2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

# HEXAR: a Hierarchical Explainability Architecture for Robots

Tamlin Love[1*],Ferran Gebellí[2*],Pradip Pramanick[3*],
Antonio Andriella[1],Guillem Alenyà[1],Anaís Garrell[1],Raquel Ros[4],Silvia Rossi[3]

***Abstract*— As robotic systems become increasingly complex, the need for explainable decision-making becomes critical. Existing explainability approaches in robotics typically either focus on individual modules, which can be difficult to query from the perspective of high-level behaviour, or employ monolithic approaches, which do not exploit the modularity of robotic architectures. We present HEXAR (Hierarchical EXplainability Architecture for Robots), a novel framework that provides a plug-in, hierarchical approach to generate explanations about robotic systems. HEXAR consists of specialised component explainers using diverse explanation techniques (e.g., LLM-based reasoning, causal models, feature importance, etc) tailored to specific robot modules, orchestrated by an explainer selector that chooses the most appropriate one for a given query. We implement and evaluate HEXAR on a TIAGo robot performing assistive tasks in a home environment, comparing it against end-to-end and aggregated baseline approaches across 180 scenario-query variations. We observe that HEXAR significantly outperforms baselines in root cause identification, incorrect information exclusion, and runtime, offering a promising direction for transparent autonomous systems.**

## I. Introduction

Robotic software systems are inherently complex, typically comprising architectures with numerous modules that accomplish diverse capabilities and employ various interfaces [1]. Despite the emerging end-to-end learning approaches [2], [3], [4], where a single black-box module processes sensor data to produce near-final actuator signals, modular architectures remain essential for providing structured, aggregated and meaningful information channels. These internal signals serve as intermediate and efficient means to transmit information, whilst also functioning as internal data representations that facilitate human understanding. For instance, in object navigation tasks, while pure end-to-end approaches may perform well in simulation, modular approaches perform better in real-world scenarios [5], with the additional benefit of informative intermediate signals such as the planned path.

The ability to explain the decisions and behaviours related to these internal processes has been acknowledged as a key factor for improving human understanding of eXplainable Artificial Intelligence (XAI) systems [6] and autonomous robots [7]. However, existing explainability approaches in robotics focus on individual modules such as addressee selection [8], navigation [9], manipulation [10], motion planning [11], or task planning [12]. These approaches typically evaluate isolated scenarios wherein only the module in question is utilised. In the context of a general-purpose robot, end-users lack knowledge of internal robot architectures unless explicitly informed [13] and seek explanations for high-level behaviours, which potentially involve multiple components. Consequently, people tend to pose general "Why" questions [14] (e.g., "Why did the robot not go to the kitchen"), rather than directing their explanation-seeking queries to specific components (e.g., "Why did pose estimation fail?").

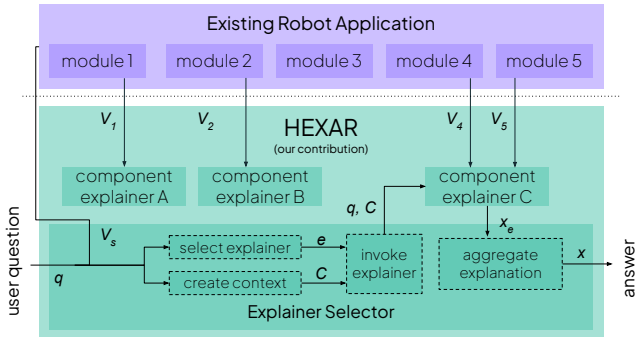


Fig. 1. Our novel framework, HEXAR (see Sec. III), in which different specialised component explainers are orchestrated by an explainer selector, which selects an appropriate component explainer to answer a given query.

There exist some system-wide explainability approaches that employ monolithic methods whereby centralised Large Language Models (LLMs) exploit robot logs and other diagnostic information [15], [16], [17], or that assume a single behaviour tree representation [18]. However, it has been argued that explainability in robotics should aim for modular architectures and hybrid models [19], [20]. Our contribution is HEXAR, a hierarchical explainability framework (Fig. 1). This framework allows a system to utilise different specialised explainers, each corresponding to one or more application components. The system selects the most appropriate explainer based on the user query and other contextual factors. This approach motivates our research question: *Is a hierarchical explanation system, composed of specialised explainer modules orchestrated by a selector module, better at accurately explaining complex and modular robotic systems than equivalent monolithic systems?*

To answer this question, we implement our approach (described in Sec. III) in a home assistant robot use case (Sec. IV) and evaluate the generated explanations across a range of scenarios (Sec. V)[1]. We provide evidence that

*These authors contributed equally
[1]Institut de Robòtica i Informàtica Industrial (CSIC-UPC), Llorens i Artigas 4-6, 08028, Barcelona, Spain
[2]PAL Robotics, Pujades, 77-79, 7-7, 08005 Barcelona, Spain
[3]University of Naples Federico II, Naples, Italy
[4]Artificial Intelligence Research Institute (IIIA-CSIC), Barcelona, Spain

[1]The annotated datasets, implementation and reproducible results are in https://github.com/fgebelli/HEXAR

our approach produces more accurate explanations in less time than two baselines (Sec. VI), laying a foundation for explainable autonomous robot architectures.

## II. Related Work

While the automatic generation of robot explanations has been identified as an important research area [7], many existing approaches target specific robot modules (software components that target specific functionalities) rather than considering entire systems. For example, in [8], robot addressee selection is explained using attention maps, feature importance and decision confidence. In [9], robot navigation is explained using a mix of ontologies, qualitative spatial reasoning [21], and LIME [22], which is a widely used method for producing feature-importance explanations for black-box models. The work in [10] presents an architecture for generating explanations about manipulation tasks employing a Visual Language Model (VLM). In another work [11], motion planning is explained using a templated approach that exploits planning constraints and algorithm parameters. To explain general task plans, the method presented in [12] produces contrastive explanations by generating a contrastive PDDL plan. From these examples and the wider literature [23], it is clear that many approaches (which are usually module-specific) employ very different explanation generation techniques, with some being more appropriate than others for particular functionalities. One important feature of our proposed framework is that it allows the combination of different explanation techniques.

Some other works have proposed broader explanation modules that can explain a full robotic system, usually employing LLMs. In [17], all the robot's logs are collected by an LLM, which can then answer user queries about the system. Although this approach is presented as generic for any robot using ROS 2 [1], it is validated only in a navigation scenario. The REFLECT approach [15] goes further and constructs a robot event summary based on the robot plan, a scene graph built from RGBD images, and audio data. This event summary is then used to verify success for each plan subgoal, and triggers two different LLM prompts depending on whether a subgoal failed (execution analysis) or not (planning analysis). Similarly, the RONAR [16] framework uses the robot plan, an RGBD-based scene graph, and base/joint states to generate narration summaries that can be used to generate explanations about failures. The RACCOON framework [24] generates explanations of system-level behaviour by first selecting relevant robot modules, whose information is passed to an LLM for explanation generation, taking advantage of the application modularity, although all the modules are explained by the same centralised LLM. In another approach [18], the whole robot decision process is assumed to be embedded in a single behaviour tree. Then, given the predefined possible sequences of actions represented by the behaviour tree, either templated or LLM-generated explanations can provide questions to general user queries.

These monolithic approaches do provide explanations to general questions about a robot's behaviour, but do not allow for specialised explainers that target specific robot functionalities and that might employ tailored techniques for the explanation generation. There have been a few initial propositions that go in the direction of such a modular explainability architecture, though they do not fully implement it. The theoretical work in [19] advocates for such an architecture while presenting several integration issues, including how to orchestrate different explainers, how to connect those explainers to the application components, or how to store the relevant data over time. The work in [25] defines and validates an architecture for explainable behaviour generation. Within the requirements, it mentions component-level inspectability and interpretable inter-component communication interfaces. However, the presented architecture remains a unique centralised module that combines information from the decision-making and episodic memory. In COPAL [26], a system architecture orchestrates 3 cognitive levels for reasoning, planning, and motion generation, respectively. Each level is implemented as an LLM agent that interacts with the others. Nevertheless, this hierarchical approach is never applied to explainability, which is mentioned as a feature for future work. Finally, SNAPE [27] formalises the explanation generation process as inherently hierarchical. Nonetheless, the purpose of this hierarchy is to have local Markov Decision Processes (MDPs) provide online explanations adapted to changes in the interaction, while the source explainability information is pre-computed in a global explanation plan.

In this work, we present and validate a framework that takes advantage of a robotic system's modularity to implement a hierarchical explanation generation architecture that is able to answer general questions about a robot's behaviour and decision-making process by selecting the most appropriate specialised component explainers.

## III. Hierarchical Explanation Framework

We propose a novel hierarchical system for explaining a robot's behaviours and decision-making in response to a user query. The system, which we name HEXAR (*Hierarchical EXplainability Architecture for Robots*), is composed of a set of specialised **component explainers** $\mathcal{E}$, which can provide explanations for one or more robot modules (encompassing both high- and low-level modules), and an **explainer selector** $s$, which selects a subset $\mathcal{E}_s \subseteq \mathcal{E}$ of component explainers that should answer a given query $q$. We assume that the robot system emits a sequence of events $\mathcal{V}$. Each component explainer $e$ independently observes a subset $\mathcal{V}_e \subseteq \mathcal{V}$ of events, as does the explainer selector, which observes $\mathcal{V}_s \subseteq \mathcal{V}$. In practice, this observation could be implemented in a publisher-subscriber architecture, such as component explainer nodes listening to topics in ROS. An overview of the presented framework can be seen in Fig. 1.

**Component explainers** are responsible for generating explanations targeting either execution-oriented skills or actions (e.g., navigation, manipulation, human interaction) or higher-level modules of the robot architecture (e.g., task planning, world modelling). We can represent a component explainer as a tuple $e = \langle f_e, \mathcal{V}_e \rangle$, where $f_e(q, C)$ is a

function that takes in a user query $q$ and a context vector $C$ (containing, for example, task information, time windows, etc.) and returns a natural language explanation $x_e$. We note that the interface by which $f_e$ is called is agnostic to its underlying implementation, which could use a diverse set of explanation generation techniques (e.g., counterfactual, feature importance, templated, etc.). The implementation of $f_e$ may involve the invocation of other processes or components. For example, if an interaction module invokes a grasping module to perform object handover, the component explainer assigned to the interaction module could invoke the one assigned to the grasping module, if required.

Necessarily, HEXAR requires a mapping $M$ from the set of robot modules to $\mathcal{E}$. It may be that a single component explainer is responsible for multiple robot modules, perhaps because they can be explained in a similar manner or rely on the same data. It is also possible for multiple component explainers to be assigned to a single robot module, representing different approaches to explaining the module. If it is not required to explain a particular robot module, it is not necessary to implement a corresponding component explainer.

Upon receiving a user query $q$, the **explainer selector** $s$ performs three functions in sequence to produce a natural language explanation $x$. Firstly, it is responsible for selecting an appropriate set $\mathcal{E}_s \subseteq \mathcal{E}$ of component explainers, using some selection function $select(q, \mathcal{V}_s)$. The exact selection logic itself is use case dependent. It may follow a heuristic (e.g., selecting component explainers for modules that have failed) or be determined by a classifier (e.g., an LLM). If multiple component explainers are available for a single robot module, this selection logic may choose between them using information in $\mathcal{V}_s$ (such as different tasks, user types, queries or interaction states).

Additionally, the explainer selector provides a context vector $C$, computed from the query $q$ and observed events $\mathcal{V}_s$, which is passed to the selected component explainers. The context vector may include information such as the relevant task to explain, the time window in which a module is executed, etc.

Finally, after executing $f_e, \quad \forall e \in \mathcal{E}_s$, the explainer selector must aggregate the set of explanations $\{x_e | e \in \mathcal{E}_s\}$ into a single explanation $x$ if $|\mathcal{E}_s| > 1$. The aggregation method may be implemented in a number of ways, for example, using an LLM to summarise the explanation set.

As a complete framework, HEXAR is designed as an add-on to any existing modular robot architecture, making use of already existing information flows ($\mathcal{V}$), such as diagnostics logs or interfaces between application modules. The main advantage of this plug-in approach is that the application implementation and architecture remain the same, without the need for modification. Of course, the quality of explanations may be limited by the quality and level of detail of the information made available by the robot. Depending on the use case, greater explanation coverage may be achieved by augmenting the logging and other diagnostic interfaces of the robot modules. Here, we reinforce the argument for continuously logging key events from sensing, actuation, and decision-making subsystems [19], [28], given that generic data logging can be beneficial to explain unforeseen factors beyond what is anticipated for each module.

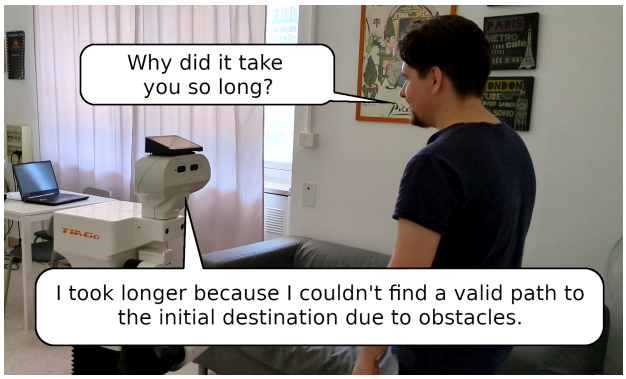


Fig. 2. Use case of a Tiago robot that assists in a home environment.

To design the explainability add-on and decide the specific architecture and required component explainers, several methodologies can be used to identify the user explainability needs and necessary internal information sources, such as co-design [29] or explainability-by-design [30].

## IV. Implementation

To evaluate the proposed framework, we implement it on a physical robotic platform within a home-assistance use case.

### A. Use Case

We use a TIAGo robot, as depicted in Fig. 2, which we equip with 4 skills and a task planner, each implementing a robot module. The skills we use in this evaluation are — navigating to different rooms (*navigation*), speaking (*text to speech*), asking humans for help to complete simple objectives (*ask human for help*), and recommending pizza recipes based on available ingredients (*pizza recommender*). Some skills are themselves complex and hierarchical. For example, the *ask human for help* skill, implemented using a finite state machine, encompasses detecting humans, approaching one, asking for help, and confirming that the human has helped, making use of the *navigation* and *text to speech* skills. Other skills are relatively simple, such as the *pizza recommender*, which is implemented using a decision tree classifier trained on a small dataset of pizza recipes. To this system we add an "*explain*" skill, which triggers the HEXAR framework (see Sec. IV-B).

Using these skills, the robot can perform several assistive tasks such as bringing objects from one room to another, delivering messages to other people, or holding conversations with users. Achieving these tasks is facilitated by a *task planner* [31], which converts a user request to a sequence of skills.

### B. Framework Instantiation

For each of the five robot modules discussed in Sec. IV-A, we design and implement a component explainer that is tailored to answer queries about the specific skill. To

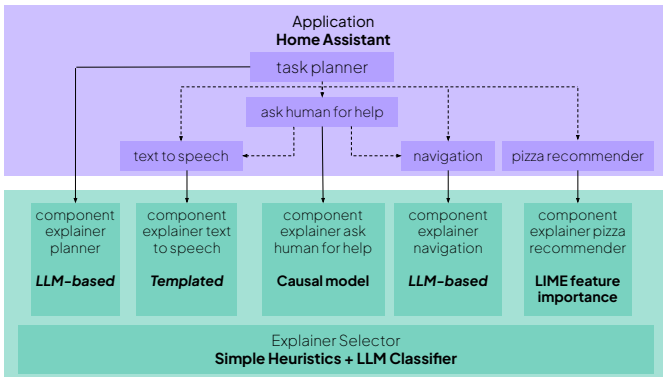


Fig. 3. The HEXAR framework applied to the home assistant use case, showing the dependencies between modules. A component explainer is implemented for each robot module, employing a variety of explanation techniques.

illustrate that our framework can incorporate arbitrary component explainers, each of them uses a different explanation generation mechanism. Fig. 3 provides an overview of the implemented architecture.

Firstly, the *component explainer planner* is responsible for generating explanations for queries related to task planning. We utilise the broad "common sense" reasoning capabilities demonstrated by LLMs, similar to previous works on explanation generation for task planning [15], [16]. In particular, we prompt an LLM to generate the explanation by providing a context that includes the human instruction, generated task plan, and errors in grounding it to a skill sequence (if any), status of each skill (e.g., *waiting, failed*), and the user's explanation-seeking query. This open-ended prompting with the user query allows this component to address various types of plan issues (see Table I for examples).

The component explainer for the *text-to-speech* skill adopts a straightforward, template-based strategy that addresses a single failure condition, namely a skill timeout arising from excessively long utterances. In such instances, the explainer explicitly notifies the user of the timeout event.

Inspired by the work introduced in [17], we implement the component explainer for the *navigation* component with an LLM that receives mainly the logs from the ROS 2 navigation Nav2 package [32]. Repetitive and irrelevant logs are discarded based on the known string patterns. The LLM is given an exhaustive list of examples, including possible failure and suboptimal situations that the logs can reveal, in order to provide explanations for user questions.

To implement the component explainer for the *ask human for help* skill, we adapt the causal model approach of [33], representing a family of approaches for contrastive, causal and counterfactual explanations. The method automatically builds a causal model of the skill execution and queries to generate counterfactual explanations of the form "$Y$ occurred because $X = x$. If $X = x^*$, $Y^*$ would have occurred instead" in response to the query "Why $Y$ and not $Y^*$". In failure cases, explanations provide changes that would result in success. These counterfactual templates are converted to a more natural explanation with an LLM. If no failure occurred, the model is consulted to determine if particular negative conditions were met, such as a high variance in the human detection, for which a templated explanation is given.

The final component explainer, tailored to the *pizza recommender* skill, makes use of LIME [22], a widely-used explainer which ranks input features based on their relevance to a decision. In this case, LIME is used to determine the most relevant ingredient used to recommend a given type of pizza.

To implement the explainer selector, we define $select(q, \mathcal{V}_s)$ as a two-stage process. Firstly, the explainer selector fetches the last available plan (in the form of a sequence of skills and their execution statuses) and checks if one of the skill executions failed or if the plan itself is invalid. If so, the corresponding component explainer is automatically selected (the component explainer planner in the case of invalid plans). Otherwise, if no failure or invalid plan is detected in the skill executions, the user's query is passed to an LLM, which selects the component explainer that corresponds best to the content of the query. Thus, this particular implementation selects only a single component explainer, negating the need for explanation aggregation.

The context vector $C$ consists of the task being explained, the sequence of skills and their return statuses, the plan validity, and the time window of the task. This context is provided to the selected component $e$ explainer when invoking $f_e$.

In this section, we have presented one possible implementation of the general and flexible framework presented in Sec. III. We emphasise that the concrete realisation of the framework can be highly tailored to the specific use case, domain constraints, user requirements and final implementation choices.

## V. Experiments

In this section, we evaluate explanations generated by HEXAR in the use case presented in Sec. IV and compare them against baseline approaches. To demonstrate that HEXAR can run locally, we have used *phi4* [34] (a model with $14B$ parameters that can run on consumer-grade GPUs) for all LLMs within the explanation add-on and application. We further use greedy decoding in the LLM by setting the *temperature* parameter to 0 for the entire experiment to produce deterministic outputs.

### A. Metrics

We employ three metrics to evaluate the explanations. Firstly, we use a *root cause identification* metric which is marked as 1 if the generated explanation contains the ground truth root cause of the failure/behaviour, and 0 otherwise. This metric represents the ability of the explanation system to correctly identify the root cause of a failure or decision.

However, in some explanations, erroneous information is included, potentially in addition to correct information about the root cause. Thus, we also employ the *presence of incorrect facts* metric, which is 1 whenever the explanation contains false information about the failure or task, and 0 otherwise. We merge the above into a combined *explanation*

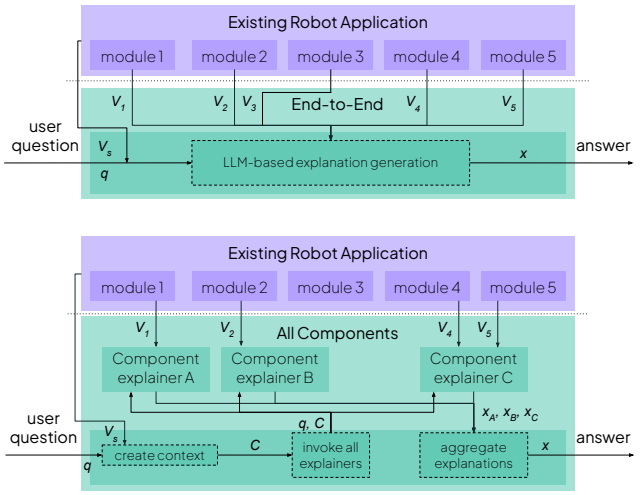


Fig. 4. Evaluated baselines: *end-to-end* (top) and *all components* (bottom).

*accuracy* metric, which is 1 only if *root cause identification* is 1 and *presence of incorrect facts* is 0, and is 0 otherwise.

In addition, we report the runtime required to compute the explanation, ensuring that all variants are evaluated on the same hardware. Specifically for HEXAR, we further measure the *component explainer selection accuracy*, defined as the proportion of instances in which the explainer selector identifies the correct component explainer for a given scenario.

### B. Baselines

Representative of monolithic approaches based on LLMs, the *end-to-end* baseline (Fig. 4 top) uses a single LLM to parse all relevant logs and diagnostic information to produce an explanation in response to a query, similar to the approach in [17]. To ensure a fair comparison, this baseline is given all the same information used by HEXAR, consisting of logs and parameter values. While this approach has access to all the same information, it lacks the specialised explainability of the component explainers and the discriminative power of the explainer selector.

To differentiate between the effects of the component explainers and the explainer selector, we also implement an *all components* baseline (Fig. 4 bottom), which is a modification of HEXAR that retains all the component explainers. However, rather than selectively choosing a single component explainer to provide an explanation as in HEXAR, it triggers all component explainers and then aggregates their outputs into a single explanation using an LLM, prompted only to select the relevant information. In this way, we examine the specific effect of selecting a single component explainer to answer a query.

We expect HEXAR to outperform the two baselines in all the metrics defined in Sec. V-A. We also expect the *end-to-end* baseline to perform worse than *all components*, since it does not benefit from the specialised component explainers.

### C. Test cases

To obtain test cases for evaluation, we begin by identifying situations in the home assistant robot use case that might require an explanation. Following a prior user study [14], we have identified 7 broad categories of such situations that apply to our use case. By mapping each applicable category to a relevant module, we obtain 20 specific scenarios, as shown in Table I. To force the particular failure/behaviour specified by these scenarios, we configured the robot by changing default parameters in modules, such as lowering the timeout in *text to speech* module or the tolerance threshold for variance in person localisation. For other scenarios, we modified the environment at runtime, e.g., by placing obstacles when the robot is navigating. To reliably simulate task planning errors, we manually injected the incorrect plans, bypassing the task planner module. For each scenario, we define a ground truth, a minimal description of the root cause of the failure/sub-optimal behaviour.

For each scenario, we introduce three variations of the user task instruction, which leads to a total of 60 task-specific scenario instances. We then execute the scenario on the TIAGo robot in a studio apartment environment and record all the necessary information required to produce explanations, using the *rosbag* format. The same 60 *rosbag* executions are used for each of the three methods tested.

For each scenario, we introduce variability by providing three distinct explanation-seeking queries, formulated as possible user questions regarding the failure/behaviour of the robot. The queries differ in their level of specificity, from generic (e.g., *"What happened?"*) to more specific queries that either include the task context (*"Why didn't you bring it?"*) or an observed problem (*"Why did it take you so long?"*). By having 3 variations in the query for each of the 60 task-specific scenario instances, we obtain 180 explanations each from HEXAR, *end-to-end* and *all components* baselines, totalling to 540 samples.

We generate the explanations by running the explainability-add on concurrently with the *rosbag* and then providing a user query. To generate explanations, we use an 11th Gen Intel® Core™ i5-11400H @ 2.70GHz × 12 with 16 GB of RAM, where the LLMs are executed in an NVIDIA GeForce RTX 3080 with 12GB of memory.

### D. Annotation Procedure

To compute the metrics described in Sec. V-A, the explanations were labelled by 3 human annotators (co-authors) working independently and with a blind, randomised table including the testcase description, task instruction, user question, ground truth and generated explanation. Before the annotation phase, the 3 annotators discussed the metrics to have a common criterion in potentially ambiguous or corner cases. The final disagreement rate between the annotators was 0.93% for the *root cause identification* metric and 1.30% for the *presence of incorrect facts* metric. In Sec. V-E, we use a final value computed as the majority value across the annotators.

### E. Results

Across all scenarios (see Fig. 5), HEXAR achieved the highest (best) mean *root cause identification* rate of 97%

| # | Category | Relevant Module | Issue/Failure |
|---|---|---|---|
| 1 | Agent Error | Planning | The robot's planner produces a plan with an invalid skill |
| 2 | Agent Error | Planning | The robot's planner produces a plan with invalid parameter names and/or values |
| 3 | Agent Error | Planning | The robot's planner produces a plan which does not fulfil the user's request |
| 4 | Inability | Planning | The robot is instructed to perform a task which it is unable to complete |
| 5 | Unforeseen Circumstances | Navigation | Static obstacles prevent the robot from reaching a desired location |
| 6 | Inability | Navigation | The robot's joystick controller is enabled, overriding autonomous navigation |
| 7 | Inability | Navigation | The robot is plugged into its charger, overriding autonomous navigation |
| 8 | Sub-Optimal Behaviour | Navigation | The robot is badly localised in its map, negatively impacting navigation |
| 9 | Sub-Optimal Behaviour | Navigation | Moving obstacles force the robot to replan its path during navigation execution |
| 10 | Normal/Successful | Navigation | No errors, but the user still questions the robot's movement properties (e.g. speed) |
| 11 | Unforeseen Circumstances | Ask human for help | The robot does not detect anybody that can assist it in completing its task |
| 12 | Inability | Ask human for help | The robot detects someone, but they are too far away to ask for help |
| 13 | Uncertainty | Ask human for help | The robot detects someone, but not long enough for a stable detection |
| 14 | Unforeseen Circumstances | Ask human for help | The robot detects someone, but is unable to approach them for help due to obstacles |
| 15 | Unforeseen Circumstances | Ask human for help | The robot asks someone to assist it, but they refuse |
| 16 | Unforeseen Circumstances | Ask human for help | Someone agrees to help the robot, but does not confirm completion of their assistance |
| 17 | Social Norm Violation | Ask human for help | The robot approaches someone poorly due to suboptimal navigation |
| 18 | Social Norm Violation | Ask human for help | The robot approaches someone poorly due to high variance in the person's detection |
| 19 | Agent Error | Text to speech | The robot's text-to-speech skill times out before its utterance is complete |
| 20 | Normal/Successful | Pizza recommender | The robot explains its choice of pizza with reference to available ingredients |

TABLE I

OVERVIEW OF THE 20 SITUATIONS IN THE EVALUATION DATASET. EACH SITUATION IS CLASSIFIED BY THE SCENARIO CATEGORY AS DEFINED BY WACHOWIAK ET AL. [14] AND BY THE RELEVANT MODULE (SKILL).

($\sigma^2 = 0.16$), compared to the 73% ($\sigma^2 = 0.45$) obtained by the end-to-end baseline and the 92% ($\sigma^2 = 0.27$) obtained by the all components baseline. For the *presence of incorrect facts* metric, HEXAR achieves the lowest (best) score at 7% ($\sigma^2 = 0.26$), followed by the end-to-end baseline at 28% ($\sigma^2 = 0.45$) and the all components baseline at 32% ($\sigma^2 = 0.47$). For the combined *explanation accuracy* metric, HEXAR achieves the highest (best) score at 93% ($\sigma^2 = 0.26$), followed by the all components baseline at 67% ($\sigma^2 = 0.47$) and the end-to-end baseline at 66% ($\sigma^2 = 0.48$). We perform Cochran's Q test ($df = 2$) for the *root cause identification* ($Q = 60.04, p < 0.001$), *presence of incorrect facts* ($Q = 45.50, p < 0.001$), and *explanation accuracy* ($Q = 50.85, p < 0.001$) metrics, followed by pairwise McNemar tests with Holm correction to determine the statistical significance of results.

Considering only the *explanation accuracy* metric, we also compare performance across the five robot modules discussed in Sec. IV. The same statistical tests are conducted, and results are presented in Fig. 6.

Additionally, we note that across the 180 explanations, HEXAR correctly selects the appropriate component explainer in 179 cases, resulting in a *component explainer selection accuracy* of 99.44%.

Finally, across all scenarios, we report a mean runtime of 1.73 seconds ($\sigma^2 = 1.66$) for HEXAR, compared to 7.86 seconds ($\sigma^2 = 2.04$) for the end-to-end baseline and 10.05 seconds ($\sigma^2 = 1.62$) for the all components baseline.

## VI. DISCUSSION

We now use the results described in the previous section to compare and contrast HEXAR against two baselines. It is worth noting that these results are obtained from a single use case and implementation, and other HEXAR implementations and use case particularities will necessarily alter the explanation generation performance. However, we assume that many of the patterns and considerations from our evaluation will continue to be valid, since they are related to the way information flows are structured and not particular to specific requirements, constraints or technologies.

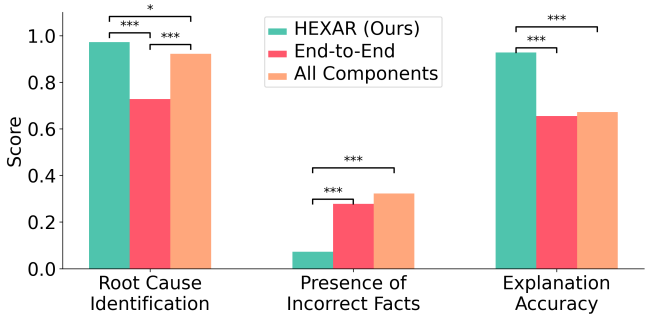


Fig. 5. Mean scores for each method, grouped by metric. For each pairwise difference, $*$ denotes $p < 0.05$ and $***$ denotes $p < 0.001$. Unlabelled pairs lack statistically significant difference.

Results indicate that the *end-to-end* baseline has a significantly lower *root cause identification* value than the other two versions, which we attribute to using "raw" information from the application in a centralised LLM instead of processing it through specialised modules that are better at identifying the root causes for particular modules. The *all-components* baseline provides a lower value compared to HEXAR, although it is somewhat higher than the *end-to-end* baseline. This indicates that although the relevant component explainer response is necessary to find the root cause, irrelevant explanations from other components can add noise, which may obscure the root cause.

Considering the *presence of incorrect facts* metric, the *all components* baseline performs the worst, with similar values

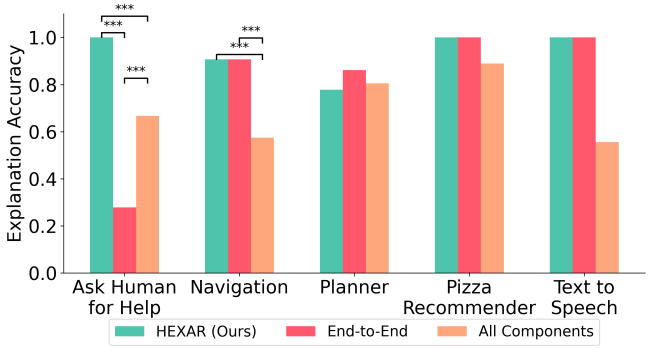


Fig. 6. Mean *explanation accuracy* for each method, grouped by relevant robot module. All pairwise differences are not statistically significant, except those labelled $***$, for which $p < 0.001$.

to the *end-to-end* baseline. These results suggest that any extra irrelevant context, either first processed and then aggregated or directly aggregated, usually injects explanations with incorrect information, regardless of whether the explanation also points to the correct root cause. For example, in the scenario 7 from Table I, HEXAR correctly produced "I couldn't navigate because the robot is charging, which disables autonomous navigation.", while the *all components* baseline added a secondary, wrong statement about the high-level plan which was incorrect: "The reason I didn't bring it [...] autonomous navigation was disabled while I was charging. Additionally, my plan was incorrect as I attempted to go directly to the living room without first navigating to the kitchen to pick up the coffee."

Regarding the consolidated *explanation accuracy* metric, it can be negatively affected either because of a missing root cause or by including incorrect information. Results indicate that the metric is higher for HEXAR compared to the two baselines. These baselines perform relatively similarly, but for the *all components* baseline, the *presence of incorrect facts* is mainly responsible for the drop in accuracy, while in the case of the *end-to-end* baseline, it is the *root cause identification* that is the main contributor.

With respect to the runtime, results confirm that first selecting the relevant component explainer and executing only that dedicated module is more efficient than executing every single component explainer and then aggregating the information (*all components* baseline) or having an LLM with a very extended context (*end-to-end* baseline).

When comparing the five robot modules (Fig. 6), no baseline consistently outperforms HEXAR across any category with any statistical significance. The *end-to-end* baseline yields competitive performance in tasks where component explainers are LLM-based (i.e., navigation, planning) or where the explanations are relatively simple and repetitive (i.e., pizza recommender, text to speech). In contrast, this baseline exhibits difficulties in explaining more complex skills (ask human for help), where a tailored causal model provides superior identification of the actual root causes. For the *all components* baseline, the poor performance in most categories other than planning—with the exception of the pizza recommender—is primarily attributable to the planner component introducing non-existent issues in the high-level plan, which are then incorporated into the final explanations alongside the correct root causes.

Finally, when analysing specifically the HEXAR performance, we corroborate that the high *component explainer selection accuracy* has been key to achieving the positive results. Only in one experiment was the wrong component explainer selected. In scenario 3 (Table I), where the planner incorrectly misses a step where the robot navigates to the living room in order to deliver a book, the system was asked "Why didn't you go to the living room?". The system responded with "I do not have enough information to answer this question.", wrongly selecting the navigation component explainer instead of the planning component explainer.

## VII. Limitations and Future Work

There are several promising directions for extending this work. In the current implementation, inter-dependence between component explainers was relatively limited. Future research could explore more complex instances of HEXAR, designed to resemble robotic applications with additional modules, deeper inter-dependencies, greater structural depth, and more complex selection algorithms. Such extensions might involve assigning multiple component explainers to more complex modules, enabling explainers to invoke one another when appropriate, and systematically comparing the behaviour of different LLM models within this framework. Future work could also investigate how well HEXAR and derived frameworks scale to larger, more complex hierarchical robot architectures.

In this work, we considered only textual (i.e., natural language) explanations. However, many techniques produce explanations in other modalities, and future implementations should attempt to integrate these approaches into HEXAR. For example, to explain human detection error or uncertainty, visual explanation components such as bounding boxes can be coherently combined with a textual component to explain how a lower-level module leads to task failure.

The evaluation procedure and metrics used in this work allowed us to objectively determine which systems produced accurate explanations containing root causes and excluding incorrect information, thus sufficiently answering our research question. In the future, user studies can be performed to further evaluate subjective attributes and the effects of explanations on users.

Finally, beyond providing explanations, HEXAR could be extended to support replanning, reactive failure recovery and prevention, as well as the generation of corrective action recommendations for the user.

## VIII. Conclusions

We have presented HEXAR, a hierarchical explainability architecture for robots that addresses the challenge of explaining modular robotic systems through specialised component explainers orchestrated by a selector. Our

framework provides a plug-in approach that leverages existing robot interfaces.

We provide an example implementation of the framework for a real robot performing assistive tasks in a home environment. We provide evidence that HEXAR significantly outperforms two baselines: an end-to-end approach using a single LLM and another approach that first invokes all component explainers and then aggregates their outputs. Based on 20 scenarios where situations requiring explanations are recreated, we validate that HEXAR is more effective at identifying root causes and avoiding incorrect information while being more time-efficient, affirmatively answering our research question. Our approach not only enhances explanation accuracy but also facilitates the integration of heterogeneous explanation techniques tailored to various robot capabilities with different complexities and dependencies. As robotic systems continue to grow in complexity, hierarchical explainability architectures offer a promising path toward transparent autonomous robots.

## ACKNOWLEDGEMENTS

This work has been supported by Horizon Europe Marie Skłodowska-Curie grant No. 101072488 (TRAIL), the Horizon Europe CoreSense grant No. 10107025 (CoreSense) and HORIZON-CL4-2024-DIGITAL-EMERGING-01-101189557 (TORNADO).

## REFERENCES

[1] S. Macenski, T. Foote, B. Gerkey, *et al.*, "Robot operating system 2: Design, architecture, and uses in the wild," *Science robotics*, vol. 7, no. 66, p. eabm6074, 2022.

[2] Y. Ma, Z. Song, Y. Zhuang, *et al.*, "A survey on vision-language-action models for embodied AI," *arXiv preprint arXiv:2405.14093*, 2024.

[3] M. J. Kim, K. Pertsch, S. Karamcheti, *et al.*, "OpenVLA: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024.

[4] G. R. Team, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, T. Armstrong, A. Balakrishna, R. Baruch, M. Bauza, M. Blokzijl, *et al.*, "Gemini robotics: Bringing ai into the physical world," *arXiv preprint arXiv:2503.20020*, 2025.

[5] T. Gervet, S. Chintala, D. Batra, *et al.*, "Navigating to objects in the real world," *Science Robotics*, vol. 8, no. 79, p. eadf6991, 2023.

[6] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial Intelligence*, vol. 267, pp. 1–38, 2019.

[7] S. Anjomshoae, A. Najjar, D. Calvaresi, and K. Främling, "Explainable agents and robots: Results from a systematic literature review," in *International Conference on Autonomous Agents and Multi-Agent Systems*, pp. 1078–1088, 2019.

[8] I. Bečková, Š. Pócoš, G. Belgiovine, *et al.*, "A multi-modal explainability approach for human-aware robots in multi-party conversation," *Computer Vision and Image Understanding*, vol. 253, p. 104304, 2025.

[9] A. Halilovic and F. Lindner, "Visuo-textual explanations of a robot's navigational choices," in *Companion of the International Conference on Human-Robot Interaction*, pp. 531–535, 2023.

[10] J. Duan, W. Pumacay, N. Kumar, *et al.*, "Aha: A vision-language-model for detecting and reasoning over failures in robotic manipulation," *arXiv preprint arXiv:2410.00371*, 2024.

[11] M. Brandao, G. Canal, S. Krivić, and D. Magazzeni, "Towards providing explanations for robot motion planning," in *International Conference on Robotics and Automation*, pp. 3927–3933, IEEE, 2021.

[12] B. Krarup, M. Cashmore, D. Magazzeni, and T. Miller, "Model-based contrastive explanations for explainable planning," in *International Conference on Automated Planning and Scheduling*, 2019.

[13] L. Hindemith, C. B. Wiebel-Herboth, H. Wersing, *et al.*, "Improving HRI through robot architecture transparency," *International Journal of Social Robotics*, pp. 1–21, 2025.

[14] L. Wachowiak, A. Fenn, H. Kamran, *et al.*, "When do people want an explanation from a robot?," in *International Conference on Human-Robot Interaction*, pp. 752–761, 2024.

[15] Z. Liu, A. Bahety, and S. Song, "Reflect: Summarizing robot experiences for failure explanation and correction," in *Conference on Robot Learning*, pp. 3468–3484, PMLR, 2023.

[16] Z. Wang, B. Liang, V. Dhat, *et al.*, "I can tell what I am doing: Toward real-world natural language grounding of robot experiences," in *Conference on Robot Learning*, pp. 1863–1890, PMLR, 2025.

[17] D. Sobrín-Hidalgo, M. A. González-Santamarta, Á. M. Guerrero-Higueras, *et al.*, "Explaining autonomy: Enhancing human-robot interaction through explanation generation with large language models," *arXiv preprint arXiv:2402.04206*, 2024.

[18] G. LeMasurier, C. Tagliamonte, J. Breen, *et al.*, "Templated vs. generative: Explaining robot failures," in *International Conference on Robot and Human Interactive Communication*, pp. 1346–1353, IEEE, 2024.

[19] M. Winikoff, "Towards engineering explainable autonomous systems," in *International Workshop on Engineering Multi-Agent Systems*, pp. 144–155, Springer, 2024.

[20] A. S. Adebayo, O. O. Ajayi, and N. Chukwurah, "Explainable AI in robotics: A critical review and implementation strategies for transparent decision-making," *Journal of Robotics and AI Systems*, vol. 12, no. 4, pp. 101–118, 2024.

[21] C. Freksa, "Qualitative spatial reasoning," in *Cognitive and linguistic aspects of geographic space*, pp. 361–372, Springer, 1991.

[22] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why should I trust you?" Explaining the predictions of any classifier," in *International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144, 2016.

[23] D. Sobrín-Hidalgo, Á. M. Guerrero-Higueras, and V. Matellán-Olivera, "Generating explanations for autonomous robots: a systematic review," *IEEE Access*, 2025.

[24] S. Bustamante, M. Knauer, J. Thun, *et al.*, "Raccoon: Grounding embodied question-answering with state summaries from existing robot modules," in *International Conference on Robotics and Automation*, pp. 4322–4329, IEEE, 2025.

[25] S. Stange, T. Hassan, F. Schröder, *et al.*, "Self-explaining social robots: An explainable behavior generation architecture for human-robot interaction," *Frontiers in Artificial Intelligence*, vol. 5, p. 866920, 2022.

[26] F. Joublin, A. Ceravola, P. Smirnov, *et al.*, "CoPAL: corrective planning of robot actions with large language models," in *International Conference on Robotics and Automation*, pp. 8664–8670, IEEE, 2024.

[27] A. S. Robrecht and S. Kopp, "SNAPE: A sequential non-stationary decision process model for adaptive explanation generation.," in *International Conference on Agents and Artificial Intelligence*, pp. 48–58, 2023.

[28] A. F. Winfield and M. Jirotka, "The case for an ethical black box," in *Conference Towards Autonomous Robotic Systems*, pp. 262–273, Springer, 2017.

[29] F. Gebellí, R. Ros, S. Lemaignan, and A. Garrell, "Co-designing explainable robots: A participatory design approach for HRI," in *International Conference on Robot and Human Interactive Communication*, pp. 1564–1570, IEEE, 2024.

[30] T. D. Huynh, N. Tsakalakis, A. Helal, *et al.*, "Explainability-by-design: A methodology to support explanations in decision-making systems," *arXiv preprint arXiv:2206.06251*, 2022.

[31] S. Cooper, R. Ros, S. Lemaignan, *et al.*, "Demonstration of an open-source ROS 2 framework and simulator for situated interactive social robots," in *International Conference on Human-Robot Interaction*, pp. 1770–1772, IEEE, 2025.

[32] S. Macenski, F. Martin, R. White, and J. Ginés Clavero, "The marathon 2: A navigation system," in *International Conference on Intelligent Robots and Systems*, 2020.

[33] T. Love, A. Andriella, and G. Alenyà, "Temporal counterfactual explanations of behaviour tree decisions," *arXiv preprint arXiv:2509.07674*, 2025.

[34] M. Abdin, J. Aneja, H. Behl, *et al.*, "Phi-4 technical report," *arXiv preprint arXiv:2412.08905*, 2024.