

# Explaining LLM Adaptation Decisions in Self-Adaptive Systems via Faithful Input Attribution

Balint Mate<sup>✉</sup>

FZI Research Center for Information Technology  
Karlsruhe, Germany  
Karlsruhe Institute of Technology  
KASTEL  
Karlsruhe, Germany  
[mate@fzi.de](mailto:mate@fzi.de)

Diego Perez-Palacin

Linnaeus University  
Växjö, Sweden  
Karlsruhe Institute of Technology  
KASTEL  
Karlsruhe, Germany  
[diego.perez@lnu.se](mailto:diego.perez@lnu.se)

Vincenzo Scotti

Karlsruhe Institute of Technology  
KASTEL  
Karlsruhe, Germany  
[vincenzo.scotti@kit.edu](mailto:vincenzo.scotti@kit.edu)

Raffaella Mirandola

Karlsruhe Institute of Technology  
KASTEL  
Karlsruhe, Germany  
[raffaella.mirandola@kit.edu](mailto:raffaella.mirandola@kit.edu)

## Abstract

Large language models (LLMs) are increasingly investigated as decision-making components in MAPE-K self-adaptive loops. For such autonomous behavior to be trustworthy, stakeholders need evidence with which to assess and justify confidence in its decisions. We envision a modular and extensible platform for explaining adaptation decisions generated by LLMs. Each platform module checks one candidate hypothesis about what a decision depends on, operationalizes it with one or more explanation methods, and evaluates the explanation for faithfulness and plausibility. We identify four initial hypotheses with their respective modules and implement one module, which uses replay-based input attribution to trace each recorded action to the information that supports it, separating which component to act on from what to do to it. We evaluate the module with three LLM planners on a well-established self-adaptive-system exemplar. The results distinguish system-wide dependence in component selection from stronger component grounding in action specification.

## CCS Concepts

• **Computing methodologies** → **Artificial intelligence**; • **Software and its engineering** → *Software design engineering*.

## Keywords

Large Language Model, Self-Adaptive Systems, Explainability, Trustworthiness

## ACM Reference Format:

Balint Mate, Vincenzo Scotti, Diego Perez-Palacin, and Raffaella Mirandola. 2026. Explaining LLM Adaptation Decisions in Self-Adaptive Systems via



This work is licensed under a Creative Commons Attribution 4.0 International License. TRUST '26, Munich, Germany

© 2026 Copyright held by the owner/author(s).  
ACM ISBN 979-8-4007-2994-2/2026/10  
<https://doi.org/10.1145/3843782.3844659>

Faithful Input Attribution. In *Proceedings of the 1st International Workshop on Trustworthy and Responsible Autonomous Systems (TRUST '26)*, October 12–16, 2026, Munich, Germany. ACM, New York, NY, USA, 8 pages. <https://doi.org/10.1145/3843782.3844659>

## 1 Introduction

Large Language Models (LLMs) are increasingly investigated as decision-making components in Self-Adaptive Systems (SAS), with potential application and roles throughout the MAPE-K feedback loop [8]. Existing approaches range from merging Analyze and Plan into an LLM-based synthesis step [2] to embedded and agentic integrations [7] (an embedded LLM chooses adaptations within the feedback loop, whereas an agentic LLM manages the loop end to end). In either case, the basis for an adaptation becomes harder to inspect as the LLM replaces explicit decision logic.

Making the basis of adaptation decisions inspectable is directly relevant to trustworthiness, as explanations can help stakeholders assess and diagnose autonomous behavior and provide evidence for trustworthiness assessments [11]. In this work, we focus on decisions made by the embedded planner architecture because it provides a well-bounded explanation target, letting us study isolated decision dependencies without first having to disentangle the combined Analyze and Plan or the entire agent trajectory.

When integrating an LLM in the managing system, the inputs and outputs of the component are textual information. Optionally, a description of the task and context, information about the current managed system, or the outcome of past decisions may be given in the input text. Producing a syntactically valid output may be difficult; therefore, few-shot examples may be inputted as well. While feeding in more information may be necessary for the LLM to make proper decisions, every piece of input can lead to undesired behavior. Adaptation rules may be ignored or misunderstood, the current state may be misread or left unused, past decisions may be copied over, and few-shot examples could be memorized rather than used only as a format guide. Knowing what a decision depends on

is therefore what makes such an LLM-based component auditable and debuggable.

Explaining the planner is not possible with a single measurement: what its decisions depend on splits into distinct questions such as whether it uses the current state at all, whether it uses the right part of it, etc. We envision a reusable, modular and extensible *platform* for explaining decisions generated by LLMs that turns each such question into a *module*: each module checks one hypothesis by operationalizing it with one or more explanation methods, then validating the explanation for faithfulness and plausibility.

We identify four initial hypotheses and implement and evaluate one module in this work, which focuses on the following question: assuming an LLM-based planner also conditions on the current state, does it rely on the right part of it? We engineer an input-attribution method for structured adaptation plans and separately analyze component selection and action specification to answer this question. We require the attributions to be *faithful*, as in: the LLM planner’s output must depend on the state fields the attribution credits. Given a faithful attribution, we next assess whether the attributed dependencies are *plausible* with respect to domain knowledge [6, 15].

We answer the following research questions:

- **RQ1** Do the module’s input-attribution rankings faithfully reflect how the recorded plan depends on the system state?
- **RQ2** How is the planner’s state dependence distributed across components during component selection and action specification, and is stronger component-level attribution locality associated with agreement with the domain rule?

This work makes the following contributions.

- We envision an extensible platform for explaining adaptation decisions made by LLMs, organized as one module per hypothesis about what a decision depends on and a shared diagnostic pipeline in which each module operationalizes its hypothesis and evaluates the explanation with respect to faithfulness and plausibility, thereby providing evidence for assessing confidence in the planner. We identify four initial hypotheses and implement one module.
- For the chosen module, we introduce a span-separated replay attribution method for structured LLM plans by separating which component the planner acts on from what it does to that component, and evaluate the explanations for both faithfulness and plausibility.
- We demonstrate the application of the module on the DeltaIoT exemplar across three LLMs.

We organize the remainder of this paper into the following sections. Section 2 introduces the running example, Section 3 discusses related work, and Section 4 presents the modular explanation approach. Section 5 describes the implementation of the selected *State*-dependency module, and Section 6 evaluates it. Finally, Section 7 discusses threats to validity, and Section 8 concludes the paper.

## 2 Running Example

We use DeltaIoT [5] as the running example throughout. This well-established SAS exemplar is a battery-powered multi-hop wireless sensor network.

We select this exemplar primarily because it provides a controlled environment for evaluating the explanation method while an LLM planner in this setting promises to handle unanticipated changes without an expert extending the adaptation logic each time [8].

The simulator configuration has 15 nodes, 14 transmitting sensor nodes relay data to one gateway over 17 directed links. In this configuration, adaptation trades off two objectives exposed to the deployed planner, energy consumption and packet loss, which must both be minimized. The controllable and observed state is per link:

- Signal-to-Noise Ratio (SNR) quantifies link quality, with negative values indicating a very noisy link and positive values a cleaner signal.
- Transmission power is an integer in  $[0, 15]$ . Increasing power over a link raises SNR but costs energy.
- Forwarding distribution values over multiple outgoing links of the same mote specify the ratio of the traffic routed through each link.

The exemplar also reports the two global metrics (energy and packet loss).

The deployed planner prompt consists of four blocks:

- (1) *Task description*: the problem description and the adaptation rules.
- (2) *Demonstration*: a few-shot example (drawn once per run and fixed throughout the run).
- (3) *Memory*: a short sliding window of the previous status snapshots and the LLM’s own previous plans.
- (4) *State*: the current per-link status plus the two global metrics.

The planner outputs adaptation actions as text, one per line, at link level (increase or decrease power or distribution) or mote level (equalize distribution); Fig. 1 shows examples.

## 3 Related Work

Our work sits at the intersection of explainability for SAS, LLMs used as adaptation components, and post-hoc attribution with faithfulness evaluation. Our choice of an input-attribution method follows the survey of explainability for tool-augmented LLMs and agents by Roth et al. [15]. Explainability in SASs is a small but methodologically diverse research area [19]. The works closest to ours explain a runtime adaptation decision, but for hand-built models, probabilistic checkers, learned classifiers, or reinforcement-learning policies rather than an LLM: the nearest are the explanation-driven adaptation technique XDA-II, which applies analysis of partial-dependence and feature-importance to embedded predictive models [13], and the explainable reinforcement-learning technique XRL-DINE, which explains a learned planner with a user study [3, 12]. None explains an LLM planner.

LLMs inside the MAPE-K loop are a recent development. Li et al. [8] identify planner integration and decision-model explanation as emerging topics. Donakanti et al. [2] synthesize adaptation plans without an explanation mechanism, and Vázquez et al. [21] adapt how a plan is *presented* to a user but do not study why the planner chose it. The embedded-planner experiments reported by Kara et al. [7] provide validated actions and closed-loop measurements, but not input-level explanations of individual decisions. We are not aware of prior work that attributes an LLM planner’s in-loop decision to its structured input state.

| Task description           |                                                                                                                                                                                                                                                                                       |
|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Problem description</b> | DeltaIoT is a battery-powered multi-hop network of 15 motes over 17 links. Global metrics: Packet Loss; Energy Consumption. Per-link state: SNR; Power; Forwarding Probability.                                                                                                       |
| <b>Instructions</b>        | Always check the current Power value. Do not generate 'increase power' if Power is already 15. Be selective: target only the links that need it.                                                                                                                                      |
| <b>Adaptation rules</b>    | Negative SNR: increase power only if current Power < 15. Positive SNR: decrease power only if current Power > 0. Route more traffic toward the link with higher SNR.                                                                                                                  |
| <b>Output format</b>       | Express each action on its own line:<br>- Link [src] -> [dest] increase/decrease power / distribution<br>- Mote [n] equalize distribution                                                                                                                                             |
| <b>Demonstration(s)</b>    | <pre> [User]: ## Example 1 ### Step 68 Metrics:   Energy: 15.8;   Packet loss: 11.8%  Mote 4:   Link -&gt; 1:     SNR: -0.1;     Power: 12;   ... [LLM]: Adaptation plan:   - Link 4 -&gt; 1 increase power;   ... [User]: ### Step 69 ... [LLM]: ... [User]: ## Example 2 ... </pre> |
| <b>Memory</b>              | <pre> [User]: ## Running ### Step 41 Metrics:   Energy: 28.0;   Packet loss: 1.8%  Mote 12:   Link -&gt; 7:     SNR: 7.4;     Power: 14;   ... [LLM]: Adaptation plan:   - Link 12 -&gt; 7 decrease power;   - Mote 10 equalize distribution;   ... </pre>                            |
| <b>State</b>               | <pre> [User]: ### Step 42 Metrics:   Energy: 34.1;   Packet loss: 16.2%  Mote 12:   Link -&gt; 7:     SNR: 7.1;     Power: 13;     Forwarding Probability: 70%;   Link -&gt; 3:     SNR: -4.0;     Power: 15;     Forwarding Probability: 30%;   ... </pre>                           |
| <b>Planner output</b>      | <pre> [LLM]: Adaptation plan:   - Link 12 -&gt; 3 decrease power;   - Link 14 -&gt; 12 decrease power;   - Mote 7 equalize distribution;   ... </pre>                                                                                                                                 |

**Figure 1: The deployed planner prompt (solid-line boxes) and output completion (dashed-line box).**

Methodologically, the closest analogue is explainable reinforcement learning, which explains a state-to-action mapping. Greydanus et al. [4] perturb regions of the state and read the change in the policy distribution, the direct precedent for our field-level occlusion, while Septon et al. [18] motivate reporting local decision explanations together with aggregate summaries across decisions. Because we attribute a generated *sequence* rather than a single label, our per-action, per-span target is inspired by sequence attribution. Yin and Neubig [22] show that per-token “why token A instead of

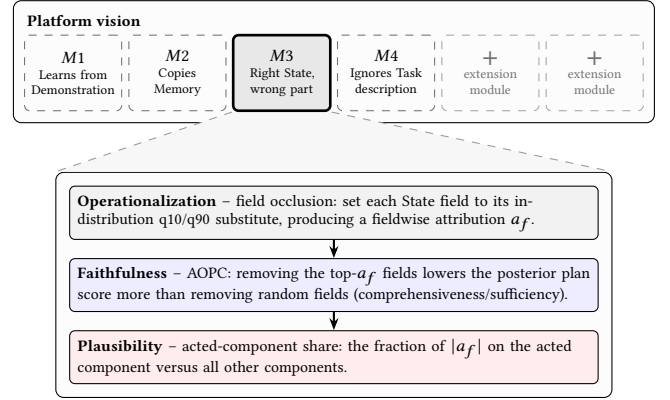


**Figure 2: Envisioned modular and extensible platform for explaining LLM adaptation decisions with focus on the implemented and evaluated module M3.**

B” attribution is the informative target for generation, with tooling from Sarti et al. [17]. The underlying perturbation machinery is the standard LIME/SHAP family [14, 9].

Finally, we evaluate explanations rather than assume them. We test faithfulness with the comprehensiveness/sufficiency protocol of DeYoung et al. [1], and adopt the plausibility-versus-faithfulness distinction of Jacovi and Goldberg [6] to report both the faithfulness and plausibility of the explanation. The contribution of this work is therefore the structured, span-separated instantiation and evaluation of attribution for an LLM *planner* in a MAPE-K loop.

## 4 Modular Explanation Approach

In this section, we outline our explainability approach. First, we propose our modular and extensible diagnostic decomposition to identify the dependency hypothesis to examine. Then we introduce the managed-system decomposition required to examine the selected hypothesis.

### 4.1 Diagnostic Decomposition

An LLM planner can use information from the four prompt blocks shown in Fig. 1: the *Task description*, *Demonstration*, *Memory*, and *State*. Identifying which block drives its decisions permits better assessment and narrows the possible failure mechanism. Our approach organizes this diagnosis into four hypotheses (Fig. 2), each representing a possible influencing factor on the planner’s behavior.

- (1) The LLM learns solely from the *Demonstration*, predicting from it rather than the *State*.
- (2) The LLM follows its own previous plan, copying the *Memory* rather than generating an appropriate plan based on the *State*.
- (3) The LLM correctly focuses on the *State*; however, it relies on the wrong part of it.
- (4) The LLM attends to the right part of the *State* but does not apply the adaptation rule provided in the *Task description*.

Each hypothesis defines a module in the envisioned platform that operationalizes the hypothesized dependency with one or more explanation methods and evaluates the explanation for faithfulness and plausibility.

## 4.2 Managed-System Decomposition

This paper implements module *M3* and leaves the rest of the platform to future work. Analyzing which part of the *State* influences a decision requires a corresponding decomposition of the managed system and its adaptation plan.

We use *component* for an identifiable part of the managed system whose state can be represented by a group of fields. An *action on a component* is a plan instruction that identifies such a component and specifies a change to apply to it. The method applies to managed systems whose *State* fields can be assigned to identifiable components and whose structured actions can be mapped to an affected component and change.

## 5 Explaining State Dependence across Components

In this section, we describe the implementation of the module *M3* detailed in Fig. 2 in terms of posterior plan scoring, field-level attribution, faithfulness, and plausibility.

### 5.1 Posterior Plan Scoring

For a tokenized prompt  $X$ , let  $Y = (y_1, \dots, y_T)$  be an output of  $T$  tokens. We score its  $t$ -th token  $y_t$  as

$$\ell_t(X, Y) = \log P_\theta(y_t \mid X, Y_{<t}), \quad (1)$$

where  $P_\theta$  is the conditional token distribution of the LLM with parameters  $\theta$ , and  $Y_{<t} = (y_1, \dots, y_{t-1})$  is the output prefix preceding  $y_t$ . A target set  $A = \{y_{i_1}, \dots, y_{i_m}\} \subseteq Y$  selects the tokens to be explained. We obtain its aggregate score by summing the token scores,

$$s_A(X, Y) = \sum_{y_t \in A} \ell_t(X, Y). \quad (2)$$

Taking  $A = Y$  gives the posterior plan score, that is, the log-probability the LLM assigns to the whole output conditioned on  $X$ . Restricting  $A$  to a subset of  $Y$  gives the score of a particular output span.

For a recorded decision, let  $X_0$  denote the exact deployed prompt and  $Y_0$  the verbatim recorded output through its last identified action, retaining text before and between actions. During attribution,  $\theta$  and  $Y_0$  are fixed while only  $X$  is perturbed. Every token score therefore uses the recorded prefix  $Y_{0,<t}$ , and the LLM does not generate a new plan. This preserves the decision under explanation, avoids stochastic or decoding-induced differences between regenerated plans, and provides graded scores with one parallel forward pass rather than sequential generation. For readability below we write  $s(X, Y)$  for  $s_A(X, Y)$  when the current target  $A$  is clear, and  $s_0 = s_A(X_0, Y_0)$  for its unperturbed score.

For each structured action on a component  $C$ , we define a *selection span* containing the tokens that identify  $C$  and a *decision span* containing the tokens that specify the action applied to  $C$ . Whole-output, selection, and decision scores differ only in the token set  $A$  used in eq. (2). The order of the selection and decision spans in the output introduces an autoregressive asymmetry: the later span is scored conditional on the earlier span because the earlier tokens remain in its output prefix. Which span is conditioned on the other therefore depends on the plan syntax. This conditioning is part of

the score and should not itself be interpreted as a causal relation between component selection and action specification. Separating the spans nevertheless allows their input dependencies to be examined individually.

### 5.2 Field-Level Attribution by In-Distribution Occlusion

The explanation unit is a *State* field, such as one measured or controlled property of a component, rather than a single prompt token. For field  $f$ , replacing its rendered value by an in-distribution substitute  $v$  produces  $X_{f \rightarrow v}$ . Its effect on the recorded plan is

$$d_f^v = s_0 - s(X_{f \rightarrow v}, Y_0). \quad (3)$$

We use the empirical 10th and 90th percentiles of the corresponding per-component field distribution as substitutes. If that distribution is degenerate, we fall back to the pooled empirical distribution of the same field type. The attribution is the mean absolute score change over only the substitutes that change the rendered prompt. When both substitutes displace the value,

$$a_f = \frac{1}{2} (|d_f^{q_{10}}| + |d_f^{q_{90}}|). \quad (4)$$

A substitute that renders identically after formatting is a *render-no-operation* and is excluded rather than counted as evidence of zero importance. If neither substitute changes the rendered value, attribution is unavailable for that field in that decision. Sampling both distribution tails reduces dependence on a single baseline, while empirical substitutes avoid deletions or arbitrary zeros that could measure out-of-distribution surprise instead of field dependence [20].

### 5.3 Faithfulness

The score  $a_f$  ranks fields by their measured single-field  $q_{10}/q_{90}$  effect on the posterior plan score by construction. We adapt ERASER's comprehensiveness and sufficiency protocol [1], an instance of the broader self-consistency principle [10], to test whether this ranking transfers to joint perturbations of structured *State* fields in a generated plan. The scored quantity is the posterior plan score in eq. (2) rather than a class probability, and the removal unit is a *State* field set to an in-distribution substitute rather than a deleted token. For a field count  $k$ , let  $X_{\text{remove top-}k}$  denote the prompt in which the  $k$  highest-attributed fields are replaced, and let  $X_{\text{keep top-}k}$  denote the prompt in which those fields retain their observed values while all other fields are replaced. We define

$$\begin{aligned} \text{comp}(k) &= s_0 - s(X_{\text{remove top-}k}, Y_0), \\ \text{suff}(k) &= s_0 - s(X_{\text{keep top-}k}, Y_0). \end{aligned} \quad (5)$$

A large comprehensiveness drop means the highest-attributed fields were needed; a small sufficiency drop means that retaining them largely preserves the output's likelihood. In an Area Over the Perturbation Curve (AOPC) test [16, 1], we sweep  $k$  over 10–50% of the fields in ten-percentage-point steps and compare the attribution ranking with equal-size random sets via the average over five cutoffs. The evaluation occlusion sets a field to its in-distribution median, differing from the quantile operator used to obtain the ranking and thereby reducing circularity. We call a ranking *AOPC-faithful* when top-ranked removal has greater comprehensiveness

and lower sufficiency than random removal under one-sided paired Wilcoxon signed-rank tests. We test the whole output and both spans separately.

#### 5.4 Plausibility

An AOPC-faithful ranking identifies which perturbations most affect the recorded plan’s posterior score; plausibility asks whether the resulting allocation agrees with the domain knowledge. Attribution across components forms the *component-level attribution distribution*. We use *component-level attribution locality* for the degree to which this distribution concentrates on the acted component rather than on other components, and quantify it using acted-component share and acted-component top rate. The expected component-level attribution locality can differ by span. Selecting one component may require comparison across the system, whereas specifying an action after that component is fixed may be expected to concentrate on its own state. We instantiate this expectation with domain knowledge and compare it with a reference controller where one is available.

For an action on component  $C$ , let  $F$  be the set of attributed *State* fields and  $F_C \subseteq F$  the fields belonging to  $C$ . The *acted-component attribution share*, or *acted-component share* for short, is the fraction of attribution mass on those fields,

$$\text{acted-component share} = \left( \sum_{f \in F_C} a_f \right) / \left( \sum_{f \in F} a_f \right), \quad (6)$$

and the *acted-component top rate* is the frequency with which  $C$  is the most-attributed component. The median-baseline ablation recomputes each attribution with a single empirical median substitute and re-derives the acted-component share. We additionally report total *State* dependence as the mean absolute score change when all fields are jointly set to their lower or upper quantile substitutes. Since sequence-score magnitudes vary by LLM and output, we use it only as a within-LLM diagnostic.

We interpret these component-level attribution locality measures only when the attribution ranking passes the AOPC test and the perturbations provide adequate coverage. A render-no-operation occurs when a substitute and the original value render identically after formatting, so the prompt does not change. If this happens frequently, statistical significance does not establish a stable allocation between a component’s own fields and other fields. We therefore report render-no-operation rates and require the component-level attribution locality results to remain qualitatively stable under the median-baseline ablation before drawing a plausibility conclusion.

We aggregate per decision rather than per action. The acted-component share is averaged within a decision and then across decisions, preventing a few large plans from dominating by action count. Cross-LLM comparisons use scale-free effect sizes and rates rather than raw log-probability magnitudes, which vary with output length and tokenizer.

## 6 Evaluation

We evaluate the explanations produced by  $M3$ ’s explainability method along the two dimensions defined in the approach. We first describe how DeltaIoT instantiates the method and how the recorded decisions are sampled and replayed. We then answer **RQ1** by testing explanation faithfulness and **RQ2** by examining

component-level attribution locality and its association with agreement with the domain rule.

### 6.1 Setup

We evaluate the method on DeltaIoT (Section 2) with Qwen3.5-9B, Ministral-3-8B-Instruct-2512, and gemma-3-12b-it. The checkpoints and logged embedded-planner runs were reported by Kara et al. [7]. We replay each decision only with the LLM that generated it, using the released checkpoint and precision, the exact logged prompt as  $X_0$ , and the verbatim recorded output through its last identified action, retaining text before and between actions as  $Y_0$  [7]. Qwen3.5-9B’s optional thinking mode was disabled during deployment, so  $Y_0$  contains no reasoning trace. The replay uses single-sequence scoring without padding under Torch 2.11.0 and Transformers 5.9.0.

DeltaIoT instantiates the generic method as follows. The components are its 17 directed links. The 40 perturbable fields comprise 17 SNR values, 17 transmission-power values, and six traffic-distribution values. Within the *State* block, we edit only these per-link fields; its global metrics remain fixed. The *Task description*, *Demonstration*, and *Memory* blocks also remain fixed. A link reference instantiates the selection span, and the associated action value instantiates the decision span. In the deployed output syntax, the link reference precedes the action value, so the decision-span score is conditioned on the recorded selection tokens. Mote-level equalize actions do not identify one link as their own component, so they remain in whole-output scoring but are excluded from the component-level attribution locality analysis. Quantile substitutes come from the empirical pool for each link and field, with a pooled same-field-type fallback when a link-specific pool is degenerate. The rule-based controller supplied with DeltaIoT labels whether each power action agrees with the domain rule on the same *State*.

We score  $n = 150$  adaptation decisions per LLM, stride-sampled across the logged repetitions and uncapped in plan size, with attribution by  $q_{10}/q_{90}$  occlusion on the deployed prompt and per-decision aggregation. The samples are drawn from 865/705/509 logged adapting decisions (Qwen/Ministral/Gemma). To account for dependence within runs, we calculate 95% confidence intervals (CIs) from 20,000 bootstrap samples that resample the ten trajectories with replacement and keep each trajectory’s decisions together; a fixed random seed makes the results reproducible. Since the target is the recorded output through its last identified action, the sample also includes actions the validator later rejected. The deployed planner had a 200-token output cap. We do not exclude sampled decisions containing truncations, duplicate restatements, or actions that the response itself rejects, because these artifacts are part of the deployed behavior. We account for them when interpreting the results.

We run all attribution scoring on a single workstation with one NVIDIA RTX PRO 6000 Blackwell GPU, and score each LLM at its released precision. The scoring took approximately 5.3/1.9/7.3 GPU-hours (Qwen/Ministral/Gemma).

Comprehensiveness and sufficiency are averaged over decisions and measured in nats, using summed token log-probabilities. Each AOPC value also averages the five  $k$  cutoffs defined in Section 5.3. Raw values are compared only within the same output target because the whole output and individual spans contain different numbers of tokens. Ministral’s span and locality rows use 149 decisions

**Table 1: AOPC and component-level attribution locality on DeltaIoT.**

|                                                   | Qwen           | Ministral      | Gemma           |
|---------------------------------------------------|----------------|----------------|-----------------|
| $n$ decisions/runs                                | 150/10         | 150/10         | 150/10          |
| comp top/rnd/bot                                  | 2.60/1.44/0.06 | 6.08/3.42/0.36 | 1.22/0.65/0.00  |
| suff top/rnd                                      | 1.05/2.38      | 1.96/4.99      | 0.73/1.40       |
| $p_{\text{comp}}/p_{\text{suff}}$ (runs)          | .001/.001      | .001/.001      | .003/.001       |
| AOPC gate                                         | pass           | pass           | not established |
| attribution-locality result                       | usable         | usable         | rejected        |
| <i>Selection span (which component to act on)</i> |                |                |                 |
| acted-comp. share                                 | 0.11           | 0.10           | 0.02            |
| acted-comp. top rate                              | 0.18           | 0.26           | 0.00            |
| <i>Decision span (what to do to it)</i>           |                |                |                 |
| acted-comp. share                                 | 0.34           | 0.33           | 0.40            |
| acted-comp. top rate                              | 0.78           | 0.82           | 0.50            |

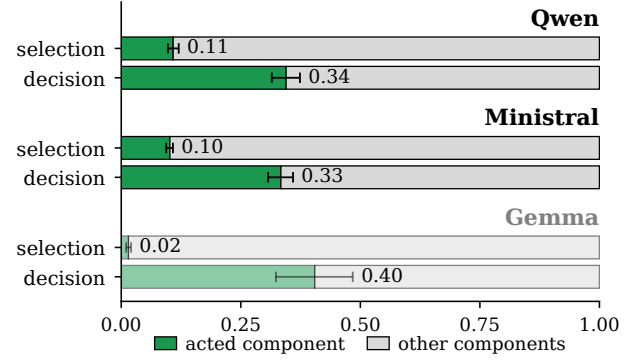
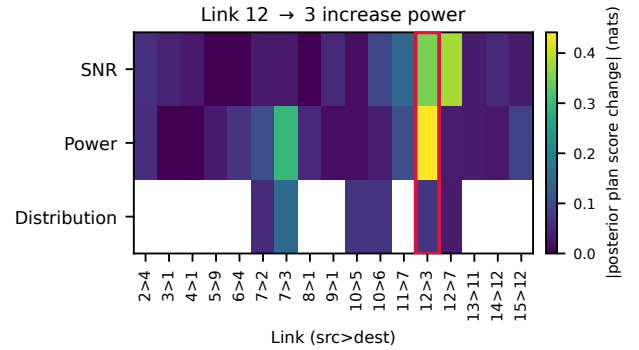
because one sampled decision contains only mote-level actions. The reported  $p$ -values come from exact one-sided Wilcoxon signed-rank tests on the ten trajectory means. Whole-output rank-biserial effect sizes are  $+1.00/+1.00/+0.93$  for comprehensiveness and  $-1.00$  for sufficiency. Removing bottom-ranked fields provides an additional baseline alongside random removal.

## 6.2 RQ1: Faithfulness

We interpret an LLM’s component-level attribution locality only when both comprehensiveness and sufficiency beat the random baseline with adequate perturbation coverage. Removing the top-attributed fields reduces the recorded plan’s posterior score more than random or bottom-ranked removal, while retaining only the top fields preserves it better (Table 1). The trajectory-level tests retain the whole-output AOPC pass for Qwen and Ministral. The per-span AOPC sweep confirms the result for selection and decision separately: decision-span comp(top)/comp(random) is 0.42/0.12 and 0.50/0.19 for Qwen and Ministral, while selection gives 0.49/0.27 and 0.64/0.38. All corresponding trajectory-level comprehensiveness and sufficiency tests pass ( $p \leq 0.003$ ). Together, these tests establish faithful rankings for Qwen and Ministral on the whole output and both spans.

Gemma’s logged *State* observations are highly clustered, with 53% of its other-component (and 41% of its acted-component)  $q_{10}/q_{90}$  perturbations rendering identically to the current value (vs  $\sim 5\%$  for the other two). Correspondingly, 27 of its 40 link-specific field pools are degenerate (vs 7), and most of its median replacements are render-no-operations as well, so its nominal AOPC pass does not establish a faithful ranking. We therefore report Gemma’s values descriptively but withhold both a faithfulness verdict and a component-level attribution locality interpretation.

**RQ1 (faithfulness).** Qwen and Ministral yield faithful whole-output and span-level attribution rankings, with adequate perturbation coverage and stable acted-component share results, while Gemma’s clustered recordings leave its faithfulness not established and its component-level attribution locality diagnostically unusable.

**Figure 3: Acted-component attribution share and trajectory-cluster 95% CIs; Gemma is diagnostic only.****Figure 4: Worked Qwen decision-span attribution. The acted link is outlined; its fields receive 0.27 of the attribution mass.**

## 6.3 RQ2: Plausibility

Having established faithful attribution rankings and diagnostic usability for Qwen and Ministral, we analyze component-level attribution locality (Fig. 3). The two spans describe the two parts of an action. Component selection depends on system-wide *State* information. On the selection span, 0.108 [0.098, 0.120] is the acted-component share for Qwen and 0.102 [0.094, 0.108] for Ministral; the brackets give the 95% CIs. For reference, uniform attribution gives sample-specific acted-component shares of 0.064/0.061 for Qwen/Ministral. The acted-component top rates are only 0.18/0.26. Deciding *what* to do to that component is more strongly grounded in the acted component. On the decision span, acted-component share rises to 0.345 [0.315, 0.374] for Qwen and 0.334 [0.307, 0.359] for Ministral, and the acted-component top rate rises to 0.78/0.82. The decision-span result remains stable under the median-baseline ablation (0.35–0.42; actions whose acted-component fields are all render-no-operations are excluded) and a conservative treatment of render-no-operation fields (upper-bound share  $\leq 0.36$ ). For this bound, excluded acted-component fields receive the largest measured attribution and all other excluded fields receive zero.

Fig. 4 illustrates one retained Qwen action. For link  $12 \rightarrow 3$ , the SNR is  $-4.0$  and power is 12; increasing power agrees with the domain rule. The link’s own power is the highest-attributed field and its own SNR is third, while fields of other links account for the remaining mass. This is the action-level evidence summarized by acted-component share.

Finally, we ask whether component-level attribution locality is related to the domain rule. The rule-based controller is used to label whether each power action agrees with the rule on the same *State*. For both groups, we record the LLM’s decision-span acted-component share. Rule-matching actions have a higher acted-component share: 0.374 vs 0.312 for Qwen (difference 0.062, 95% CI [0.034, 0.089]) and 0.373 vs 0.235 for Ministral (difference 0.139, [0.098, 0.177]). This domain-coherence check shows that actions agreeing with the rule are more strongly grounded in the acted link’s own *State*.

**RQ2 (plausibility).** For Qwen and Ministral, component selection depends on system-wide *State* information, whereas action specification is more strongly grounded in the acted component. Power actions agreeing with the domain rule have a higher acted-component share than disagreements.

## 7 Threats to Validity

**Construct Validity.** Our analysis uses acted-component share to characterize how attribution is allocated between the acted component and the rest of the *State*. The posterior score of an action bundles *which* component to act on with *what* to do on it, so a low acted-component share could reflect correct global prioritization rather than misattention. We mitigate this with the selection/decision span split. The method explains the recorded action and does not assume that a uniquely correct action exists. The controller comparison tests agreement with one domain rule and serves as a plausibility reference rather than ground truth. A fixed output structure with one action per component in a fixed order could further isolate component selection from action specification.

**Internal Validity.** Each LLM adapted its own run, so trajectories, plan memories, and encountered states differ. We replay the exact prompt, output, and checkpoint and base the tests on within-LLM comparisons. The four hypotheses are non-exclusive: *M3* measures how sensitivity is distributed within the current *State*, but does not rule out simultaneous *Memory* dependence. Because *Memory* contains both previous *State* snapshots and prior plans, removing it completely would also remove legitimate temporal information and would not isolate copying. The adaptation rules in the *Task description* stay fixed under every perturbation, so *M3* measures *State* dependence given those rules and does not test whether the planner follows them. While that question belongs to module *M4*, *M3* provides some insight: a planner that does not depend on the fields named in the rules could not apply those rules, whereas apparent agreement with a rule can arise without the rule text explicitly being considered.

**Conclusion Validity.** When an LLM’s *State* observations are highly clustered, most perturbations are no-operations and the metric is computed over a reduced, non-uniform field set. We harden against

this with the median-baseline ablation, the reported render-no-operation rate, and an upper-bound acted-component share, and we exclude Gemma’s component-level attribution locality from the headline.

**External Validity.** Attribution is evaluated on DeltaIoT, three relatively small LLMs, one embedded architecture, and one attribution family. The component-level attribution locality analysis requires identifiable components, *State* fields assignable to them, and actions that can be mapped to the affected components. Systems without this decomposition require a different explanation unit. The component-level attribution locality measures and controller comparison assess technical plausibility, not whether stakeholders understand or can use the explanations; stakeholder validation remains future work. A second exemplar (e.g. MalwareInfection, whose categorical/graph state needs a different field treatment), larger LLMs, and a high-sample LIME/SHAP cross-check would establish generality.

## 8 Conclusion

We outlined a vision for an extensible platform for explaining what an LLM component’s adaptation decisions depend on: each module operationalizes one hypothesis and assesses its explanation for faithfulness and plausibility. Together, these assessments provide technical evidence for evaluating the planner’s trustworthiness. We implemented one of four identified modules based on posterior plan scoring for structured plans to investigate how an LLM’s dependence on the managed system’s *State* is distributed across system components. The method groups *State* fields by managed-system component, separates the token spans that select a component from those that specify its adaptation, and measures how attribution is divided between the acted component and the rest of the system. It establishes attribution faithfulness before comparing component-level attribution locality with domain knowledge as a plausibility evaluation. On the DeltaIoT exemplar, Qwen and Ministral pass the faithfulness test and provide sufficient perturbation coverage and stability for locality interpretation, whereas Gemma’s faithfulness is not established due to insufficient perturbation coverage. Component selection depends on system-wide *State* information, action specification is more strongly grounded in the acted-on component, and power actions agreeing with the domain rule have a higher acted-component share than disagreements.

Future work will complete the platform through the other three envisioned modules and a common software interface, investigate how findings from different modules interact, and evaluate a second exemplar, larger LLMs, and alternative attribution estimators such as LIME and SHAP.

## Data Availability

The replication package of our experiments is available at <https://doi.org/10.5281/zenodo.21475861>.

## Acknowledgments

This work was supported by funding from the pilot program Core Informatics at KIT (KiKIT) of the Helmholtz Association (HGF) and KASTEL Security Research Labs, Karlsruhe.

Some parts of our work were assisted by generative AI. We used Anthropic's Claude (Opus 4.8 and Fable 5) and OpenAI's GPT-5-based Codex to help us build the evaluation setup and refine the manuscript.

## References

- [1] Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. 2020. ERASER: a benchmark to evaluate rationalized NLP models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, Online, 4443–4458. doi:10.18653/v1/2020.acl-main.408.
- [2] Raghav Donakanti, Prakhar Jain, Shubham Kulkarni, and Karthik Vaidhyanathan. 2024. Reimagining self-adaptation in the age of large language models. In *2024 IEEE 21st International Conference on Software Architecture Companion (ICSA-C)*. (June 2024), 171–174. doi:10.1109/ICSA-C63560.2024.00036.
- [3] Felix Feit, Andreas Metzger, and Klaus Pohl. 2022. Explaining online reinforcement learning decisions of self-adaptive systems. In *2022 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*, 51–60. doi:10.1109/ACSOS55765.2022.00023.
- [4] Samuel Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. 2018. Visualizing and understanding Atari agents. In *Proceedings of the 35th International Conference on Machine Learning*. PMLR, (July 2018), 1792–1801. Retrieved June 1, 2026 from <https://proceedings.mlr.press/v80/greydanus18a.html>.
- [5] Muhammad Usman Iftikhar, Gowri Sankar Ramachandran, Pablo Bollansée, Danny Weyns, and Danny Hughes. 2017. DeltaIoT: a self-adaptive internet of things exemplar. In *2017 IEEE/ACM SEAMS*. (May 2017), 76–82. doi:10.1109/SEAMS.2017.21.
- [6] Alon Jacovi and Yoav Goldberg. 2020. Towards faithfully interpretable NLP systems: how should we define and evaluate faithfulness? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, (Eds.) Association for Computational Linguistics, Online, (July 2020), 4198–4205. doi:10.18653/v1/2020.acl-main.386.
- [7] Kerem Kara, Balint Mate, Vincenzo Scotti, Diego Perez-Palacin, and Raffaella Mirandola. 2026. Large Language Models for Self-Adaptive Systems: Feedback-Loop Integration and Agentic Adaptation. Tech. rep. 46.23.01; LK 01. Karlsruher Institut für Technologie (KIT). 10 pp. doi:10.5445/IR/1000194592.
- [8] Jialong Li, Mingyue Zhang, Nianyu Li, Danny Weyns, Zhi Jin, and Kenji Tei. 2024. Generative AI for self-adaptive systems: state of the art and research roadmap. *ACM Transactions on Autonomous and Adaptive Systems*, 19, 3, (Sept. 2024), 1–60. doi:10.1145/3686803.
- [9] Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. Curran Associates Inc., Long Beach, California, USA, 4768–4777. ISBN: 9781510860964.
- [10] Andreas Madsen, Sarath Chandar, and Siva Reddy. 2024. Are self-explanations from large language models faithful? In *Findings of the Association for Computational Linguistics: ACL 2024*. Lun-Wei Ku, Andre Martins, and Vivek Srikumar, (Eds.) Association for Computational Linguistics, Bangkok, Thailand, (Aug. 2024), 295–337. doi:10.18653/v1/2024.findings-acl.19.
- [11] Bálint Máté, Diego Perez-Palacin, Vincenzo Scotti, and Raffaella Mirandola. 2026. A conceptual model of operationalizable trustworthiness for autonomous systems. In *2026 IEEE 50th Annual Computers, Software, and Applications Conference (COMPSAC)*, 2148–2154. doi:10.1109/COMPSAC69091.2026.00317.
- [12] Andreas Metzger, Jan Laufer, Felix Feit, and Klaus Pohl. 2024. A user study on explainable online reinforcement learning for adaptive systems. *ACM Trans. Auton. Adapt. Syst.*, 19, 3, Article 15, (Sept. 2024), 44 pages. doi:10.1145/3666005.
- [13] Francesco Renato Negri, Niccolò Nicolosi, Matteo Camilli, and Raffaella Mirandola. 2026. Efficient self-adaptation through explanation-driven white-box optimization. *ACM Trans. Auton. Adapt. Syst.*, 21, 3, Article 24, (June 2026), 31 pages. doi:10.1145/3737648.
- [14] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. “Why should I trust you?”: explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. Association for Computing Machinery, San Francisco, California, USA, 1135–1144. ISBN: 9781450342322. doi:10.1145/2939672.2939778.
- [15] Benjamin Roth, Nicholas Edwards, Pingjun Hong, Loris Schoenegger, and Sebastian Schuster. 2026. From Models to Systems: A Survey of Explainability for Tool-Augmented Language Models and AI Agents. Discussion Paper. University of Vienna, (Jan. 2026). doi:10.5281/zenodo.18373874.
- [16] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. 2017. Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems*, 28, 11, 2660–2673. doi:10.1109/TNNLS.2016.2599820.
- [17] Gabriele Sarti, Nils Feldhus, Ludwig Sickert, and Oskar van der Wal. 2023. Inseq: an interpretability toolkit for sequence generation models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*. Danushka Bollegala, Ruihong Huang, and Alan Ritter, (Eds.) Association for Computational Linguistics, Toronto, Canada, (July 2023), 421–435. doi:10.18653/v1/2023.acl-demo.40.
- [18] Yael Septon, Tobias Huber, Elisabeth André, and Ofra Amir. 2023. Integrating policy summaries with reward decomposition for explaining reinforcement learning agents. In *Advances in Practical Applications of Agents, Multi-Agent Systems, and Cognitive Mimetics. The PAAMS Collection: 21st International Conference, PAAMS 2023, Guimarães, Portugal, July 12–14, 2023, Proceedings*. Springer-Verlag, Guimarães, Portugal, 320–332. ISBN: 978-3-031-37615-3. doi:10.1007/978-3-031-37616-0\_27.
- [19] Raphael Straub, Florian Sihler, Ali Torbati, Cong Wang, Raffaella Groner, Verena Klös, and Matthias Tichy. 2026. Explainability in self-adaptive systems: a systematic literature review. In *Software Engineering and Advanced Applications*. Vol. 16082. Davide Taibi and Darja Smite, (Eds.) Springer Nature Switzerland, Cham, 280–297. ISBN: 978-3-032-04200-2. doi:10.1007/978-3-032-04200-2\_19.
- [20] Pascal Sturmfels, Scott Lundberg, and Su-In Lee. 2020. Visualizing the impact of feature attribution baselines. *Distill*. <https://distill.pub/2020/attribution-baselines>. doi:10.23915/distill.00022.
- [21] Grisel Vázquez, Alexandros Evangelidis, Sepeedeh Shahbeigi, Radu Calinescu, and Simos Gerasimou. 2026. Mind the prompt: self-adaptive generation of task plan explanations via LLMs. In *Proceedings of the 21st International Conference on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '26)*. Association for Computing Machinery, New York, NY, USA, (Apr. 2026), 162–175. ISBN: 9798400724459. doi:10.1145/3788550.3794879.
- [22] Kayo Yin and Graham Neubig. 2022. Interpreting language models with contrastive explanations. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, (Eds.) Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, (Dec. 2022), 184–198. doi:10.18653/v1/2022.emnlp-main.14.

Received 2026-07-23; accepted 2026-08-23