



An Abstract Architecture for Explainable Autonomy in Hazardous Environments

DOI:

[10.1109/REW56159.2022.00027](https://doi.org/10.1109/REW56159.2022.00027)

Document Version

Accepted author manuscript

[Link to publication record in Manchester Research Explorer](#)

Citation for published version (APA):

Luckuck, M., Taylor, H., & Farrell, M. (2022). An Abstract Architecture for Explainable Autonomy in Hazardous Environments. In *IEEE 30th International Requirements Engineering Conference Workshops (REW): 2nd International Workshop on Requirements Engineering for Explainable Systems* (pp. 108-113). IEEE 30th International Requirements Engineering Conference Workshops. <https://doi.org/10.1109/REW56159.2022.00027>

Published in:

IEEE 30th International Requirements Engineering Conference Workshops (REW)

Citing this paper

Please note that where the full-text provided on Manchester Research Explorer is the Author Accepted Manuscript or Proof version this may differ from the final Published version. If citing, it is advised that you check and use the publisher's definitive version.

General rights

Copyright and moral rights for the publications made accessible in the Research Explorer are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Takedown policy

If you believe that this document breaches copyright please refer to the University of Manchester's Takedown Procedures [<http://man.ac.uk/04Y6Bo>] or contact uml.scholarlycommunications@manchester.ac.uk providing relevant details, so we can investigate your claim.



An Abstract Architecture for Explainable Autonomy in Hazardous Environments

Matt Luckcuck
Department of Computer Science
& Hamilton Institute,
Maynooth University,
Ireland

Hazel M Taylor
Department of Computer Science,
University of Manchester,
United Kingdom

Marie Farrell
Department of Computer Science
& Hamilton Institute,
Maynooth University
Ireland

Abstract—Autonomous robotic systems are being proposed for use in hazardous environments, often to reduce the risks to human workers. In the immediate future, it is likely that human workers will continue to use and direct these autonomous robots, much like other computerised tools but with more sophisticated decision-making. Therefore, one important area on which to focus engineering effort is ensuring that these users *trust* the system. Recent literature suggests that *explainability* is closely related to how trustworthy a system is. Like safety and security properties, explainability should be designed into a system, instead of being added afterwards. This paper presents an abstract architecture that supports an autonomous system explaining its behaviour (explainable autonomy), providing a design template for implementing explainable autonomous systems. We present a worked example of how our architecture could be applied in the civil nuclear industry, where both workers and regulators need to trust the system’s decision-making capabilities.

I. INTRODUCTION

Autonomous robotic systems (robotic systems controlled by software capable of making decisions without human intervention) are being used, or proposed for use, in a variety of environments that are hazardous to humans. For the nuclear industry, these hazards are often chemical or radiation; for other hazardous environments, they could include: extreme weather, high pressure, or lack of breathable atmosphere [1]. Using autonomous robots in these environments reduces the risk of harm to humans, or enables tasks that humans are not capable of. These tasks could include: remote inspections, e.g. [2], [3]; remote handling, or laser cutting [4]. However, introducing autonomy to sectors like the nuclear industry (which are often, needfully, cautious about new technology) requires that the systems are trusted; trusted by the workers using them and by the sector’s regulator.

Our previous work presents high-level principles for autonomous robotics systems that will be used in hazardous environments [5], which was co-authored by representatives from the Office for Nuclear Regulation, the UK’s civilian nuclear regulator. In that work, we argue that a system’s autonomous components “should be as transparent and verifiable as possible”, and “demonstrably trustworthy” to operators.

Recent work has shown that the concept of *explainability* supports a system being trustworthy. Balasubramaniam et al. [6] conducted a study showing that “explainability is tightly coupled to transparency and trustworthiness of AI

systems.” Loss of human trust in a system is one of the *ethical hazards* identified in the ‘Ethical Design and Application of Robots and Robotic Systems’ standard [7], which suggests using explanations to help restore user trust. Explanations can also compensate for mildly undesirable agent behaviour [8]. Similarly to safety, security, or other ethical hazards identified in [7], explainability should be designed into the system rather than hoping that it can be added during development [9].

This vision paper describes an abstract architecture for building explainable autonomous systems: autonomous systems that can explain and justify their decisions to humans. We refer to this as *explainable autonomy*, to indicate that both symbolic and sub-symbolic AI techniques can be included. Our architecture separates high-level and low-level decision-making. We propose that a Belief-Desire-Intention (BDI) [10], [11] agent is used to make the high-level decisions, which is a style of symbolic AI, where an agent interprets information about its environment into logical statements, which it then uses to rationally choose which of a set of plans to perform. An agent’s *beliefs* are its knowledge about itself and its environment, its *desires* are the agent’s long-term goals, and its *intentions* are the goals it is currently pursuing. Using BDI enables executive decisions to be verified [12, Recipe 1] [5].

Our architecture supports explanations in two ways. 1) the *Central Executive* (the BDI Agent and Explainer) receives all of the system’s information *and* makes its executive decisions, making it easier to build explanations. This is similar to the proposal of an Ethical Black Box [13]. 2) the BDI Agent reasons logically, using knowledge and beliefs, which supports building an explanation. This enables two modes of explanation. 1) explanations about previous actions, e.g. “Why did you take this route instead of that route?”, which justify the system’s behaviour. 2) explanations about future action/inaction, e.g. “Why *can’t* you move over there?”, which check that the system will obey a safety rule and give a deeper insight into how the system works, thereby enhancing safety.

§ II overviews eliciting a system’s explainability requirements, though this is not the paper’s main focus. § III describes our architecture, its Central Executive, and the verification benefits of our approach. § IV presents a worked example of how our architecture supports explanations. Finally, § V gives our concluding remarks.

II. DESIGNING EXPLAINABILITY

A system’s *explainability requirements* – what and how it needs to provide explanations – need to be carefully elicited, because there is no easily reusable detailed definition of what it means for an autonomous system to be explainable; or how this concept overlaps with transparency, explicability, or traceability. A useful explanation should be easily understood by humans, but (in contrast to other software) autonomous systems may make surprising decisions, which means that useful explanations are not always obvious. To design effective explainability, one must know the system’s stakeholders and context: to whom are you explaining, and what the system will be doing [14].

Our work focusses on explaining past action and future inaction; both of which will be requested by a human asking “Why” questions [15]. Explaining past action is useful to check the decision-making behind (either correct or incorrect) behaviour. This is likely, especially in social settings, to be requested by contrastive questions, e.g. “Why did you do *that* instead of *this*?” [16]. Explanations of future inaction answer queries about barriers to performing behaviour. These are requested by questions of the form: “Why can’t you do *this*?”, similar to the system in [17]. Our position is that supporting two types of explanation can help a wide range of stakeholders – including users and regulators – to trust the decisions that the system has made, and that it might make in the future.

The system’s explainability requirements should be carefully elicited, using representative samples of **all** of the system’s stakeholders, and considering the system’s context. The explanations should suit all stakeholders who will depend on explanations for their trust in the system, which means that the types of stakeholders should be carefully identified, e.g. using Tomsett et al.’s model of stakeholder roles [18]. Considering the context aims to produce explanations that are appropriate for the system’s operating environment.

Safety-critical systems are likely to have additional explanation requirements [19]. For example, in an emergency situation where the system is being asked to explain its behaviour, it may be appropriate to provide an explanation with less detail, to avoid wasting time that could be used to avert a dangerous situation [14]. Even in nominal situations, the level of detail in an explanation should not overload operators with information, which could delay their own decisions at a critical moment.

There is emerging research on how best to provide explanations. For example, causal explanations that refer to a behaviour’s intended goal are often preferred [20], [8], but the explanation of some behaviours benefit from referring to its enabling conditions (guards) [20]. However, these were both small user studies (n=30 [20] and n=38 [8]), and the authors of [8] suggest their result could be due to the causal explanations being longer and more detailed than the others that they tested. Also, both of these studies build the system and then test it with users. However, stakeholders (especially end-users or operators) should be involved in the design and evaluation of the explainability; this is important in general,

as well as in the nuclear domain.

Approaches for including stakeholders in requirements elicitation are being developed. In previous work, we begin to interpret the factors of an explanation that stakeholders need in safety-critical nuclear environments by presenting stakeholders (n=16) with scenarios, asking what information they want to know, and comparing the findings to how people explain [21]. A similar approach prompts users with vignettes involving ethical requirements, rather than asking about them directly, as a way of side-stepping social desirability bias [22]. Another approach combines user-centred and participatory design for robotic systems [23], yielding higher participant system acceptance (than before their study). The case study was a socially assistive robot, but given that explanations are a social interaction [16], their approach is likely to be applicable in eliciting explainability requirements.

Requirements elicitation approaches from the literature should be used to ensure that the system’s explainability requirements are valid, unbiased, and unambiguous. Our focus on past and future (in)actions is supported by the literature, but may need extending to meet stakeholder requirements for a specific system or context. The literature on requirements elicitation is moving towards more human-centred approaches to designing explanations (e.g. [24], [25]) but this is not yet standard practice. The next section introduces our architecture, which provides a target for implementing explainable autonomy. We intend this to support and promote further research into requirements elicitation for explainable autonomy.

III. AN EXPLAINABLE AUTONOMY ARCHITECTURE

In this section, we describe our abstract explainable autonomy architecture. Following our suggestion in prior work [12], our architecture (Fig. 1) uses a rational BDI agent to make executive decisions, e.g. choosing the next waypoint or scheduling tasks. The symbolic AI techniques that underpin BDI agents are explicit and based on mathematical logic, so they are (relatively) easy to inspect. We restrict connectionist/sub-symbolic AI (such as Machine Learning or Neural Networks) to more granular functions, e.g. vision classification.

Existing work in the literature explores the utility of BDI agents in providing explanations. For example, by extracting a chain of trace events (stored internally by the agent) that go back through the system’s runtime to explain behaviour [26].

This decision-making Agent and the Explainer component (which builds explanations) form the *Central Executive*¹ Our architecture routes all of the system’s information to this Central Executive, enabling it to collect and store the information needed to produce an explanation. *What* information is needed for explanations is described by the system’s explainability requirements (§ II), but this routing ensures that all of the information is available to the Explainer. Clearly, if a part of the system operates entirely separately from the components that make the decisions and produce the explanations, it will be difficult to explain its behaviour.

¹We borrow the term *Central Executive* from Baddely and Hitch’s Working Memory Model [27].

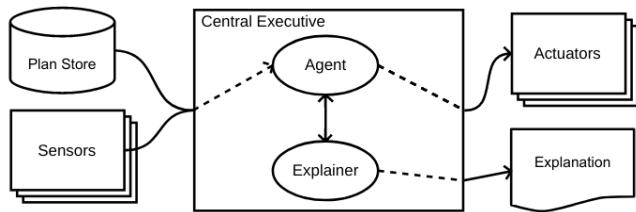


Fig. 1. High-Level view of our abstract architecture. The squares are system modules (which may contain hardware and software), the cylinder is a data store, and the square with the curved base is the explanation ‘document’ (which could be text, speech, etc.). The ovals are sub-modules in the Central Executive. The arrows show the information flow, with the dashed lines for inside the Central Executive.

A. High-Level Architecture

Fig. 1 gives a high-level view of our architecture: an abstracted autonomous robotic system that has sensors, actuators, and a Central Executive. The sensors and actuators enable the system to sense and interact with its environment; both components are likely to be wrapped in software that interprets the raw sensor information or translates commands into low-level actuator control signals, respectively. The system will likely have or build a model of its environment, but this is not shown in Fig. 1 because it is not the focus of our architecture.

Our architecture’s key feature is that it routes all of the system’s information through the Central Executive. This ensures that when an explanation is requested, the required information is available. To avoid bottlenecks, components may directly communicate as long as the Central Executive simultaneously receives the same information.

Here we outline the architecture, but we see several implementation routes. Robotic systems are often implemented using node-based software frameworks [28], the most well-used in the literature is the Robot Operating System (ROS) [29], and others include: G^{en}oM² and NASA’s Core Flight System³. These frameworks often contain common concepts [30], which can enable reuse between frameworks [31]. The modularity in our abstract architecture means that it is framework-agnostic.

B. Central Executive

As previously mentioned, the Central Executive comprises the BDI Agent and an Explainer. These components collaborate to produce explanations.

The BDI Agent makes the system’s executive decisions; it decides on the system’s behaviour, based on sensor input and the user-defined plans. The types of sensors needed will depend on the system’s function and context. The raw data from the sensor hardware is likely to be processed by a software module before being sent to the Agent, which will interpret the information into logical properties that it can reason about (its beliefs). Its other input is a set of plans, written *a priori* for the system’s particular mission and environment⁴. The

Agent chooses a plan, based on its beliefs about the world and other information that it has stored, following it until new information triggers a different plan. The information about which events trigger which plans is used by the Explainer.

The Explainer produces explanations, in response to questions from an operator. The explanation’s format (e.g. textual, graphical, vocal, etc.) is not restricted by our architecture. Both of these features depend on the system’s explainability requirements (§ II). The Explainer stores information from the rest of the system (e.g. the raw sensor information and corresponding beliefs, which plan was triggered and why, etc.) to enable it to explain the system’s behaviour. The context of the question may also change the explanation that is produced. For example: the stakeholder’s role (e.g. from [18]) may alter the level of detail in the explanation or its format; time-pressure on the reply may prompt a terser answer; or the question being about past action or future (in)action.

Work in the literature shows that a BDI approach can be used to provide explanations of an agent’s behaviour. For example, a log of a BDI agent’s goals and behaviour can be used to produce an explanation [20]. Further, the work in [32], which highlights the utility of logic-based autonomy for providing explanations; and [17], which presents an approach to answer “*why?*” and “*why not?*” questions. However, both of these approaches work through debugging features, suggesting that their explanations may be mainly aimed at software developers. While this is undoubtedly useful, the explanations provided may not be suitable for other stakeholders.

When explaining past action, the Explainer can passively collect the information stored about the system and the Agent’s decisions. However, when explaining future inaction, the Explainer needs to be able to trigger the Agent’s reasoning cycle to evaluate a particular goal’s feasibility. This ‘dry-run’ functionality will likely require modifications to an agent to enable the Explainer to identify which plan corresponds to the requested action, and trigger the Agent’s reasoning logic to investigate what is preventing that plan from being selected. § IV gives a more detailed example of this functionality.

C. Verification Benefits

The choice of a BDI Agent enables the verification of the system’s executive decisions, as recommended in our previous work [12, Recipe 1]. In particular, BDI agents are amenable to Formal Methods (FM), mathematically-based techniques for the specification and verification of systems, which can provide strong guarantees about software correctness.

For example, the GWENDOLEN agent language [33] can be exhaustively verified by its *program model checker* the Agent Java PathFinder (AJPF) [34]. A GWENDOLEN program can be checked to prove that its decisions preserve formal properties, written in a temporal logic that has been extended with modalities capturing BDI concepts such as *the agent believes...* so that its reasoning can be checked. The benefit of this approach is that AJPF checks the program code that the

²G^{en}oM: <https://git.openrobots.org/projects/genom3>

³Core Flight System: <https://cfs.gsfc.nasa.gov/>

⁴Our approach should also be applicable to systems that learn new plans, as long as the Agent and Explainer both have access to the plan store.

agent will run during operation⁵; but does so off-line, which does not impact the agent’s performance. For an overview of the available FM approaches for autonomous systems see [28].

An example of a property could be that the agent will ‘eventually believe that it has reached the goal location’, and AJPF can check if this is true for the agent’s program. These temporal logic properties could be translated into natural language [35], to explain them to stakeholders without a formal background. This would help stakeholders to check that the verified properties correspond to the system’s requirements.

IV. EXPLANATION EXAMPLES

Here we present a worked example of how our abstract architecture (§ III) supports explanations of both past and future (in)action. We consider a wheeled-rover using our architecture and Central Executive (§ III-A), performing a remote inspection mission in a nuclear waste store (adapted from [36, §4.2.2]). The rover’s mission is to patrol waypoints inside a nuclear waste store, where low-level radioactive waste is stored in steel drums. The rover: takes radiation readings at each waypoint, to assess the integrity of the storage drums; maintains a given distance from the drums; and must avoid high-radiation areas if it can. The radiation readings and high-radiation areas are reported for later investigation.

To navigate around the waste store, the Agent decides on its plan based on the sensor information and its goal (to take radiation readings at each of the waypoints). As previously mentioned, all of the sensor information is routed through the Central Executive; which makes decisions, sends instructions to the actuators, and collects the information needed by the Explainer component. The explanations are realisable because BDI logics capture the Agent’s beliefs about the world. The Explainer has access to this information, facilitating explanations. Here, we describe how an explanation about past action and future inaction could be realised.

A. Past Action

In this example, the rover is navigating from Waypoint A to Waypoint B and the Agent interprets sensor information as an obstacle. Because the rover should not collide with obstacles, this new sensor information causes it to abort navigating to Waypoint A and navigate to Waypoint C instead.

The rover’s operator might ask “why did you go to Waypoint C instead of Waypoint A?”. Because the Central Executive has: received the sensor information that suggests there is an obstacle *en route* to Waypoint B, interpreted the sensor information into a belief that there is an obstacle, and decided to change the plan; it has the information needed to answer the question. It could explain that “my sensors tell me there is an obstacle ahead of me, between Waypoint A and Waypoint B; and I am not supposed to collide with obstacles.” Here, different forms of explanation could be useful, e.g. the system could explain textually/verbally and provide an image of the perceived obstacle for the operator to check. As previously

mentioned, work like [26] provides a route to realise these explanations through the Agent’s trace of events.

B. Future Inaction

In this example, a nuclear inspector is checking that the rover’s agent will not *choose* to break safety rules, similarly to the expectations of a human worker. Here, the Central Executive could ‘dry-run’ an instruction that should fail, and explain why it cannot be carried out.

For example, the inspector could ask “why can’t you go to Waypoint X?”, where Waypoint X is too close to a storage drum. Here, the Explainer would request the Agent to evaluate the instruction ‘go to Waypoint X’ and interpret the result in much the same way as in § IV-A. As previously mentioned, this requires an Agent with a reasoning cycle modified so that it can be triggered to evaluate a particular instruction.

The Explainer will send the instruction to ‘go to Waypoint X’ to the Agent, triggering it to attempt to enact the relevant plan. When the plan cannot be executed, the feedback from the Agent allows the Explainer to answer the inspector: “I cannot go to Waypoint X because my map tells me that it is closer to a storage drum than I am allowed to go”.

This feature enables safety tests of the Agent’s adherence to safety rules. It would also be extremely useful when testing the autonomous system before it was attached to the robotic system. A simulated environment could be used, and the safety properties could each be queried to provide confidence that the integrated autonomous robotic system will not be dangerous if used in field tests. This ability to ‘dry-run’ actions has appeared in the literature before, e.g. in [37], where they replicate the agent’s reasoning internally to enable it to assess the ethical consequences of its own actions.

V. CONCLUSION AND FUTURE WORK

This paper presents an abstract architecture for enabling explainable autonomy. The architecture routes the system’s information through a Central Executive, which comprises a BDI Agent and an Explainer component. We provide worked examples of how this approach supports explanation of past and future (in)actions.

The major area of future work is to develop and implement our architecture. We intend to use Architecture Analysis and Design Language (AADL) (or a similar notation) to develop a high-level design that is applicable to a variety of robotic software frameworks, exploiting their commonalities [30]. Implementing the design will then enable us to demonstrate the architecture’s utility.

Another important line of future work is to incorporate approaches that elicit explanation requirements, and assess their suitability for particular stakeholders. The approach in [23] provides a useful starting point for involving stakeholders in the design process. The basis of our ongoing requirement elicitation work will be built on our previous study [21], which involves eliciting requirements from nuclear sector stakeholders.

⁵GWENDOLEN executes as ByteCode on a Java Virtual Machine. Nevertheless, AJPF exhaustively verifies the agent’s source code.

REFERENCES

- [1] M. Fisher, R. C. Cardoso, E. C. Collins, C. Dadswell, L. A. Dennis, C. Dixon, M. Farrell, A. Ferrando, X. Huang, M. Jump, G. Kourtis, A. Lisitsa, M. Luckcuck, S. Luo, V. Page, F. Papacchini, and M. Webster, "An overview of verification and validation challenges for inspection robots," *Robotics*, vol. 10, no. 2, 2021. [Online]. Available: <https://www.mdpi.com/2218-6581/10/2/67>
- [2] R. C. Cardoso, M. Farrell, M. Luckcuck, A. Ferrando, and M. Fisher, "Heterogeneous verification of an autonomous curiosity rover," in *NASA Formal Methods Symposium*. Springer, 2020, pp. 353–360.
- [3] R. C. Cardoso, L. A. Dennis, M. Farrell, M. Fisher, and M. Luckcuck, "Towards Compositional Verification for Modular Robotic Systems," in *Proceedings 2nd Workshop on Formal Methods for Autonomous Systems (FMAS) [Virtual]*, ser. EPTCS, vol. 329, 2020, pp. 15–22.
- [4] M. Luckcuck, "Workshop Report: Scoping Certifiable Autonomous Nuclear Robotics," University of Liverpool, Technical Report ULCS-20-001, 2020. [Online]. Available: <https://intranet.csc.liv.ac.uk/research/techreports/tr2020/ulcs-20-001.pdf>
- [5] Luckcuck, Matt, M. Fisher, L. Dennis, S. Frost, A. White, and D. Styles, "Principles for the Development and Assurance of Autonomous Systems for Safe Use in Hazardous Environments," Zenodo, Tech. Rep., Jun. 2021. [Online]. Available: <https://zenodo.org/record/5012322>
- [6] N. Balasubramaniam, M. Kauppinen, K. Hiekkänen, and S. Kujala, "Transparency and Explainability of AI Systems: Ethical Guidelines in Practice," in *Requirements Engineering: Foundation for Software Quality*, V. Gervasi and A. Vogelsang, Eds., vol. 13216. Springer, 2022, pp. 3–18, series Title: Lecture Notes in Computer Science. [Online]. Available: https://link.springer.com/10.1007/978-3-030-98464-9_1
- [7] British Standards Institution, "BS 8611:2016 - Robots and robotic devices. Guide to the ethical design and application of robots and robotic systems," 2016. [Online]. Available: <http://shop.bsigroup.com/ProductDetail?pid=000000000030320089>
- [8] S. Stange and S. Kopp, "Effects of a Social Robot's Self-Explanations on How Humans Understand and Evaluate Its Behavior," in *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*. Cambridge United Kingdom: ACM, Mar. 2020, pp. 619–627. [Online]. Available: <https://dl.acm.org/doi/10.1145/3319502.3374802>
- [9] Luckcuck, Matt, M. Fisher, L. Dennis, S. Frost, A. White, and D. Styles, "Principles for the Development and Assurance of Autonomous Systems for Safe Use in Hazardous Environments," Zenodo, Tech. Rep., Jun. 2021. [Online]. Available: <https://zenodo.org/record/5012322>
- [10] M. E. Bratman, *Intentions, Plans, and Practical Reason*. Harvard University Press, 1987.
- [11] A. S. Rao and M. Georgeff, "BDI Agents: From Theory to Practice," in *Proc. First International Conference on Multi-Agent Systems (ICMAS)*, San Francisco, USA, 1995, pp. 312–319.
- [12] M. Luckcuck, "Using Formal Methods for Autonomous Systems: Five Recipes for Formal Verification," *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 2021. [Online]. Available: <https://doi.org/10.1177/1748006X211034970>
- [13] A. F. T. Winfield and M. Jirotko, "The case for an ethical black box," in *Towards Autonomous Robotic Systems*, Y. Gao, S. Fallah, Y. Jin, and C. Lekakou, Eds. Springer, 2017, pp. 262–273.
- [14] W. Brunotte, L. Chazette, V. Klös, and T. Speith, "Quo Vadis, Explainability? – A Research Roadmap for Explainability Engineering," in *Requirements Engineering: Foundation for Software Quality*, V. Gervasi and A. Vogelsang, Eds., vol. 13216. Springer, 2022, pp. 26–32, series Title: Lecture Notes in Computer Science. [Online]. Available: https://link.springer.com/10.1007/978-3-030-98464-9_3
- [15] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining Explanations: An Overview of Interpretability of Machine Learning," in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*. Turin, Italy: IEEE, Oct. 2018, pp. 80–89. [Online]. Available: <https://ieeexplore.ieee.org/document/8631448/>
- [16] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial Intelligence*, vol. 267, pp. 1–38, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0004370218305988>
- [17] M. Winikoff, "Debugging agent programs with why?: Questions," in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, K. Larson, M. Winikoff, S. Das, and E. H. Durfee, Eds. ACM, 2017, pp. 251–259. [Online]. Available: <http://dl.acm.org/citation.cfm?id=3091166>
- [18] R. Tomsett, D. Braines, D. Harborne, A. Preece, and S. Chakraborty, "Interpretable to whom? a role-based model for analyzing interpretable machine learning systems," in *Proceedings of the ICML Workshop on Human Interpretability in Machine Learning*, 2018. [Online]. Available: <https://arxiv.org/abs/1806.07552>
- [19] M. Hall, D. Harborne, R. Tomsett, V. Galetic, S. Quintana-Amate, A. Nottle, and A. Preece, "A systematic method to understand requirements for explainable ai (xai) systems," in *Proceedings of the IJCAI Workshop on eXplainable Artificial Intelligence (XAI 2019), Macau, China*, vol. 11, 2019. [Online]. Available: https://drive.google.com/file/d/1ma5wilaj31A0d5KC4I2fYaTC_Lqm_d9X/view
- [20] J. Broekens, M. Harbers, K. Hindriks, K. van den Bosch, C. Jonker, and J.-J. Meyer, "Do you get it? user-evaluated explainable bdi agents," in *Multiagent System Technologies*, J. Dix and C. Witteveen, Eds. Springer, 2010, pp. 28–39.
- [21] H. M. Taylor, C. Jay, B. Lennox, A. Cangelosi, and L. Dennis, "Should ai systems in nuclear facilities explain decisions the way humans do? an interview study," in *International Conference on Robot and Human Interactive Communication (RO-MAN 2022)*. IEEE, 2022 (In Press).
- [22] C. Negri-Ribalta, "A method to deal with social bias and desirability in ethical requirements," Joint Proceedings of REFSQ-2022 Workshops, Doctoral Symposium, and Poster & Tools Track, 2022. [Online]. Available: <http://ceur-ws.org/Vol-3122/PT-paper-3.pdf>
- [23] K. Winkle, P. Caleb-Solly, A. J. Turton, and P. Bremner, "Mutual shaping in the design of socially assistive robots: A case study on social robots for therapy," *Int. J. Soc. Robotics*, vol. 12, no. 4, pp. 847–866, 2020. [Online]. Available: <https://doi.org/10.1007/s12369-019-00536-9>
- [24] C. T. Wolf, "Explainability Scenarios: Towards Scenario-Based XAI Design," ser. IUI '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: <https://doi.org/10.1145/3301275.3302317>
- [25] D. Cirqueira, D. Nedbal, M. Helfert, and M. Bezbradica, "Scenario-Based Requirements Elicitation for User-Centric Explainable AI," in *Machine Learning and Knowledge Extraction*, A. Holzinger, P. Kieseberg, A. M. Tjoa, and E. Weippl, Eds. Springer, 2020, pp. 321–341.
- [26] V. J. Koeman, L. A. Dennis, M. Webster, M. Fisher, and K. Hindriks, "The "Why Did You Do That?" Button: Answering Why-Questions for End Users of Robotic Systems," in *Engineering Multi-Agent Systems*, L. A. Dennis, R. H. Bordini, and Y. Lespérance, Eds., vol. 12058. Springer, 2020, pp. 152–172, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-030-51417-4_8
- [27] A. D. Baddeley and G. Hitch, "Working memory," ser. Psychology of Learning and Motivation, G. H. Bower, Ed. Academic Press, 1974, vol. 8, pp. 47–89. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0079742108604521>
- [28] M. Luckcuck, M. Farrell, L. A. Dennis, C. Dixon, and M. Fisher, "Formal Specification and Verification of Autonomous Robotic Systems: A Survey," *ACM Computing Surveys*, vol. 52, no. 5, pp. 1–41, 2019.
- [29] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," in *Workshop on Open Source Software*. IEEE, 2009.
- [30] A. Shakhimardanov, N. Hochgeschwender, and G. Kraetzschmar, "Component models in robotics software," in *Perform. Metrics Intell. Syst. ACM*, 2010, p. 82. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2377576.2377592http://dl.acm.org/citation.cfm?doid=2377576.2377592>
- [31] H. S. Ahn, M. H. Lee, and B. MacDonald, "Design of a Robotic Software Manager for using Heterogeneous Components of Different Frameworks," *J. Softw. Eng. Robot.*, vol. 8, no. 1, pp. 45–64, 2017.
- [32] K. V. Hindriks, "Debugging is explaining," in *PRIMA 2012: Principles and Practice of Multi-Agent Systems*, I. Rahwan, W. Wobcke, S. Sen, and T. Sugawara, Eds. Berlin, Heidelberg: Springer, 2012, pp. 31–45.
- [33] L. A. Dennis and B. Farwer, "Gwendolen: A bdi language for verifiable agents," in *Proceedings of the AISB 2008 Symposium on Logic and the Simulation of Interaction and Reasoning, Society for the Study of Artificial Intelligence and Simulation of Behaviour*, 2008, pp. 16–23.
- [34] L. A. Dennis, "The mcapl framework including the agent infrastructure layer and agent java pathfinder," *The Journal of Open Source Software*, 2018.
- [35] H. Cherukuri, A. Ferrari, and P. Spoletini, "Towards Explainable Formal Methods: From LTL to Natural Language with Neural Machine

Translation,” in *Requirements Engineering: Foundation for Software Quality*, V. Gervasi and A. Vogelsang, Eds., vol. 13216. Springer, 2022, pp. 79–86, series Title: Lecture Notes in Computer Science. [Online]. Available: https://link.springer.com/10.1007/978-3-030-98464-9_7

- [36] T. Wright, A. West, M. Licata, N. Hawes, and B. Lennox, “Simulating ionising radiation in gazebo for robotic nuclear inspection challenges,” *Robotics*, vol. 10, no. 3, 2021. [Online]. Available: <https://www.mdpi.com/2218-6581/10/3/86>
- [37] L. A. Dennis, M. Fisher, and A. F. T. Winfield, “Towards verifiably ethical robot behaviour,” in *Artificial Intelligence and Ethics, Papers from the 2015 AAAI Workshop, Austin, Texas, USA, January 25, 2015*, ser. AAAI Technical Report, T. Walsh, Ed., vol. WS-15-02. AAAI Press, 2015. [Online]. Available: <http://aaai.org/ocs/index.php/WS/AAAIW15/paper/view/10119>