Towards an Agentic MAPE-K Architecture for Self-Adaptive Robotic Arm Systems

Joaquim Ribeiro
Universidade Estadual do Ceará
Fortaleza, Brazil
joaquim.emanuel@aluno.uece.br

Beatriz Andrade
Universidade Estadual do Ceará
Fortaleza, Brazil
bea.andrade@aluno.uece.br

Suyane Freitas
Universidade Estadual do Ceará
Fortaleza, Brazil
suyane.freitas@aluno.uece.br

Lucas Alves
Universidade Estadual do Ceará
Fortaleza, Brazil
lucas.vieira@aluno.uece.br

Paulo Henrique Maia
Universidade Estadual do Ceará
Fortaleza, Brazil
pauloh.maia@uece.br

Abstract

Robotic arm systems operate in dynamic and uncertain environments that require runtime adaptation. Although MAPE-K is widely used to structure self-adaptive systems, rule-based implementations often rely on predefined adaptation knowledge, limiting their ability to handle situations outside the anticipated adaptation space. This paper proposes an Agentic MAPE-K architecture for self-adaptive robotic arm systems, augmenting the Analyser and Planner with intelligent agents to support diagnosis and adaptation planning when predefined knowledge is insufficient. The architecture was instantiated as a proof of concept in a simulated robotic arm environment and examined using a single execution trace that encountered an obstacle whose height exceeded the anticipated range. After an initial predefined plan failed, the resulting runtime evidence supported an agent-assisted diagnosis and the reconsideration of available adaptation plans. Because an acceptable predefined alternative remained available, the complete fallback branch for generating a new plan was not exercised. The results therefore provide preliminary evidence of the feasibility of the proposed architectural integration and of using adaptation outcomes to inform subsequent MAPE-K decisions, rather than validation of end-to-end or agent-generated adaptation.

Keywords

Self-Adaptive Systems, MAPE-K, Agentic AI, Robotic Arm Systems, Unanticipated Situations.

1 Introduction

Robotic arm systems are increasingly used in industrial manufacturing, research, and defense, where they must operate under dynamic and partially uncertain conditions [1]. Workspace changes, such as unexpected obstacles, variations in object or goal positions, and execution failures, can invalidate design-time assumptions [2]. Reliable operation in these environments requires self-adaptive robotic arm systems to adapt their behaviour at runtime with minimal human intervention [3].

Self-Adaptive Systems (SAS) offer principles and mechanisms to manage environmental and operational changes at runtime [4]. The MAPE-K feedback loop is a widely used architectural model that organizes adaptation logic into Monitor, Analyze, Plan, and Execute functions, all supported by shared Knowledge [5]. In many rule-based MAPE-K implementations, diagnoses, adaptation rules, and plans are primarily defined at design time based on anticipated situations [6].

Although this organization supports runtime adaptation, its effectiveness declines when the system faces conditions not represented in the available adaptation knowledge. Unexpected obstacles, new failure conditions, or changes in task constraints can move the system beyond its anticipated adaptation space. In these cases, predefined diagnostic rules and adaptation plans may not provide an adequate response [6, 7].

Recent advances in Agentic AI offer opportunities to enhance predefined adaptation knowledge with context-sensitive reasoning [8, 9]. Large Language Model (LLM)-based agents can interpret runtime evidence, reason about goals and constraints, and leverage historical knowledge to support diagnosis and adaptation planning when existing rules are insufficient. Integrating these capabilities into a feedback loop requires maintaining the separation of concerns in MAPE-K and clearly defining the roles of agentic and deterministic mechanisms. However, such integrations of agentic reasoning into the MAPE-K feedback loop remains largely unexplored, particularly in the context of robotic manipulation systems.

In this context, we propose an **Agentic MAPE-K Architecture for Self-Adaptive Robotic Arm Systems**. This architecture augments the Analyser and Planner components with intelligent agents. The Analyser uses monitored runtime data and historical knowledge to diagnose the current situation. The Planner considers the diagnosis, adaptation goals, task constraints, and available plans to determine an adaptation response. Our approach maintains the modular structure of the MAPE-K loop and its shared Knowledge, combining predefined adaptation mechanisms with agent-supported reasoning for situations not explicitly addressed by existing adaptation knowledge.

The main contributions of this paper are: (i) an Agentic MAPE-K architecture that integrates agent-based reasoning into the Analyser and Planner of a self-adaptive robotic manipulator; (ii) an architectural allocation of agentic reasoning responsibilities to the Analyser and Planner, while preserving the modular structure of MAPE-K; and (iii) a proof-of-concept implementation in a simulated robotic arm system. This proof of concept provides preliminary evidence of the feasibility of integrating agent-supported reasoning into the MAPE-K feedback loop through a single execution trace in a simulated robotic arm environment.

2 Background and Related Work

Self-Adaptive Systems (SAS) adapt their behaviour at runtime in response to changes in the system or its environment [10]. They are commonly organized into a *managed system*, responsible for domain functionality, and a *managing system*, which performs adaptation through a feedback loop [11]. A widely adopted realization of this loop is MAPE-K, composed of Monitor, Analyser, Planner, and Executor functions supported by shared Knowledge [5]. The Monitor collects runtime information, the Analyser evaluates the system state and identifies adaptation needs, the Planner selects or defines an adaptation plan, and the Executor applies it to the managed system.

A key challenge for SAS is uncertainty, which may arise from environmental changes, sensor information, changing goals, or incomplete knowledge [12]. In this work, we distinguish between anticipated uncertainties, which can be addressed using adaptation knowledge defined at design time, and unanticipated situations, which are not adequately covered by the available adaptation knowledge. This distinction is particularly relevant to rule-based MAPE-K implementations, whose diagnosis and planning mechanisms commonly depend on predefined conditions and adaptation strategies.

Related work relevant to this study spans two complementary directions: MAPE-K-based self-adaptive systems and recent approaches that incorporate LLM-based or multi-agent reasoning into adaptive systems.

Iglesia and Weyns [11] proposed reusable formal templates for MAPE-K using Timed Automata and TCTL, supporting rigorous specification and verification of adaptation behaviour. Although their approach improves the reuse of MAPE-K models, analysis and planning remain based on behaviour and adaptation knowledge defined in advance. Our work preserves the MAPE-K structure while augmenting the Analyser and Planner with agentic reasoning mechanisms to support situations where predefined adaptation knowledge is insufficient.

Alves et al. [13] proposed MAPER, which extends MAPE-K with an LLM-based Reasoner for handling unanticipated situations during analysis and planning. MAPER employs iterative reasoning and validation mechanisms, including LLM-as-a-Judge, to generate and assess adaptation strategies. In contrast, our approach adopts an Agentic AI perspective by incorporating specialized reasoning responsibilities directly into the Analyser and Planner. Moreover, while MAPER was evaluated in the SWIM software-system domain, our work investigates this integration in robotic manipulation, which introduces distinct operational and safety constraints.

Pandey et al. [14] introduced POLARIS, a multi-agent architecture that distributes the responsibilities for reasoning, learning, and decision-making among specialized agents. While POLARIS demonstrates the potential of multi-agent reasoning for SAS, it is not explicitly organized around the MAPE-K feedback loop. Our approach instead preserves MAPE-K as the architectural foundation while integrating agentic capabilities into its Analyser and Planner components.

3 Architecture

To enhance the runtime adaptation capabilities of self-adaptive robotic arm systems, we propose an Agentic MAPE-K architecture. The robotic arm and its operating environment constitute the managed system, while the proposed feedback loop acts as the managing system. The architecture preserves the modular organization and responsibilities of MAPE-K while augmenting the Analyser and Planner with agentic reasoning mechanisms. These mechanisms are activated when the predefined adaptation knowledge is insufficient to address the observed runtime situation.

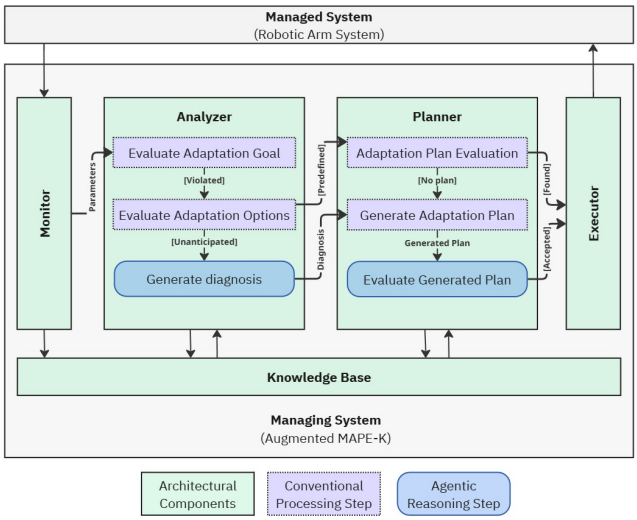

Figure 1: Architecture of the Agentic MAPE-K approach.

As illustrated in Figure 1, the architecture comprises the standard MAPE-K components: *Monitor*, *Analyser*, *Planner*, *Executor*, and shared *Knowledge*. Its main extension lies in the internal organization of the Analyser and Planner. Both components preserve their conventional MAPE-K responsibilities while incorporating agentic steps that are triggered when predefined adaptation options or plans cannot adequately address the current situation. The responsibilities and internal behaviour of each component are described below.

Knowledge. The Knowledge component stores and provides the information shared across the MAPE-K loop. This includes current and historical runtime data, adaptation goals and their associated metrics, task and system constraints, anticipated situations, predefined adaptation options and plans, and the outcomes of previous adaptations. It also records diagnoses, candidate options, and plans generated by the agentic components, as well as relevant agent configurations. This shared knowledge supports both the conventional and agentic paths of the Analyser and Planner.

Monitor. The Monitor collects sensor data, task information, execution states, and other runtime parameters from the robotic arm system and its environment. These observations include the position and velocity of the robotic arm, the state and position of the manipulated object, target coordinates, obstacle information, task progress, execution time, and other relevant properties.

The collected data are organized into a representation of the current runtime state and stored in the Knowledge component. This information provides the Analyser with the parameters required to evaluate the adaptation goals. Monitored data are also retained as historical records, allowing subsequent MAPE-K cycles to consider previous states, events, and adaptation outcomes.

Analyser. The Analyser extends the MAPE-K analysis function with an agentic reasoning mechanism that is activated when predefined adaptation knowledge is insufficient. Its main responsibility is to evaluate the current system state against the adaptation goals and determine whether available adaptation options can address an identified goal violation.

The analysis process begins with the *Evaluate Adaptation Goal* step. Based on the parameters provided by the Monitor, the Analyser determines whether the current state of the robotic arm, its task, and the surrounding environment satisfies the adaptation goals stored in the Knowledge component. These goals are operationalized through quality metrics, conditions, or thresholds related to task progress, target achievement, execution time, obstacle avoidance, and other relevant system properties. In the reactive adaptation strategy considered in this work, if no adaptation goal is violated, no adaptation is required, and the feedback loop returns to the Monitor.

If an adaptation goal is violated, the Analyser proceeds to the *Evaluate Adaptation Options* step. It characterizes the current situation and attempts to match the observed conditions to anticipated situations or diagnoses represented in the Knowledge component. The Analyser then retrieves the associated predefined adaptation options and evaluates them according to their applicability to the current situation, their expected contribution to restoring the violated goal, and their compatibility with the task constraints.

When one or more predefined adaptation options are applicable, the Analyser forwards the identified situation, the violated adaptation goal, the supporting runtime evidence, and the applicable options to the Planner. In this case, the system follows the conventional analysis path without invoking agentic reasoning.

When no predefined adaptation option can adequately address the observed situation, the Analyser identifies a gap in the available adaptation knowledge. In the proposed architecture, this condition is treated as an unanticipated situation with respect to the available adaptation knowledge, thereby activating the agentic extension of the Analyser. During the *Generate Diagnosis* step, the Analyser Agent reasons over the monitored parameters, the violated adaptation goal, the task constraints, the available system capabilities, and relevant historical knowledge.

The agent produces a structured diagnosis explaining why the adaptation goal is not being satisfied. The diagnosis may include the identified issue type, the affected adaptation goal, an anomaly indicator, urgency, a confidence indicator, supporting evidence, and other relevant contextual information. Based on this diagnosis, the Analyser Agent provides a context-specific characterization of the detected condition and may propose a new candidate adaptation option when the existing options are insufficient. The resulting diagnosis and supporting evidence are forwarded to the Planner, which first determines whether existing adaptation knowledge can be reconsidered in light of the refined diagnosis before generating a new plan.

The Analyser therefore forwards to the Planner the detected goal violation, the supporting evidence, and either applicable predefined adaptation options or an agent-generated diagnosis and candidate adaptation option.

Planner. The Planner receives the violated adaptation goal, the supporting runtime evidence, and either applicable predefined adaptation options or a new candidate option generated by the Analyser Agent. Its responsibility is to select an adaptation option and determine how that option can be realized through an executable adaptation plan. Similar to the Analyser, the Planner follows a conventional planning path when the predefined adaptation knowledge is sufficient and an agentic path when no suitable plan is available.

When the Analyser provides one or more applicable adaptation options, the Planner selects an option based on the analysis results, the violated adaptation goal, the task constraints, and the policies stored in the Knowledge component. It then searches the Knowledge component for predefined adaptation plans associated with the selected option. Each plan specifies the actions, parameters, and execution sequence required to realize the adaptation option in the robotic arm system.

The conventional planning path proceeds with the *Adaptation Plan Evaluation* step. In this step, the Planner assesses whether the retrieved plans are applicable to the current situation, compatible with the task constraints, and supported by the available capabilities of the robotic arm. If a predefined plan satisfies these criteria, it is selected and forwarded to the Executor without invoking agentic reasoning.

The agentic planning path is activated when no predefined adaptation plan is associated with the selected option or when none of the retrieved plans is considered suitable during the *Adaptation Plan Evaluation* step. This commonly occurs when the Analyser Agent generates a new candidate option for an unanticipated situation.

During the *Generate Adaptation Plan* step, the Planner Agent reasons over the diagnosis, the selected adaptation option, the violated adaptation goal, the task constraints, the available system capabilities, and relevant historical adaptation outcomes. Based on this information, the agent creates a structured adaptation plan describing how the selected option can be realized by the robotic arm system.

The generated plan then proceeds to the *Evaluate Generated Plan* step. In this step, the Planner Agent assesses the plan's expected execution by estimating its feasibility, likelihood of success, execution cost, compatibility with the task constraints and system capabilities, and ability to restore the violated adaptation goal. If the plan satisfies the defined acceptance criteria, it is forwarded to the Executor.

Executor. The Executor receives the adaptation plan selected or generated by the Planner and applies it to the managed robotic arm system. The plan specifies the actions, parameters, and execution sequence required to realize the selected adaptation option. The Executor translates these elements into commands supported by the robotic arm, such as activating a control strategy, changing the task configuration, modifying movement parameters, or executing an available robotic action.

During execution, the Executor records the status and outcome of the applied actions in the Knowledge component. This information includes whether the plan was successfully executed, any

errors that occurred, and the resulting state of the robotic arm and its environment. In the subsequent MAPE-K cycle, the Monitor observes the resulting runtime state, and the Analyser reevaluates whether the adaptation goal has been restored or whether further adaptation is required.

This organization preserves the modular structure of MAPE-K while restricting agentic reasoning to situations in which the predefined adaptation options or plans are insufficient.

4 Evaluation

This section presents a proof-of-concept evaluation of the Agentic MAPE-K architecture using a simulated self-adaptive robotic arm system. The evaluation focuses on a single execution trace involving an obstacle whose height exceeds the anticipated range of the initially applicable adaptation knowledge. It examines the interactions among the Monitor, Analyser, Planner, and Executor across successive adaptation cycles, with particular attention to how runtime evidence and previous adaptation outcomes support diagnosis refinement and the reconsideration of available adaptation plans.

Experimental Setup. The architecture was instantiated using the local LLM `qwen2.5:7b`, a Qwen 2.5 model with 7 billion parameters served via Ollama. The model was set to temperature 0 to minimize output variability and enhance consistency in diagnosis and plan generation. This configuration ensures reproducibility of the agentic reasoning process during the proof-of-concept evaluation.

Case Study and Unanticipated Situation. To evaluate the Agentic MAPE-K architecture, we used logs generated by an autonomous robotic arm simulator. The simulator represents a self-adaptive robotic arm responsible for moving an object from its initial position to a target in the environment [15]. To introduce a condition requiring adaptation, an obstacle was placed along the trajectory between the object and the target.

The predefined adaptation knowledge covers two anticipated operating conditions. When no obstacle is present, the robotic arm uses a predefined push/slide strategy. When an obstacle falls within the anticipated dimensions, it switches to a predefined pick-and-place strategy, using a standard lifting height that safely clears the tallest anticipated obstacle.

The evaluation addresses a scenario in which the obstacle dimensions are detected at runtime, but its height exceeds the anticipated range, as shown in Figure 2. More specifically, the observed obstacle is higher than the maximum height supported by the standard lifting configuration of the predefined pick-and-place plan. Consequently, this plan can no longer guarantee sufficient clearance. Although the obstacle height exceeds the range handled by the standard pick-and-place configuration, the Knowledge component still contains alternative predefined adaptation knowledge that can be reconsidered after the initial adaptation attempt. Therefore, the scenario challenges the applicability of the initially selected strategy without necessarily exhausting all predefined adaptation alternatives.

Although the obstacle is observable, the available adaptation knowledge does not include a suitable strategy or parameter configuration to handle its dimensions.

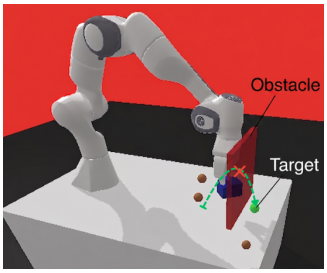

Figure 2: Obstacle exceeding the anticipated height.

This scenario represents a condition that exceeds the anticipated range of the initially applicable adaptation knowledge. The execution therefore allows us to examine how additional runtime evidence, including the unsuccessful outcome of a previous adaptation attempt, can trigger agentic diagnosis and support the subsequent reconsideration of available adaptation plans. As reported in Section 5, the complete agentic fallback for generating a new adaptation plan is not exercised because an acceptable predefined alternative remains available.

In this scenario, the Monitor collects the runtime evidence required by the Analyser to assess task progress, identify the obstacle, and determine whether the available adaptation knowledge is applicable to the observed situation. Table 1 summarizes the evidence used during diagnosis, planning, execution, and subsequent goal reevaluation.

Table 1: Monitored parameters used by the Analyser and Planner.

Monitored parameters	Definitions
Current task	Identifies the manipulation operation being performed.
End-effector position	Robotic arm position during execution.
Gripper state	Indicates whether the object is being grasped or released.
Object position	Represents the current position of the manipulated object.
Target position	Defines the destination of the manipulated object.
Object–target distance	Supports task and goal evaluation.
Obstacle presence	Indicates whether the path is blocked.
Obstacle dimensions	Assesses safe obstacle handling.
Active strategy or script	Identifies the adaptation tactic currently being executed.
Task success	Indicates whether the object reached the target.

Based on the monitored runtime evidence, the agentic steps exchange diagnosis and planning information through structured outputs. Table 2 summarizes the output schemas adopted by the Analyser Agent and Planner Agent in the proof-of-concept implementation. When the agentic analysis path is activated, the Analyser Agent produces a structured diagnosis containing the identified problem, anomaly indicator, urgency, confidence, and contextual information about the current task and system state. This diagnosis and its supporting evidence are then made available to the Planner Agent, whose output represents the selected or generated adaptation strategy, the corresponding actions, the expected outcome, and the reasoning supporting the adaptation decision.

These schemas describe the interfaces adopted by the agentic components and should not be interpreted as results from a specific execution. The concrete values observed during the evaluated scenario are presented in Section 5.

Table 2: Output schemas of the Analyser and Planner Agents.

Field	Description
Analyser Agent	
`anomalies`	Indicates whether an anomaly was detected.
`issue_type`	Type of problem identified by the agent.
`urgency`	Severity level assigned to the detected issue.
`confidence`	Confidence score associated with the diagnosis.
`diagnosis`	Textual description of the system state and detected issue.
`details`	Additional contextual information about the analysis.
`current_task`	Current task being executed by the robotic arm.
`dist_cube_target`	Distance between the object and the target position.
Planner Agent	
`selected_strategy`	Strategy selected or generated to address the current situation.
`actions`	Adaptation plan structured as a sequence of actions for the Executor.
`reasoning`	Explanation supporting the adaptation decision based on runtime evidence, simulation results, and historical context.
`expected_outcome`	Expected result after executing the adaptation plan.
`is_custom_tactic`	Indicates whether the selected tactic was generated dynamically (`true`) or corresponds to a predefined tactic (`false`).

5 Results and Discussion

This section presents the execution trace of the proposed Agentic MAPE-K architecture in a scenario involving an unanticipated high obstacle. The results focus on how runtime evidence and previous adaptation outcomes are incorporated across successive MAPE-K cycles to refine the diagnosis and reconsider available adaptation tactics.

During the experiment, the Monitor collected and structured information about the robotic arm, the manipulated object, the target, and the obstacle. This evidence was processed in accordance with the previously described architectural flow. The Analyser evaluated the current adaptation goal and the applicability of available adaptation options. At the same time, the Planner assessed the corresponding tactics based on their predicted success, estimated cost, and prior execution outcomes.

In the first MAPE-K cycle, the Analyser identified the situation as *path blocked*, indicating that an obstacle prevented progress toward the target. However, the presence of an obstacle is an anticipated uncertainty for which an adaptation strategy is already available in the Knowledge component. Therefore, the anomaly indicator remained set to `false`, and the system selected the predefined adaptation option *switch to pick-and-place*. This analysis was then passed to the Planner, which retrieved the corresponding plan and forwarded it to the Executor.

In the following cycle, the Monitor reported that the blockage persisted after the pick-and-place plan was executed. This unsuccessful outcome was stored as additional runtime evidence and used by the Analyser to reassess the situation. At this stage, the agentic reasoning mechanism was activated to reason over the available runtime evidence and provide a more context-sensitive diagnosis. Based on the monitored parameters and the outcome of the previous adaptation attempt, the Analyser identified that the obstacle height exceeded the range covered by the available adaptation knowledge. The condition was therefore characterized as unanticipated given existing knowledge, and the anomaly indicator was set to `true`.

The resulting diagnosis and its supporting runtime evidence were then forwarded to the Planner. Following the proposed architecture, the Planner first attempted to reuse adaptation plans already available in the Knowledge component before generating a new plan. Two predefined tactics were evaluated: *switch to pick-and-place*, with a predicted success of 0.57, and *activate left-right script*, with a predicted success of 0.91. The lower predicted success associated with the pick-and-place tactic was consistent with the available runtime evidence indicating that this tactic had not resolved the blockage in the previous cycle. The Planner therefore selected the *left-right script*, which presented the highest predicted success among the available alternatives.

According to the proposed planning flow, a predefined plan is considered acceptable when its evaluation satisfies the minimum acceptance threshold of 0.5. If none of the available predefined plans had satisfied this criterion, the Planner would have activated the agentic fallback path to generate and subsequently evaluate a new adaptation plan. However, this fallback branch was not exercised in the reported execution because the *left-right script* remained an acceptable predefined alternative.

After executing the left-right script, the blockage indicator changed to `false`. In the subsequent cycle, the Analyser no longer identified a blocked path. This result indicates that the condition that triggered the previous adaptation had been removed. However, the overall task had not yet been completed because the object had not reached the target. Therefore, the feedback loop continued to monitor the managed system and evaluate the adaptation goals based on the updated runtime state.

Table 3: Observed adaptation trace across successive cycles.

Cycle	Analyser output	Planner decision	Observed outcome
1	*Path blocked*; anomaly = `false`	Predefined *switch to pick-and-place* plan selected	The blockage persisted after execution.
2	*Path blocked*; obstacle height identified as outside the anticipated range; anomaly = `true`	Pick-and-place (0.57) and left-right script (0.91) evaluated; left-right script selected	The blockage indicator changed to `false`.

Table 3 summarizes the adaptation decisions and outcomes observed during the two adaptation cycles. The subsequent monitoring state, in which the blockage had been removed but the overall task remained incomplete, is described above.

The observed execution provides preliminary evidence that adaptation outcomes can be incorporated into subsequent MAPE-K cycles to refine subsequent adaptation decisions. When the initially selected pick-and-place plan failed to remove the blockage, the resulting state was monitored and incorporated as additional runtime evidence. This evidence supported a more specific diagnosis in the following cycle and informed the reassessment of the predefined adaptation plans. The subsequent selection of the left-right script removed the blockage condition, illustrating how execution outcomes can influence later decisions within the feedback loop.

The anomaly indicator should be interpreted as a cycle-specific output of the Analyser rather than as the sole criterion for characterizing the overall situation as unanticipated. In the first cycle, the presence of an obstacle was handled as an anticipated uncertainty because a predefined adaptation option was available. After the

pick-and-place plan failed, the additional runtime evidence allowed the Analyser to identify that the obstacle height exceeded the range covered by the available adaptation knowledge. The condition was therefore classified as unanticipated with respect to the existing knowledge in the subsequent cycle, and the anomaly indicator was set to `true`.

Although the selected alternative removed the blockage condition, the overall task was not completed because the object did not reach the target. The reported execution should therefore be interpreted as preliminary evidence of iterative diagnosis and replanning rather than as evidence of successful end-to-end adaptation. Furthermore, although the proposed architecture includes an agentic fallback path to generate and evaluate a new adaptation plan when predefined alternatives are insufficient, this branch was not exercised during the reported execution because the predefined left-right script satisfied the acceptance criterion. Consequently, the present execution does not provide empirical evidence regarding the effectiveness of agent-generated adaptation plans.

The execution times provide an additional characterization of the runtime overhead observed in the current proof-of-concept implementation. The Analyser required approximately 80 seconds per cycle to process the runtime evidence and produce its diagnosis. In comparison, the Planner required an average of 182 seconds to evaluate the available tactics and produce its decision. Since the study did not define an explicit timing requirement or include a comparison baseline, these measurements should not be interpreted as evidence of real-time performance. Rather, they characterize the computational overhead of the current prototype and indicate that its present implementation should be regarded as a supervisory-level proof of concept rather than as a low-level, time-critical robotic control mechanism.

6 Threats to Validity

The conclusions of this work are based on a proof-of-concept evaluation with a single case study execution. Therefore, the results do not support claims of robustness or stable behaviour across different scenarios. Additional evaluations across multiple conditions and comparisons with traditional MAPE-K approaches are required to provide stronger empirical evidence.

Moreover, the complete agentic fallback branch was not exercised because an acceptable predefined adaptation plan remained available. Consequently, the current results do not provide empirical evidence regarding the generation or effectiveness of new plans produced by the Planner Agent.

7 Conclusion

This work proposed an Agentic MAPE-K architecture for self-adaptive robotic arm systems, augmenting the Analyser and Planner with agentic reasoning while preserving the modular structure of MAPE-K. The architecture was instantiated as a proof of concept in a simulated environment involving an obstacle whose height exceeded the anticipated range. The execution showed how an unsuccessful adaptation outcome can be incorporated as runtime evidence to refine the diagnosis and reassess predefined adaptation plans, leading to the selection of an alternative tactic that removed the blockage.

The results provide preliminary evidence of the feasibility of the proposed architectural integration and of using adaptation outcomes to inform subsequent MAPE-K decisions. However, the overall task remained incomplete, and the complete agentic fallback for generating and evaluating a new adaptation plan was not exercised because an acceptable predefined alternative remained available. Thus, the results should not be interpreted as validation of successful end-to-end or agent-generated adaptation.

Future work will evaluate the complete fallback branch under conditions in which predefined adaptation knowledge is insufficient, investigate independent mechanisms for validating generated diagnoses and plans, and assess task, capability, and safety constraints before execution. Further evaluations across multiple operating conditions, quantitative experiments, and comparison baselines are also required to assess the benefits, overhead, robustness, and limitations of the agentic components.

Artifact Availability

https://github.com/biaa04/agents_MAPE-K

References

[1] Saeid Nahavandi et al. “Machine learning meets advanced robotic manipulation”. In: *Information Fusion* 105 (2024), p. 102221.

[2] Gianluca Filippone et al. “Handling uncertainty in the specification of autonomous multi-robot systems through mission adaptation”. In: *Proceedings of the 19th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. 2024, pp. 25–36.

[3] Hassan Sartaj et al. “Software engineering for self-adaptive robotics: A research agenda”. In: *arXiv preprint arXiv:2505.19629* (2025).

[4] Danny Weyns. *An introduction to self-adaptive systems: A contemporary software engineering perspective*. John Wiley & Sons, 2020.

[5] Jeffrey O Kephart and David M Chess. “The vision of autonomic computing”. In: *Computer* 36.1 (2003), pp. 41–50.

[6] Sara M Hezavehi et al. “Uncertainty in self-adaptive systems: A research community perspective”. In: *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 15.4 (2021), pp. 1–36.

[7] Danny Weyns and Jesper Andersson. “From self-adaptation to self-evolution leveraging the operational design domain”. In: *2023 IEEE/ACM 18th Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE. 2023, pp. 90–96.

[8] Lei Wang et al. “A survey on large language model based autonomous agents”. In: *Frontiers of Computer Science* 18.6 (2024), p. 186345.

[9] Xu Huang et al. “Understanding the planning of llm agents: A survey”. In: *arXiv preprint arXiv:2402.02716* (2024).

[10] Frank D Macías-Escrivá et al. “Self-adaptive systems: A survey of current approaches, research challenges and applications”. In: *Expert Systems with Applications* 40.18 (2013), pp. 7267–7279.

[11] Didac Gil De La Iglesia and Danny Weyns. “MAPE-K formal templates to rigorously design behaviors for self-adaptive systems”. In: *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 10.3 (2015), pp. 1–31.

[12] Diego Perez-Palacin and Raffaela Mirandola. “Uncertainties in the modeling of self-adaptive systems: a taxonomy and an example of availability evaluation”. In: *Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering*. ICPE ’14. Dublin, Ireland: Association for Computing Machinery, 2014, pp. 3–14. ISBN: 9781450327336. DOI: 10.1145/2568088.2568095. URL: https://doi.org/10.1145/2568088.2568095.

[13] Lucas Alves et al. “MAPER: Extending MAPE-K with LLM-Based Reasoning to Manage Unanticipated Situations in Self-Adaptive Systems”. In: *Proceedings of the 21st International Conference on Software Engineering for Adaptive and Self-Managing Systems*. SEAMS ’26. Rio de Janeiro, Brazil: Association for Computing Machinery, 2026, pp. 245–256. ISBN: 979-8-4007-2445-9. DOI: 10.1145/3788550.3794886. URL: https://doi.org/10.1145/3788550.3794886.

[14] Divyansh Pandey et al. “POLARIS: Is Multi-Agentic Reasoning the Next Wave in Engineering Self-Adaptive Systems?” In: *arXiv preprint arXiv:2512.04702* (2025).

[15] Quentin Gallouédec et al. “panda-gym: Open-Source Goal-Conditioned Environments for Robotic Learning”. In: *4th Robot Learning Workshop: Self-Supervised and Lifelong Learning at NeurIPS* (2021).