exposing the information regarding the decision-making, it will be critical to allow explainability. If we use TriQPAN as a starting point, we need then to provide standards for how to expose each element of the process: (i) trigger will benefit from events standardization mentioned previously; (ii) query will have to summarize *all* information needed to make the decision (see Section 5); and (iii) criteria used in the deliberation step (i.e., process) will have to be accurately document (e.g., *what did you value and considered when deciding your actions?*). A standard (even a de facto one) will serve as an anchor point to open other fields of research, such as query languages (see Section 5).

*Design Patterns and Best Practices.* New problems will emerge as XAgs are developed and whilst a myriad of possible solutions can be advanced, only a few will become accepted *best practices*. Best practices need to define the principles that guide the practice. Initial work on explainability principles are excellent starting points [6, 26]. More comprehensive approaches to responsible AI must be considered as well [8]. Standards on software quality [21] must be extended to include AI specific features (e.g., explainability, accountability, transparency, etc.). The best practices defined by our community will inform these decisions. Finally, *Design patterns* [15] will enable agent system developers to implement these practices. TriQPAN is a first step in this direction for explainability features.

*Exposing information (semantics + data).* Agents' decisions are generated based on its beliefs. These beliefs have associated semantics that in many cases are left implicit for the developer to interpret. Explainable events must surface data with its associated semantics, or make these semantics readily accessible.

Problems related to semantics are already well known in traditional software engineering [32]. Similar problems may arise, such as "misunderstandings" when producing explanations, e.g.,units for numerical values (e.g. Celsius or Fahrenheit for temperature), implicit categories without clear boundaries (e.g. temperatures is HIGH or LOW is subjective), cultural assumptions, etc. This issue becomes more complex when we consider that decisions may be influenced by experience, preferences or learned values. Work on knowledge representation [23] and ontology engineering [16] should be considered in addressing this challenge.

*Query languages.* XAgs will generate large amounts of events. While storing and scaling event-processing can be done efficiently with current technologies, the community will need to work on novel languages to query these *explainability event data lakes* to extract meaningful information. These languages need to inspect streams of events to find the information required to answer the user's questions. Using these query languages, dedicated explanation engines will need to be created to produce human-friendly explanations. Work on event query languages [10] would be useful here.

*Explanation generation and human interfaces.* As hinted above, an essential part will be the engines that interpret the questions and generate the *human-friendly* explanations. In this context we will have the opportunity to extend existing approaches to be able to ask not only *What happened*, but *why it happened*. These engines will have to be able to keep track of the context and dig into the events to produce contextually accurate, complete (yet succinct) answers. In this context, knowledge-graphs, UI patterns for XAI [30], and large language models [25] are natural candidates, but other innovative interfaces (e.g., augmented reality) should also be explored.

*Integrating explainaibility of other AI components.* Agent systems will rely on other AI fields for different components, e.g., deep learning for computer vision during sensing. To have truly explainable agents, our techniques will have to capture the explainability information of these other modules. This will prove to be a challenging and interdisciplinary exercise. Other AI architectures (e.g., deep learning) are developing their own explainability standards and techniques [26, 29]. Our approach to XAgs should be compatible and complementary to them, so we can use these techniques in different agent modules, ensuring explainability is not compromised.

*Complex process interactions.* Explaining the interaction between decision-making (sub-)processes (e.g., how a plan being selected impacted future goal selections) will require us to understand and explain the interactions between these sub-processes. Reasons could be hidden in long sequences of disconnected decisions. With the stream of all-inclusive events stored in a database, we can explore how to understand and explain these decisions (for instance, using causal inference). Creating information protocols for decision-making process interactions may be a good starting point [31].

*Introspection processes.* "Programs that reason about themselves" is not a new quest [2]. Can the same events used to generate explanations for humans be used by the agent to understand the limitations of its own processes and self-improve by modifying the decision processes? This should be possible for plan selection by learning better context conditions, for example.

*Explainable Teams.* Teams of agents are common-place and we are not short of *agent team* models, algorithms, and applications. As we progress towards explainable single-agents, we will need to work on teams capable of explaining their cooperative behaviours and strategies. Work in this field has already started [3, 4] but there is much to be addressed.

## 6 CONCLUSION AND CALL TO ACTION

Autonomous agent systems will play a key role in this new era of responsible AI. It is important to ensure that key features for accountability and transparency are embedded into the design of agent systems. We argue that to achieve *explainability-by-design* we need to focus our attention on three enabling features: (i) event-driven agent architectures that allow loosely couple modules to interact; (ii) design patterns and best practices to ensure explainability is embedded into the agent's design and not an "add-on" feature; and (iii) query languages and explainability engines capable of generating and presenting explanations. As a first step towards this goal, we proposed the TriQPAN pattern to expose relevant information of the decision-making process and implemented it in the SARL APL, which will be publicly released.

We call out to the AAMAS community to help us build an open repository with collections of design patterns and corresponding case studies, to allow us to further our research on explainable agents, converge on standards for key components such as event structures, and reach a maturity that main-stream software engineering has through this approach of design patterns. To kick-start this, we have a created an open repository in GitHub[1], which hosts: a design patterns catalogue (DPC) for agent systems; and a shared and open explainability events database (EEDB).

Each pattern in the DPC must follow a unified documentation structure including purpose, quality feature promoted, problem and solution, etc. Explainability patterns will have to make a contribution to the EEDB to showcase the outputs produced and the XAgentProcess event format promoted. In turn the EEDB will contain streams of explainability events generated for different application domains. Each explainability event will have to document its syntax and semantics. The goal is twofold: (i) serve as a shared knowledge base of explainability event formats; and (ii) serve as datasets and benchmarks for query languages and explanation engines.

We provide the initial contributions to this repository by documenting the TriQPAN XAg design pattern and provide our implementation examples in SARL. We also provide XAgentProcess event stream generated by our goal-oriented engine for three different domains: (i) the "Why bad coffee?" example by Winikoff et al. [36]; (ii) the Search and Rescue example for ad-hoc behaviours [28]; and (iii) the multiagent programming contest "Agents in the City" scenario [1].

All of the contributions to this repository will be community driven supported by public repositories hosted in GitHub to promoted transparency, collaboration and easy access. We hope the community, especially the EMAS sub-community will join us in our efforts in creating explainable agents by design.

## ACKNOWLEDGMENTS

This research is supported by the Commonwealth of Australia as represented by the Defence Science and Technology Group and the C2IMPRESS project funded by the EU.

[1]https://hmteams.github.io/xag/

## REFERENCES

[1] Tobias Ahlbrecht, Jürgen Dix, and Niklas Fiekas (Eds.). 2019. *The Multi-Agent Programming Contest 2018: Agents Teaming Up in an Urban Environment.* Lecture Notes in Computer Science, Vol. 11957. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-030-37959-9

[2] John Batali. 1983. Computational Introspection. (Feb. 1983).

[3] Kayla Boggess, Sarit Kraus, and Lu Feng. 2022. Toward Policy Explanations for Multi-Agent Reinforcement Learning. In *Thirty-First International Joint Conference on Artificial Intelligence*, Vol. 1. 109–115. https://doi.org/10.24963/ijcai.2022/16

[4] Kayla Boggess, Sarit Kraus, and Lu Feng. 2023. Explainable Multi-Agent Reinforcement Learning for Temporal Queries. In *Thirty-Second International Joint Conference on Artificial Intelligence*, Vol. 1. 55–63. https://doi.org/10.24963/ijcai.2023/7

[5] Mario Henrique Cruz Torres, Tony Van Beers, and Tom Holvoet. 2011. (No) More Design Patterns for Multi-Agent Systems. In *Proceedings of the Compilation of the Co-Located Workshops on DSM'11, TMC'11, AGERE! 2011, AOOPES'11, NEAT'11, & VMIL'11* (Portland, Oregon, USA) *(SPLASH '11 Workshops)*. Association for Computing Machinery, New York, NY, USA, 213–220. https://doi.org/10.1145/2095050.2095083

[6] DARPA. 2016. *Explainable Artificial Intelligence (XAI).* Technical Report DARPA-BAA-16-53. Defense Advanced Research Projects Agency, Arlington, VA.

[7] Mehdi Dastani and Bas Testerink. 2016. Design Patterns for Multi-Agent Programming. *Int. J. Agent-Oriented Softw. Eng.* 5, 2/3 (jan 2016), 167–202. https://doi.org/10.1504/IJAOSE.2016.080896

[8] Virginia Dignum. 2019. *Responsible Artificial Intelligence: How to Develop and Use AI in a Responsible Way.*

[9] Virginia Dignum. 2019. *Responsible Artificial Intelligence: How to Develop and Use AI in a Responsible Way* (1st ed.). Springer Publishing Company, Incorporated.

[10] Michael Eckert, François Bry, Simon Brodt, Olga Poppe, and Steffen Hausmann. 2011. A CEP Babelfish: Languages for Complex Event Processing and Querying Surveyed. In *Reasoning in Event-Based Distributed Systems*, Sven Helmer, Alexandra Poulovassilis, and Fatos Xhafa (Eds.). Springer, Berlin, Heidelberg, 47–70. https://doi.org/10.1007/978-3-642-19724-6_3

[11] Martin Fowler. 2005. Event Sourcing. https://martinfowler.com/eaaDev/EventSourcing.html

[12] Martin Fowler. 2011. CQRS. https://martinfowler.com/bliki/CQRS.html.

[13] Martin Fowler. 2017. What Do You Mean by "Event-Driven"? https://martinfowler.com/articles/201701-event-driven.html.

[14] Erich Gamma. 1995. *Design patterns: elements of reusable object-oriented software.* Pearson Education India.

[15] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1994. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley Professional.

[16] F. Gandon. 2002. *Ontology Engineering a Survey and a Return of Experience.* Technical Report RR-4396. INRIA, France.

[17] Maaike Harbers, Karel Bosch, and John-Jules Ch. Meyer. 2009. A Study into Preferred Explanations of Virtual Agent Behavior. In *Proceedings of the 9th International Conference on Intelligent Virtual Agents* (Amsterdam, The Netherlands) *(IVA '09)*. Springer-Verlag, Berlin, Heidelberg, 132–145. https://doi.org/10.1007/978-3-642-04380-2_17

[18] Maaike Harbers, Karel van den Bosch, and John-Jules Meyer. 2009. A Methodology for Developing Self-Explaining Agents for Virtual Training. In *Proceedings of the Second International Conference on Languages, Methodologies, and Development Tools for Multi-Agent Systems* (Torino, Italy) *(LADS'09)*. Springer-Verlag, Berlin, Heidelberg, 168–182. https://doi.org/10.1007/978-3-642-13338-1_10

[19] Gregor Hohpe and Bobby Woolf. 2003. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions* (1 edition ed.). Addison-Wesley Professional, Boston.

[20] IEEE. 2022. Foundation for Intelligent Physicall Agents (FIPA). http://fipa.org/.

[21] ISO. 2011. ISO/IEC 25010:2011 - Systems and Software Engineering — Systems and Software Quality Requirements and Evaluation (SQuaRE) — System and Software Quality Models.

[22] Joanna Juziuk, Danny Weyns, and Tom Holvoet. 2014. Design Patterns for Multi-agent Systems: A Systematic Literature Review. In *Agent-Oriented Software Engineering: Reflections on Architectures, Methodologies, Languages, and Frameworks*, Onn Shehory and Arnon Sturm (Eds.). Springer, Berlin, Heidelberg, 79–99. https://doi.org/10.1007/978-3-642-54432-3_5

[23] H J Levesque. 1986. Knowledge Representation and Reasoning. *Annual Review of Computer Science* 1, 1 (June 1986), 255–287. https://doi.org/10.1146/annurev.cs.01.060186.001351

[24] Aniek F. Markus, Jan A. Kors, and Peter R. Rijnbeek. 2021. The role of explainability in creating trustworthy artificial intelligence for health care: A comprehensive survey of the terminology, design choices, and evaluation strategies. *Journal of Biomedical Informatics* 113 (2021), 103655. https://doi.org/10.1016/j.jbi.2020.103655

[25] OpenAI. 2019. Better Language Models and Their Implications. https://openai.com/research/better-language-models.

[26] 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

[27] 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

[28] 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, [to appear].

[29] Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, and Klaus-Robert Müller. 2019. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning.* Springer Nature.

[30] Tjeerd A. J. Schoonderwoerd, Wiard Jorritsma, Mark A. Neerincx, and Karel van den Bosch. 2021. Human-Centered XAI: Developing Design Patterns for Explanations of Clinical Decision Support Systems. *International Journal of Human-Computer Studies* 154 (Oct. 2021), 102684. https://doi.org/10.1016/j.ijhcs.2021.102684

[31] Munindar P. Singh. 2011. Information-Driven Interaction-Oriented Programming: BSPL, the Blindingly Simple Protocol Language. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2 (AAMAS '11).* International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 491–498.

[32] Lin Tan, Chen Liu, Zhenmin Li, Xuanhui Wang, Yuanyuan Zhou, and Chengxiang Zhai. 2014. Bug Characteristics in Open Source Software. *Empirical Software Engineering* 19, 6 (Dec. 2014), 1665–1705. https://doi.org/10.1007/s10664-013-9258-8

[33] John Thangarajah, Lin Padgham, and Michael Winikoff. 2003. Detecting & Avoiding Interference Between Goals in Intelligent Agents. *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence* (11 2003), 721–726.

[34] John Thangarajah, Lin Padgham, and Michael Winikoff. 2003. Detecting & Exploiting Positive Goal Interaction in Intelligent Agents. *Proceedings of the Interantional Conference on Autonomous Agents* 2, 401–408. https://doi.org/10.1145/860575.860640

[35] Hironori Washizaki, Hiromu Uchida, Foutse Khomh, and Yann-Gaël Guéhéneuc. 2019. Studying Software Engineering Patterns for Designing Machine Learning Systems. *CoRR* abs/1910.04736 (2019). arXiv:1910.04736 http://arxiv.org/abs/1910.04736

[36] Michael Winikoff, Virginia Dignum, and Frank Dignum. 2018. Why Bad Coffee? Explaining Agent Plans with Valuings. In *Computer Safety, Reliability, and Security*, Barbara Gallina, Amund Skavhaug, Erwin Schoitsch, and Friedemann Bitsch (Eds.). Springer International Publishing, Cham, 521–534.

[37] YoonSeok Pyo, HanCheol Cho, RyuWoon Jung, and TaeHoon Lim. 2017. *ROS Robot Programming.* ROBOTIS Co.,Ltd.