The Journal of Systems and Software 217 (2024) 112166

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Reliable proactive adaptation via prediction fusion and extended stochastic model predictive control☆

Zhengyin Chen [a,b], Jialong Li [c], Nianyu Li [d,*], Wenpin Jiao [a,b]

[a] School of Computer Science, Peking University, Yiheyuan Road, 100871, Beijing, China
[b] Key Laboratory of High Confidence Software Technology (Peking University), MOE, China, Yiheyuan Road, 100871, Beijing, China
[c] Department of Computer Science and Engineering, Waseda University, 169-8050, Tokyo, Japan
[d] Zhongguancun Lab, Beijing, China

ARTICLE INFO

Dataset link: https://github.com/easton-chen/DART-SMPC.git

Keywords:
Proactive self-adaptation
DS-evidence theory
Model Predictive Control

ABSTRACT

Proactive self-adaptation has emerged as a vital approach in recent years, aiming to preemptively address potential goal violations or performance degradation, thus improving the system's reliability. However, this approach encounters specific challenges in prediction and decision-making, including issues such as erroneous predictions and adaptation latency. Addressing these issues, our study presents an innovative framework that leverages evidence theory to improve prediction accuracy and employs stochastic model predictive control (SMPC) for devising reliable adaptation strategies. We further refine the decision-making process by incorporating a latency-aware system model and a novel utility model inspired by the technical debt metaphor into the SMPC. Our framework's effectiveness is validated through experiments conducted on a cyber–physical system exemplar DARTSim, demonstrating notable improvements in prediction accuracy and system reliability within dynamic environments.

1. Introduction

Modern software systems are increasingly required to operate autonomously under changing conditions. This dynamic and uncertain environment will affect the reliability of the software system, undermining the ability of the software to consistently perform its intended functions without failure (Neufelder et al., 2016). Self-adaptation has been recognized as a crucial approach to manage runtime uncertainty, enabling software to adjust its behavior in response to environmental and internal changes so that it can consistently fulfill its requirements (Lemos et al., 2013). To achieve this, a self-adaptive software system should have certain self-* properties: self-configuring, self-healing, self-optimizing, and self-protecting. These properties are strongly related to software quality factors such as efficiency, availability, and especially reliability (Salehie and Tahvildari, 2009).

Traditional self-adaptive methods typically employ a goal violation trigger mechanism. This reactive feature may lead to delays in adaptation, potentially compromising the system's ability to prevent or mitigate goal violations effectively, thus reducing reliability (Moreno et al., 2015). Therefore, proactive adaptation emerges as a solution, aiming to foresee and mitigate potential disruptions before they impact the system. This approach involves predicting future context and system states to identify risks and evaluating different strategies to optimize performance (Cooray et al., 2013; Wang et al., 2018; Shin et al., 2021). It is particularly valuable in scenarios where delays in adaptation or the costs associated with goal violations are high. A notable implementation of proactive adaptation is through Model Predictive Control (MPC), which uses a system model to forecast the future state of the system and optimizes control actions by solving an iterative optimization problem (Wang et al., 2015; Angelopoulos et al., 2018; Ayala et al., 2021). MPC's ability to handle complex systems with multiple inputs and outputs makes it well-suited for the multifaceted nature of software systems, offering a sophisticated method to enhance reliability in uncertain conditions (Maggio et al., 2017).

Implementing MPC-based proactive adaptation introduces specific challenges that could impact its effectiveness. These challenges primarily revolve around the accuracy of predictions and decision-making processes. Firstly, the dependency on model prediction accuracy for MPC is a critical concern. As outlined by Fan et al. (2020) and Weyns (2020), MPC-based methods make decisions based on predictions of future states. The inherent uncertainty in predicting future contexts, essential for determining the system's behavior, poses a significant risk. Inaccurate predictions may lead to decisions that do not improve or might even compromise the software system, undermining its

☆ Editor: Prof. Raffaela Mirandola.

* Corresponding author.
E-mail addresses: chenzy512@pku.edu.cn (Z. Chen), lijialong@fuji.waseda.jp (J. Li), li_nianyu@pku.edu.cn (N. Li), jwp@pku.edu.cn (W. Jiao).

https://doi.org/10.1016/j.jss.2024.112166
Received 1 March 2024; Received in revised form 20 July 2024; Accepted 25 July 2024
Available online 26 July 2024

reliability. Secondly, the decision-making aspect of classical MPC approaches reveals two main problem-modeling issues affecting proactive adaptation. The initial issue is the oversight of adaptation latency—the delay between initiating an adaptation action and observing its effects (Keller and Mann, 2020). This oversight can result in a misjudgment of the adaptation strategy's effectiveness, leading to sub-optimal decisions that degrade software performance or even fail to adapt. Furthermore, there is often an inadequate analysis of control objectives within MPC frameworks. While the primary objectives typically include aligning system outputs with predefined setpoints and minimizing control costs, existing approaches often establish the objective function on an ad-hoc basis, devoid of systematic guidance. This may lead to less reliable objective functions that ignore critical aspects such as immediate and long-term costs, potentially compromising the soundness and thoroughness of the adaptation strategy.

To effectively tackle the challenges of inaccurate predictions and decision-making in MPC-based proactive adaptation, our approach introduces an innovative framework that integrates evidence theory with an enhanced version of stochastic Model Predictive Control. *Improving Prediction Accuracy:* To counter the issue of inaccurate predictions, our framework employs evidence theory to enhance the accuracy of context predictions by amalgamating historical prediction, and thus improve the system state prediction. This method mitigates the risks associated with premature predictions and erroneous predictions by incorporating a broader data spectrum, thereby ensuring more dependable outcomes and improving the system's reliability. Furthermore, we adopt stochastic Model Predictive Control (SMPC) to navigate the inherent uncertainty in context prediction, which manifests in probabilistic form. This approach involves solving a stochastic control problem that accounts for the probabilistic nature of future contexts, aiming to identify the adaptation strategy with the highest expected utility. This strategy exhibits greater robustness, capable of tolerating prediction errors and reducing false adaptations, in contrast to solely relying on predictions with the highest probability. *Refining the Decision-making Process:* To overcome inaccuracies in the problem-modeling aspect of decision-making, our framework introduces two key innovations. First, a latency-aware system model incorporates the delay between the initiation and the effect of adaptation actions, preventing premature adaptations that impair system performance or untimely adaptations that fail to be effective. Second, we utilize a utility model inspired by the technical debt metaphor to evaluate adaptation strategies comprehensively. This model assesses strategies based on their costs (both immediate and long-term) and benefits, allowing for a more systematic and nuanced analysis.

This paper contributes to the field of proactive software adaptation in three key areas: (1) We introduce a context prediction enhancing method based on evidence theory, which improves prediction accuracy and reduces the uncertainty in future predictions by integrating historical prediction data. (2) We propose a novel decision-making framework leveraging SMPC tailored for uncertain predictions. Furthermore, we incorporate the SMPC with a latency-aware system model and a utility model based on the technical debt metaphor, enabling more precise and informed decision-making by accounting for adaptation timing and cost–benefit analysis. (3) We validate our approach through experiments in a case study, demonstrating its effectiveness in enhancing both the accuracy of context predictions and the efficacy of decision-making in reliable proactive software adaptation.

The remainder of the paper is organized as follows. Sections 2 and 3 discuss related work and background knowledge. Section 4 provides an overview of our framework. The prediction fusion method and decision-making mechanism are introduced in Sections 5 and 6. Experiments are conducted in Sections 7 and 8 concludes the paper.

2. Related work

2.1. Context prediction

Context prediction plays a pivotal role in proactive self-adaptation, essential for forecasting the system's future state and its interaction with dynamic environments. Traditionally, the influence of environmental changes on system behavior has been implicitly modeled, with variations in system operations serving as indicators of external changes. For instance, Cooray et al. (2013) employ discrete-time Markov chains to model system components and utilize hidden Markov models to deduce the impact of environmental shifts on transition probabilities, thereby estimating system reliability. Similarly, Tanabe et al. (2017) represents the environment and system behavior using a label transition system, updating the model in real-time via stochastic gradient descent. Moreno et al. (2015) adopt a more explicit approach by employing an auto-regressive (AR) time series predictor alongside a probability tree to articulate the uncertainty inherent in environment predictions. Moreover, Shin et al. (2021) introduces PASTAA, a technique grounded in statistical model checking that leverages historical data for environment forecasting.

Contrary to these methodologies, our approach seeks to augment existing prediction techniques by integrating historical prediction data, thereby refining the accuracy of future forecasts through evidence theory. This strategy does not prescribe a specific prediction model; instead, it necessitates that the prediction aligns with the requirements set forth by evidence theory, enhancing versatility and applicability across various predictive models. In the subsequent section, we elucidate the fundamental principles of evidence theory. To our knowledge, this marks the first instance of applying evidence theory to context prediction within the realm of proactive self-adaptation, signifying a novel contribution to the field.

2.2. Model predictive decision-making

Proactive adaptation research extensively incorporates MPC concepts from control theory to enhance software system adaptability. Kusic et al. (2009) applied MPC to data centers, enabling dynamic resource provisioning that optimizes server utilization and energy efficiency while ensuring quality of service. Similarly, the CobRA method (Angelopoulos et al., 2016, 2018) integrates MPC with requirements engineering, utilizing an extended requirements model to articulate the system's goals and adjustable parameters. The ProDSPL method (Ayala et al., 2021) merges MPC with a feature model to optimize key performance indicators within the constraints of the feature model, illustrating the versatility of MPC in addressing diverse adaptation challenges.

Contrary to these direct applications of control theory, the PLA approach (Moreno et al., 2015) aligns with the overarching concept of MPC but diverges in its implementation. PLA adopts an architecture-based strategy, employing Markov decision processes for system modeling and addressing uncertainty through probabilistic model checking and stochastic dynamic programming (Moreno et al., 2018). Despite sharing a conceptual foundation with MPC, PLA and MPC-based methods differ significantly in their implementation (Moreno et al., 2017). PLA necessitates domain-specific knowledge for system modeling, whereas MPC-based methods can leverage system identification techniques. Furthermore, PLA conceptualizes adaptation strategies as tactics, while MPC approaches utilize direct control variables (Moreno et al., 2017).

Our methodology extends classical MPC in three significant ways. Firstly, it incorporates a latency-aware system model to account for the delay between the initiation of adaptation decisions and their outcomes. Secondly, we introduce a utility model inspired by the technical debt metaphor, guiding the establishment of control objectives with a focus on long-term cost–benefit analysis. Lastly, we apply

stochastic control to manage the inherent uncertainty in contextual information. These enhancements aim to refine decision-making processes in proactive adaptation, offering a comprehensive approach that balances immediate system needs with long-term viability and robustness against uncertainty.

2.3. Adaptation latency and effect assessment

A crucial aspect of timely proactive adaptation is accounting for the latency associated with executing adaptation behaviors. Cámara et al. (2014) utilized stochastic games to model systems with latency awareness, capturing the delay associated with adaptation tactics and their mutual exclusions. Building on this foundation, Moreno et al. (2015) introduced the PLA-PMC method, leveraging probabilistic model checking for proactive adaptation in uncertain environments. This methodology was further refined in the PLA-SDP method (Moreno et al., 2018) to eliminate the runtime overhead. Zhang et al. (2020) explored adaptation lead time in service-based systems through the Letpa method, employing dynamic programming to manage adaptation across different levels of control parameters. Palmerino et al. (2019) introduced the concept of tactic volatility, emphasizing the significance of accurately predicting the runtime cost and latency of tactics to ensure effective proactive adaptation. They utilized a Multiple Regression Analysis (MRA) model to forecast tactic performance, highlighting the importance of precision in adaptation planning. Li et al. (2021) shifted focus to the human aspects of adaptation, investigating the optimal lead time for engaging individuals in adaptation tasks through stochastic multi-player games. This study considers task complexity and training levels, offering insights into human-centered adaptation strategies. Despite these advancements, latency considerations have remained largely unexplored within MPC-based proactive adaptation. To bridge this gap, our work extends the traditional state–space model used in MPC to incorporate latency representations. This enhancement enables a more comprehensive and effective application of MPC in managing proactive adaptation, allowing for the consideration of adaptation execution delays and their impact on system performance.

In MPC, the effectiveness of the control is evaluated through the formulation of an objective function, which commonly addresses the deviation between system output and desired setpoints, alongside control costs. In many MPC-based self-adaptation methodologies, the definition of this objective function is tailored to the specific characteristics of the system under consideration. For instance, in studies such as Angelopoulos et al. (2018) and Wang et al. (2015), variations in control parameters are treated as costs, while in Ayala et al. (2021), the weights of enabled features are prioritized. Within the realm of adaptation, there exists a body of research aimed at quantifying the impact of adaptation using financial metrics like Return on Investment (ROI) (Gerostathopoulos et al., 2022), or concepts from software engineering such as technical debt (Chen et al., 2018; Kumar et al., 2019). Leveraging this perspective, we introduce a utility model in this paper, drawing from the technical debt metaphor, to systematically and comprehensively define the objective function.

3. Background

3.1. Evidence theory

Evidence theory is a method of fusing multiple sources of information to eliminate uncertainty, also known as Dempster–Shafer theory (Dempster, 1967; Shafer, 1976). Evidence theory is based on two ideas: the idea of obtaining degrees of belief about an issue from the subjective probabilities of related issues, and Dempster's rule for combining these degrees of belief when they are based on independent evidence. We begin by introducing the basic concepts of evidence theory.

The frame of discernment Ω is an exhaustive set of all hypotheses of a problem, all of which are mutually exclusive. Let Ω contain N elements, Ω can be expressed as:

$$\Omega = \{H_1, H_2, \ldots, H_N\} \tag{1}$$

A subset A of Ω is called a proposition and the power set 2^{Ω} of Ω denotes all subsets of Ω. A function $m : 2^{\Omega} \rightarrow [0, 1]$ is called a *basic probability assignment* or *mass function* if it has the following two important properties. The mass function assigns a probability to each proposition in the frame of discernment.

$$m(\emptyset) = 0 \tag{2}$$

$$\sum_{A \in 2^{\Omega}} m(A) = 1 \tag{3}$$

The Dempster combination rule (denoted by $m = m_1 \oplus m_2$) is used to combine two independent mass functions (m_1 and m_2). The combined mass function m contains information from m_1, m_2 to mitigate uncertainty. For all non-empty set $A \in 2^{\Omega}$, It is defined as:

$$m(A) = \frac{1}{1-K} \sum_{B,C \in 2^{\Omega}, A = B \cap C} m_1(B) \cdot m_2(C) \tag{4}$$

where,

$$K = \sum_{B \cap C = \emptyset} m_1(B) \cdot m_2(C) \tag{5}$$

The K is known as the conflict factor, which reflects the degree of conflict between two pieces of evidence. The Eq. (4) can be interpreted as direct support about A after removing the conflicting parts of the two pieces of evidence.

Evidence theory is characterized by (1) Satisfying a weaker condition than Bayesian theory, i.e., it does not have to satisfy probabilistic additivity. (2) It can directly express "uncertainty" and "don't know", information that is expressed in the mass function and which is preserved during the combination. (3) It allows one to assign belief not only to a single element of the hypothesis space but also to a subset of it, much like the human process of evidence gathering at all levels of abstraction.

3.2. Technical debt

In software engineering, technical debt refers to a metaphor that reflects technical compromises that can produce short-term benefits but may compromise the long-term health of a software system (Cunningham, 1993). To visualize technical debt, interest and principal are most commonly used to describe and explain the technical debt concept. In related work (Chen et al., 2018), three components, principal, interest, and revenue, are proposed to characterize the debt problem in adaptation, so as to assess the need for adaptation. Where principal refers to a one-time investment in an asset (e.g., software); interest is the additional cost of the asset that accumulates over time; and revenue is the return on the asset that accumulates over time. Technical debt is equal to the sum of $Principal$ and $Interest$, its net value (net debt) is calculated as $Principal + Interest - Revenue$, and the adaptation goal is to minimize net debt.

In software development, technical debt is usually calculated in monetary terms. However, in self-adaptive systems, utility values encompassing costs and benefits are often used instead to simplify the assessment of technical debt, focusing on runtime performance and quality attributes. These attributes can be monetarily quantified using service level agreements (SLA Andrieux et al., 2007). To enhance accuracy, the cost of the system is divided into principal and interest in technical debt, allowing for more precise utility value specifications.

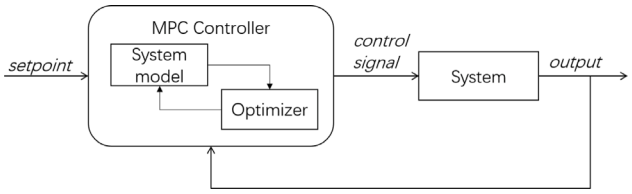

Fig. 1. Overview of Model Predictive Control.

3.3. Model predictive control

Model Predictive Control is a widely used modern control technique as it offers a compromise between optimality and computation cost (Kouvaritakis and Cannon, 2016). The overview process of MPC is shown in Fig. 1. It uses a system dynamic model (e.g. Eq. (6)) to represent the relationship of the system state $\mathbf{x}(t+1)$ to the previous system state $\mathbf{x}(t)$ and control inputs $\mathbf{u}(t)$, as well as the relationship of the system output $\mathbf{y}(t)$ to the system state $\mathbf{x}(t)$ and control inputs $\mathbf{u}(t)$ (where t indicates the time). Here we use bold letter notation to indicate the case of multiple inputs and multiple outputs.

$$\begin{aligned} \mathbf{x}(t+1) &= f(\mathbf{x}(t), \mathbf{u}(t)) \\ \mathbf{y}(t) &= g(\mathbf{x}(t)) \end{aligned} \tag{6}$$

In each control loop, the controller is required to solve a constrained optimization problem, that is to find a control signal sequence $\mathbf{u}* = \langle \mathbf{u}(0), \ldots, \mathbf{u}(h-1) \rangle$ that makes the objective function optimal over a horizon h under certain constraints, as shown in Eq. (7). The objective function usually consists of two terms, representing the terminal objective ($m(\cdot)$) and the intermediate stage objective ($l(\cdot)$). The terminal objective focuses on the final output of the system after the control, i.e. $y(h)$, while the intermediate stage objective focuses on the output and control costs of the intermediate phase, i.e. $x(i), u(i), y(i)$. The estimated system outputs need to be calculated through the system dynamic model.

$$\begin{aligned} \min_{\mathbf{u}(t),\ldots,\mathbf{u}(t+h-1)} \quad & m(\mathbf{y}(t+h)) + \sum_{i=0}^{h-1} l(\mathbf{x}(t+i), \mathbf{u}(t+i), \mathbf{y}(t+i)) \\ \textit{subject to} \quad & \mathbf{x}(i+1) = f(\mathbf{x}(i), \mathbf{u}(i)) \\ & \mathbf{y}(i) = g(\mathbf{x}(i)) \\ & \mathbf{u}_{min} \le \mathbf{u}(i) \le \mathbf{u}_{max}, \\ & \mathbf{x}_{min} \le \mathbf{x}(i) \le \mathbf{x}_{max}, \\ & i = 0, 1, 2, \ldots, h-1, \end{aligned} \tag{7}$$

In general, MPC follows the receding horizon principle, that is, in each control loop, the controller will calculate a sequence of control signals, but the controller will only retain the first term of this sequence as a control signal and discard the rest, and repeat the calculation process in the subsequent control loop using the newly measured system state. Therefore, MPC is also known as Receding Horizon Control. Due to its ability to solve optimization problems with constraints that can help improve the long-term performance of systems, MPC has already been used for the implementation of proactive self-adaptive systems.

3.4. Running example: DARTSim

DARTSim (Moreno et al., 2019), a simulation of unmanned air vehicles (UAVs) on a reconnaissance mission, serves as our running example. In this simulation, UAVs have to detect targets on the ground as they fly a planned route at constant speed. However, there are threats on the ground along the route that can attack the team. The closer the UAVs fly to the ground, the more likely they are to detect the targets, but also the higher the probability of being detected by threats. This poses a trade-off for the team, which is complicated by the uncertainty about the environment. The team can use tactics that include changing altitude, changing formation, and using electronic countermeasures (ECM) to change its behavior. In this scenario, the goal of the system is to maximize the number of detected targets and minimize the probability of being detected by threats. In particular tasks, it may add a constraint that considers detection by threats as task failure. Therefore, a reliable self-adaptive system needs to guarantee the fulfillment of the above goal with minimal adaptation costs. We choose DARTSim as our running example due to its embodiment of real-world decision-making under uncertainty, thus emphasizing the reliability of the software system. In addition, some of DARTSim's adaptation behaviors may have delays, such as increasing altitude, which raises the requirement of adaptation timeliness.

4. Approach overview

Our framework advances the reliability of the proactive self-adaptive system, as depicted in Fig. 2, by building on and enhancing the MAPE-K model (Kephart and Chess, 2003) with a unique methodology that diverges from the conventional MAPE-K loop. It comprises components such as Monitor, Context Predictor, Evidence Theory-based Prediction Fusion, Stochastic Model Predictive Control (SMPC) Planner, and Executor, with a Knowledge Base acting as a central repository for essential data and models facilitating adaptation. Due to spatial limitations, the data exchange between the Knowledge Base and other elements is described without specific labels. Together, the Context Predictor, Prediction Fusion, and SMPC Planner execute the Analysis and Planning aspects of the MAPE loop. The adaptation mechanism is structured into two primary phases: prediction fusion and decision-making.

Prediction Fusion Phase: This phase aims to enhance the accuracy of predictions by reducing uncertainty. The Monitor captures and relays context and system state data to the Knowledge Base and Context Predictor. The latter forecasts future context states, which are then refined with historical predictions through Evidence Theory in the prediction fusion process. This approach results in more accurate and reliable predictions, laying the groundwork for effective decision-making.

Decision-Making Phase: This phase focuses on devising the optimal adaptation strategy, defined as a series of control parameters. Utilizing the predictions and current system state, the planner formulates a control objective informed by a technical debt-based utility model and assesses future system states considering adaptation latency through a latency-aware system model. Given that predictions are probabilistic, the planner solves a stochastic control problem to identify the optimal control sequence that achieves the control objective while adhering to specific constraints.

Once the optimal control sequence is determined, the Executor implements the initial control action following the MPC's receding horizon principle. Subsequent actions are retained for potential future use, ensuring readiness for situations where immediate re-planning is necessary.

In DARTSim, the prediction fusion phase begins by monitoring system data (such as altitude, formation, and ECM) and context data (including images from forward-looking sensors), all stored in the Knowledge Base. The context predictor then takes this data to estimate the likelihood of encountering targets or threats, utilizing sensor probabilities for each area ahead. Although the exact estimation methodology is not detailed in this discussion, it is important to note that these initial probabilities are refined through evidence theory in the prediction fusion phase, enhancing the accuracy of the likelihood of encounters.

Following this, in the decision-making phase, the planner accesses the updated probabilities and the UAVs' current configuration from the Knowledge Base. With this information, it generates a control sequence by solving a stochastic control problem, which includes adjustments in altitude, formation, and ECM strategies for the upcoming control horizon. The executor is the final step in this process, where the

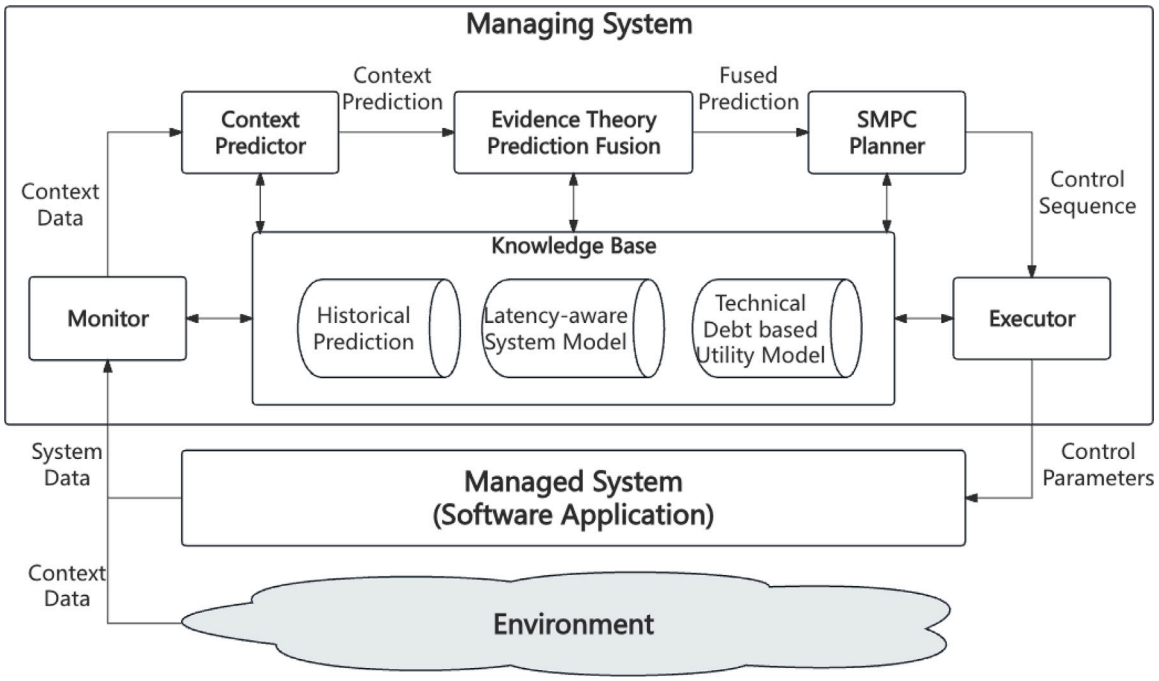

Fig. 2. Approach overview.

Table 1
Context prediction and fusion at time t.

	$t+1$	...	$t+i$	...	$t+h$
$t-h+1$	$PredCv(t+1 \mid t-h+1)$	–	–	–	–
...	...	...	...	...	...
$t-k$	$PredCv(t+1 \mid t-k)$	...	$PredCv(t+i \mid t-k)$	...	–
...	...	...	...	...	...
$t-1$	$PredCv(t+1 \mid t-1)$	...	$PredCv(t+i \mid t-1)$	...	–
t	$PredCv(t+1 \mid t)$	...	$PredCv(t+i \mid t)$	...	$PredCv(t+h \mid t)$
fused prediction at t	$\widehat{PredCv}(t+1 \mid t)$	...	$\widehat{PredCv}(t+i \mid t)$	...	$\widehat{PredCv}(t+h \mid t)$

planned adjustments are implemented, altering the UAVs' operational parameters to better suit the predicted environment. This adaptation approach allows for a more nuanced and prepared response to potential threats or targets, leveraging advanced prediction and decision-making processes to optimize the UAVs' performance in dynamic and uncertain scenarios.

5. Prediction fusion based on evidence theory

5.1. Context prediction

Self-adaptive software systems operate autonomously in dynamically changing environments. We define the portion of the environment pertinent to the software system's goals as the context (Ali et al., 2010), represented by a set of context variable values, $\mathbf{cv}(t) = (cv_1(t), \ldots, cv_{N_{cv}}(t))$, where cv_i denotes a context variable, N_{cv} the total number of context variables, and t indicates time.

For illustrative purposes, our discussion narrows to a single context variable, cv. We model cv as a discrete random variable with a range of values $ValueRange_{cv} = \{val_1, \ldots, val_{m_{cv}}\}$. The context prediction function, $PredCv(t+i \mid t)$, forecasts the probability distribution of future value $cv(t+i)$ at time t, such that $PredCv(t+i \mid t) : P(cv(t+i) = val_j) = p_j$, where $p_j \in [0,1]$ and $\sum_{j=1}^{m_{cv}} p_j = 1$. In multi-variable scenarios, we consider each variable independently for prediction purposes. This paper does not delve into specific prediction methodologies but requires that predictions conform to the aforementioned probabilistic structure, accommodating various problem-specific prediction techniques.

Context prediction is a cornerstone of proactive self-adaptation, though it faces challenges related to prediction accuracy, particularly due to the inherent uncertainties in forecasting. Predictions are generally more accurate when made closer to the event, leveraging the most recent data (Wang and Pazat, 2012). However, uncertainties, including outliers, can compromise the reliability of new forecasts. To mitigate this, we introduce an evidence theory-based method that integrates historical predictions to enhance both the timeliness and accuracy of forecasts. The choice of evidence theory is motivated by its ability to represent prediction uncertainties through mass functions, thereby improving the reliability of context predictions by aggregating evidence over time.

5.2. Prediction fusion

We posit that the predictor operates with a prediction horizon of h. At any given moment t, it is capable of generating forecasts for future contexts spanning from $t+1$ to $t+h$, represented as $PredCv(t+1 \mid t), \ldots, PredCv(t+h \mid t)$. Traditionally, these forecasts are directly utilized for analysis and planning. However, in uncertain environments, reliance on a singular forecast can introduce substantial inaccuracies or be compromised by sensor malfunctions, making the most recent forecast unavailable. At such junctures, predictions from prior time steps, such as those made at $t-1$ for contexts at $t, \ldots, t+h-1$, spanning $PredCv(t+1 \mid t-1), \ldots, PredCv(t+h-1 \mid t-1)$, become invaluable for enhancing prediction accuracy.

To this end, we advocate for the application of evidence theory to amalgamate historical predictions, as detailed in Table 1. For predicting

the value of $cv(t+i)$, we designate the frame of discernment as $\Omega = \{cv(t+i) = val_1, \ldots, cv(t+i) = val_{m_{cv}}\}$. Each forecast, $PredCv(t+i \mid t)$, is regarded as a mass function. Given the predictor's horizon limit to h future periods, at time t, not all prior predictions remain valid, as elucidated in Table 1. At t, predictions from $t-h+1$ to t are considered. For the fused forecast at $t+1$, predictions from $t-h+1$ to t are amalgamated using the Dempster combination rule to form $\hat{PredCv}(t+1 \mid t)$. This methodology is systematically applied to integrate forecasts for subsequent times.

Nevertheless, predictors often grapple with uncertainties that may yield predictions with substantial errors, a scenario referred to as the conflicting evidence phenomenon. The conventional Dempster combination rule, as introduced in Section 3.1, might produce unsatisfactory outcomes when encountering conflicting evidence. Thus, we employ the discount Dempster combination rule that assigns a trust coefficient α_k to each prediction $PredCv(t+i \mid t-k)$, symbolized as $PredCv_{\alpha_k}(t+i \mid t-k)$ after adjustment. This adjustment is governed by Eqs. (8) and (9), with α_k inversely proportional to the distance from the prediction point, aligning with the premise that predictions become less reliable as the temporal distance from the prediction point increases (Metzger et al., 2019; Wang and Pazat, 2012).

$$PredCv_{\alpha_k} : P_{\alpha_k}(cv(t+i) = val_j) = p_j * \alpha_k \tag{8}$$

$$\begin{aligned} PredCv_{\alpha_k} : P_{\alpha_k}(&cv(t+i) = val_1 \vee cv(t+i) = val_2 \vee \\ &\cdots \vee cv(t+i) = val_{m_{cv}}) = 1 - \alpha_k \end{aligned} \tag{9}$$

In the context of the DARTSim, let us consider a context variable, denoted as cv_{tar}, which has a binary value range $ValueRange_{cv_{tar}} = \{0,1\}$, indicates the presence or absence of a target. The frame of discernment for $cv_{tar}(t+i)$ is $\Omega = \{cv_{tar}(t+i) = 0, cv_{tar}(t+i) = 1\}$. Consider an example that the prediction for $t+1$ at $t, t-1$ and $t-2$ as:

$PredCv_{tar}(t+1 \mid t)$:

$P(tar(t+1) = 0) = 0.35, P(tar(t+1) = 1) = 0.65$

$PredCv_{tar}(t+1 \mid t-1)$:

$P(tar(t+1) = 0) = 0.4, P(tar(t+1) = 1) = 0.6$

$PredCv_{tar}(t+1 \mid t-2)$:

$P(tar(t+1) = 0) = 0.45, P(tar(t+1) = 1) = 0.55$

The result of this fusion is $\hat{PredCv}_{tar}(t+1 \mid t) : P(cv_{tar}(t+1) = 0) = 0.227, P(cv_{tar}(t+1) = 1) = 0.773$, showing that by fusing the historical predictions we are more confirming the conclusion that $tar(t+1) = 1$.

Further, if an incorrect prediction is obtained at t, i.e $PredCv_{tar}(t+1 \mid t) : P(cv_{tar}(t+1) = 0) = 0.8, P(cv_{tar}(t+1) = 1) = 0.2$. The final fusion result will be hugely impacted: $\hat{PredCv}_{tar}(t+1 \mid t) : P(cv_{tar}(t+1) = 0) = 0.686, P(cv_{tar}(t+1) = 1) = 0.314$. Using the discount Dempster combination rule is an effective way to improve this situation. The exact result is related to the value of the discount factor α. For example, when $\alpha = 0.8, P(cv_{tar}(t+1) = 1) = 0.403$.

6. Decision-making based on extended stochastic MPC

In this section, we outline our approach to making adaptation decisions through an enhanced version of MPC, designed specifically for proactive adaptation strategies. We start by clarifying the terms and notations related to MPC used in this paper and then introduce our planning method.

Our discussion on stochastic MPC is structured to first introduce a latency-aware system model. This model updates MPC by considering the time delays in adaptation actions, making it more suited for proactive adjustments. Following this, we leverage the concept of technical debt as a metaphorical framework to describe the utility derived from adaptation efforts, thereby formulating a holistic control objective that encapsulates both immediate and deferred adaptation benefits and costs. Finally, we address how to manage uncertainties in the system's context by framing it as a stochastic control problem. This method ensures our decision-making process is not only grounded in the robust analytical framework of MPC but also innovatively augmented and practical to meet the evolving demands of proactive self-adaptation.

6.1. Notations in MPC-based proactive adaptation

In Section 3.3, we delineate the foundational structure of the system model within MPC, encapsulated by Eq. (6). This equation elucidates the dynamic interplay between the subsequent system state $\mathbf{x}(t+1)$, its preceding state $\mathbf{x}(t)$, control inputs $\mathbf{u}(t)$, and the correlation of the system output $\mathbf{y}(t)$ with both the system state $\mathbf{x}(t)$ and control inputs $\mathbf{u}(t)$. Within the context of self-adaptive software systems, the components of Eq. (6) are conceptualized as follows:

- y represents the performance indicators that are directly related to the system goals. $\mathbf{y}(t) = (y_1(t), \ldots, y_{N_y}(t))$, N_y indicates the number of performance indicators.
- x represents the system configurations that are affected by control parameters and determine the performance indicators. $\mathbf{x}(t) = (x_1(t), \ldots, x_{N_x}(t))$, N_x indicates the number of configuration items.
- u represents the control parameters that can be modified to alter the behavior or configuration of the system. $\mathbf{u}(t) = (u_1(t), \ldots, u_{N_u}(t))$, N_u indicates the number of control parameters.

Prior to addressing the concept of latency, it is imperative to integrate contextual factors into the system model. Employing the notation established in Section 5, we represent the collective value of context variables at time t using the bold notation $\mathbf{cv}(t)$. This inclusion of context variables enriches the system model, now depicted as Eq. (10).

$$\begin{aligned} \mathbf{x}(t+1) &= f(\mathbf{x}(t), \mathbf{u}(t), \mathbf{cv}(t)) \\ \mathbf{y}(t) &= g(\mathbf{x}(t), \mathbf{cv}(t)) \end{aligned} \tag{10}$$

For example, Using the above notations, the DARTSim case can be represented as:

- Performance indicators (y): $\mathbf{y} = (y_{tar}, y_{thr}), y_{tar} \in [0,1], y_{thr} \in [0,1]$, indicates the probability of the UAVs detecting a target and the probability of a threat has detected the UAVs.
- Configurations (x): $\mathbf{x} = (x_a, x_f, x_e), x_a = [0, x_{a,max}], x_f = [0,1], x_e = \{0,1\}$. Where x_a, x_f, x_e indicate three configurations. x_a indicates the UAVs' flight altitude, with $x_{a,max}$ being the highest. x_f indicates the UAVs' formation, with 0 to 1 representing loose to tight. x_e indicates whether the ECM is on or off.
- Control parameters (u): $\mathbf{u} = (u_a, u_f, u_e)$, indicate the adjustment to the corresponding configuration.
- Context variables (cv): $\mathbf{cv} = (cv_{tar}, cv_{thr}), cv_{tar} \in \{0,1\}, cv_{thr} \in \{0,1\}$, indicates whether the target and threat exist.

Both the context and system configuration will affect the system's performance. The primary objective of adaptation is then to get to the most suitable configuration by applying control parameters when the context changes. This ideal configuration optimizes the performance indicator, aligning it with the desired goals.

The input of the planner is the context prediction and the current system configuration, while the output is the adaptation strategy, a sequence of control parameters, i.e. $AS(t,h) = \langle \mathbf{u}(t), \ldots, \mathbf{u}(t+h-1) \rangle$.

6.2. Latency-awareness in MPC

In this subsection, we expand upon the state–space model typically employed in MPC to accommodate adaptation latency. This enhancement enables more precise system predictions during decision-making, optimizing outcomes and preventing premature or untimely proactive adaptations.

In self-adaptive software systems, various types of latency impact the efficiency and responsiveness of adaptations (Keller and Mann, 2020). These include delays in monitoring, analysis, planning, execution of adaptations, and the manifestation of adaptation effects. Particularly, execution latency (l_E) and the latency for the effects of adaptation to become manifest (l_{E2}) are directly relevant to the system model, necessitating their incorporation to foster latency-aware MPC. For simplicity, we initially consider latency in the context of a singular system configuration x and control parameter u, though the approach is scalable to multiple configurations and parameters.

l_E: Execution latency in MPC-based adaptation refers to the delay in adjusting the control parameters. It is manifested in the system model by the fact that the currently planned control parameters do not affect the system configuration immediately, but require some initiation time, i.e., $x(t+1) = f(x(t), u(t-l))$, where $l \geq 1$ signifies a latency of l time steps. The specific latency value l is determined by system characteristics. For instance, if activating the ECM requires a certain duration, the control parameter u_e will experience a delay before affecting the system configuration x_e.

To accommodate this latency within our system model, we introduce auxiliary states $x_1, \ldots, x_l$, which facilitate the transmission of past control actions $u(t-l)$ to the system state $x_l(t)$, and then affect the $x(t+1)$. The augmented system model, incorporating these auxiliary states, is formalized as Eq. (11).

$$\begin{aligned} x(t+1) &= f(x(t), x_l(t), cv(t)) \\ x_l(t+1) &= x_{l-1}(t) \\ &\cdots \\ x_1(t+1) &= u(t) \end{aligned} \tag{11}$$

By transferring the value of $u(t-l)$ to $x_l(t)$ and integrating the original system state vector $\mathbf{x}$ with auxiliary states $x_1, \ldots, x_l$ into an expanded state vector $\overline{\mathbf{x}} = (\mathbf{x}, x_1, \ldots, x_l)$, we encapsulate this relationship in a new function $\overline{f}$. This formulation allows the extended system model to encapsulate execution latency l_E within the framework of a generalized model, as represented in Eq. (12). Thus, the system model employed in subsequent sections adheres to this original, yet augmented, representation.

$$\begin{aligned} \overline{\mathbf{x}}(t+1) &= \overline{f}(\overline{\mathbf{x}}(t), \mathbf{u}(t), \mathbf{cv}(t)) \\ \mathbf{y}(t) &= \overline{g}(\overline{\mathbf{x}}(t), \mathbf{cv}(t)) \end{aligned} \tag{12}$$

l_{E2}: This latency captures the delay in realizing the full effect of an adaptation within the system's dynamics. Unlike static models where system states directly correlate with control parameters, the dynamic model illustrates how the system state evolves over time due to control parameters. For example, for a linear system $x(t+1) = x(t) + b * u(t)$, the latency l_{E2} can be reflected by the coefficients b. A smaller value of b implies that the control parameters induce slower adjustments in the system state, resulting in increased latency. The original state space model in MPC is a dynamic model that already takes into account l_{E2}. For instance, adjusting the control parameter u_a does not instantaneously elevate the UAVs' altitude x_a to the desired state but introduces a delay.

The derivation of the system model can be approached from two perspectives: The first leverages domain knowledge to analytically model the software system's behavior, such as employing a queuing network model for a web service (Incerto et al., 2017). The second perspective treats the system as a black box, utilizing system identification techniques to empirically derive the model (Ljung, 1998). This approach initially hypothesizes a model structure, followed by the collection of input and output data through simulations or preliminary runs, to subsequently refine the model to accurately represent the system dynamics.

6.3. Comprehensive objective function based on technical debt

The core principle guiding the formulation of control objectives in MPC is to align the system's output closely with predefined setpoints while minimizing associated control costs. Technical debt underscores the importance of striking a balance between costs and benefits, considering both short-term and long-term implications. This philosophy aligns well with the objective of optimizing long-term performance in proactive adaptation. By embracing the technical debt metaphor, we adopt a nuanced approach to systematically defining control objectives, thereby enhancing decision-making capabilities.

Incorporating insights from relevant literature (Chen et al., 2018) and the technical debt metaphor, we employ the notions of *Principal*, *Interest*, and *Revenue* to evaluate the utility of an adaptation strategy $AS(t,h)$. Our definitions are as follows:

- **Principal:** $Principal = P(\mathbf{u}(t))$, is the overhead incurred due to the execution of the adaptation strategy at t. The overhead of execution is the sum of all the control parameters, $P(\mathbf{u}(t)) = \sum_{i=1}^{N_u} u_i(t) * cost_{u_i}$. For instance, in the DARTSim scenario, the principal encompasses the costs for adjustments in altitude and formation, assuming no cost for ECM adjustments.
- **Interest**: Expressed as $Interest = I(\mathbf{x}(t))$, captures the ongoing costs associated with the system's state post-adaptation. These costs vary with the system's configuration and accumulate over time. The calculation involves the product of each configuration's value and its respective unit cost, $I(\mathbf{x}(t)) = \sum_{i=1}^{N_x} x_i(t) * cost_{x_i}$. In DARTSim, interest reflects ECM costs, assuming a uniform cost across different altitudes and formations.
- **Revenue**: Denoted as $Revenue = R(\mathbf{y}(t))$, quantifies the benefits derived from executing the adaptation strategy. It assesses goal achievement by comparing the deviation between post-execution performance indicators y_i and their reference values ref_{y_i}, thereby computing the revenue, $R(\mathbf{y}(t)) = \sum_{i=1}^{N_y}(y_i(t) - ref_{y_i}) * revenue_{y_i}$. Here, 'revenue' also accounts for penalties incurred for unmet goals. In the DARTSim example, revenue includes the advantages of detecting targets and the penalties for encountering threats. By assigning different revenues to different performance indicators, it is possible to make system emphasize specific goals and adopt more conservative or more aggressive behavior.

The net utility of an adaptation strategy is thus evaluated as $Revenue - Principal - Interest$. Taking into account the control horizon h, we articulate the objective function as outlined in Eq. (13), facilitating a comprehensive assessment of adaptation strategies within an MPC framework. In this way, we express the ability of the system to reliably achieve its goals using net utility, thus providing a basis for subsequent optimal control.

$$J(AS(t,h), \mathbf{x}(t)) = \sum_{i=1}^{h} R(\mathbf{y}(t+i)) - P(\mathbf{u}(t+i-1)) - I(\mathbf{x}(t+i)) \tag{13}$$

6.4. Stochastic control logic

In the preceding subsections, we augmented MPC with latency-aware capabilities and established a systematic approach for formulating the objective function. Yet, our discussions hitherto have presumed deterministic contexts, overlooking the inherent uncertainty associated with context predictions. Sole reliance on the most probable context prediction disregards the potential for inaccuracies, potentially leading to the generation of adaptation strategies that may falter under erroneous predictions. To mitigate this, our study adopts a stochastic MPC approach for adaptation planning, aiming to encompass all conceivable future contexts. This approach is intended to forge a more reliable adaptation strategy capable of addressing uncertainty effectively.

We define a context scenario at t as a possible context variable taking values from $t+1$ to $t+h$, $sc^j(t) = \langle \mathbf{cv}^j(t+1), \dots, \mathbf{cv}^j(t+h) \rangle, j = 1, \dots, N_{sc}$, N_{sc} indicates the number of scenarios. Follow the assumption of the previous section that there is only one context variable cv. Then a scenario $sc^j(t)$ can be expressed as $sc^j(t) = \langle val_{j_{t+1}}, \dots, val_{j_{t+h}} \rangle, val_{j_{t+i}} \in ValueRange_{cv}$. And, given the context prediction $\hat{PredCv}(t+i \mid t), i = 1, \dots, h$, the probability of this context scenario can be calculated using Eq. (14).

$$p_{sc^j(t)} = \prod_{i=1}^{h} P(cv(t+i) = val_{j_{t+i}}) \tag{14}$$

The utility of an adaptation strategy is quantified as the weighted sum of utilities across all possible context scenarios, as detailed in Eq. (15).

$$J_E(AS(t,h), \mathbf{x}(t)) = \sum_{sc_j(t)}^{N_{sc}} p_{sc_j(t)} * J(AS(t,h), \mathbf{x}(t)) \tag{15}$$

The formulation of the stochastic control problem for proactive adaptation at time t is given by:

$$\begin{aligned} \max_{AS(t,h)} \quad & J_E(AS(t,h), \mathbf{x}(t)) \\ subject\ to \quad & \mathbf{x}(i+1) = f(\mathbf{x}(i), \mathbf{u}(i), \mathbf{cv}^j(i)) \\ & \mathbf{y}(i) = g(\mathbf{x}(i), \mathbf{cv}^j(i)) \\ & \mathbf{u}_{min} \le \mathbf{u}(i) \le \mathbf{u}_{max}, \\ & \mathbf{x}_{min} \le \mathbf{x}(i) \le \mathbf{x}_{max}, \\ & i = t, t+1, 2, \dots, t+h-1, \\ & j = 1, \dots, N_{sc}, \end{aligned} \tag{16}$$

The number of scenarios grows exponentially with respect to the prediction horizon. To keep the computational demands feasible, we use the uncertain horizon h_u. It means applying the probabilities of context variables only for the initial h_u steps while assuming constant values for these variables over the remaining $h - h_u$ duration. Consequently, this reduces the total number of contextual scenarios N_{sc}, thereby easing computational burdens.

There are many existing solvers to tackle this optimization problem, such as IBM CPLEX Optimizer, CASADI, and Optimization Toolbox in Matlab. Minimizing the objective function is usually the default setting in these solvers, so it may be necessary to take the opposite of the previously defined objective function in order to convert it to a minimization problem when solving. After the planner solves the solution $AS(t,h)$ of the above optimization problem, according to the receding horizon principle, only the first item is applied to the software system, and the subsequent items are discarded. This principle is pivotal as it compensates for inaccuracies in future state estimations, with the planner recalibrating the optimization problem based on the latest system state at subsequent intervals to derive new solutions. This iterative process also diminishes the impacts of uncertain horizons. In practical scenarios, the planner may fail to have a solution by various factors such as constrained solution time, in which case previously computed solutions may serve as fallbacks.

7. Experiments

Our research conducts extensive experiments to assess the efficacy of our proposed framework, centering on three pivotal research questions:

RQ1: Does the integration of Evidence Theory prediction fusion enhance the accuracy of context predictions and, by extension, the effectiveness of proactive adaptation strategies?

RQ2: How effective is our Stochastic Model Predictive Control (SMPC)-based planning approach in managing system adaptations?

RQ3: What is the computational overhead associated with our proposed method?

Table 2
Default experiment setting.

Name	Type	Value
failure rate	predictor	0.15
μ, σ	predictor	0.8,0.01
discount factor α	predictor	0.8
prediction horizon h	predictor	5
uncertain horizon h_u	planner	3
$x_{a,max}$	system model	5
latency l_E of u_e	system model	1
co_a	system model	2
co_f	system model	0.5
$cost_{u_a}$	utility model	1
$cost_{u_f}$	utility model	0.5
$cost_{u_e}$	utility model	0.5
$revenue_{y_{tar}}$	utility model	20
$revenue_{y_{thr}}$	utility model	−15

7.1. Experiment case: DARTSim

For the evaluation, we utilize DARTSim, with detailed descriptions in Sections 3.4 and 6.1. To enable flexible adjustments of the predictor and prediction fusion, as well as to fine-tune control parameters within the MPC framework, we developed a new simulation environment tailored for DARTSim.

Our experimental setup includes simulating an uncertain forward-looking sensor that generally provides accurate predictions but is subject to two main uncertainties: an increase in prediction error for extended ranges and a low probability risk of sensor malfunction leading to incorrect predictions. For instance, under normal conditions, the sensor's prediction ($target_{prob}$) follows a Gaussian distribution with mean μ (e.g., $\mu = 0.8$) and standard deviation σ (e.g., $\sigma = 0.01$). For longer-range predictions, adjustments are made to use a reduced μ and increased σ, whereas in malfunction scenarios, μ is adjusted to $1 - \mu$.

The latencies in the dynamics system model are as follows: (i) the control parameters u_a, u_f are linearly adjusted for altitude and formation with l_{E2} latency, i.e., $x_a(t+1) = x_a(t) + co_a * u_a(t)$, where co_a is the coefficient to reflect the latency, the range of the value of u_a and u_f are both $[0, 1]$. (ii) u_e directly changes x_e with l_E latency, i.e, $x_e(t+1) = u_e(t-l)$. u_e's legal value is $\{0, 1\}$, and we round the fractional part in the practical calculation.

The probability of target detection is computed as $y_{tar} = cv_{tar} * \frac{max(0, r_S - x_a)}{r_S}((1 - x_f) + \frac{x_f}{f_{factor}})((1 - x_e) + \frac{x_e}{e_{factor}})$, where r_S represents the maximum detection range, x_a, x_f, and x_e are the configuration values, and f_{factor} and e_{factor} are the coefficients influencing the detection probability under tight formation and active ECM conditions, respectively. This formula is also applied to calculate the probability of being detected, with adjustments to r_S, f_{factor}, and e_{factor} as necessary.

For SMPC planning, we employ the do-mpc package (Fiedler et al., 2023), which utilizes CasADi (Andersson et al., 2019) and IPOPT[1] as the optimization problem solver. The default parameters for our experiments are detailed in Table 2, providing a structured basis for evaluating the proposed framework's performance in various scenarios.

As for metrics to express the effectiveness of the adaptation methods. We evaluate the performance of the system using the metric of utility value defined in Section 6.3 ($Utility = Revenue - Principal - Interest$), which provides a comprehensive indication of the goal achievement. In addition, we specifically calculate the probability of being detected as a direct reliability metric to demonstrate for special tasks where threat detection is considered a failure.

[1] https://coin-or.github.io/Ipopt/index.html

7.2. Experiment design

RQ1:. For RQ1, our objective is to determine the efficacy of integrating historical predictions through evidence theory in improving prediction accuracy and to quantify this improvement. Given the lack of existing methods for direct benchmarking, we adopt two alternative fusion strategies: (1) directly averaging historical predictions, and (2) relying solely on the most recent predictions, a methodology frequently utilized in the relevant literature. We adjust variables such as failure probabilities and predictor parameters (μ, σ) to assess their influence on prediction accuracy. Considering the predictor outputs probabilities for each potential outcome rather than a singular predicted value, conventional metrics like precision and recall may not adequately capture the nuances of this method. Therefore, we use the accuracy score which is calculated as follows: if the actual value of the context variable cv is val_{actual}, the accuracy score is $score = P(cv = val_{actual})$.

Furthermore, we aim to investigate whether enhancements in prediction accuracy translate into more effective adaptation outcomes. To this end, we construct a simulated scenario populated with randomly generated targets and threats. Within this environment, we deploy an adaptation mechanism that leverages the most current prediction for comparative analysis. This allows us to explore the practical implications of improved prediction accuracy on the overall adaptability and responsiveness of the system in dynamic and uncertain conditions.

RQ2:. For RQ2, our evaluation of the proposed approach's effectiveness is twofold. Initially, we assess whether the overall utility and reliability that our method achieved surpasses that of the classical MPC-based planning method. To this end, we simulate various contexts and compare our approach against the CobRA approach from Angelopoulos et al. (2016). We implemented this method in the same way as in the Moreno et al. (2017) to fit our experimental conditions. In addition, we ablate our method to implement two other comparison methods: one without the latency-aware system model and another without the implementation of stochastic MPC.

Subsequently, we examine the SASO properties, namely stability, accuracy (steady-state error), settling time, and overshooting, as outlined by Hellerstein et al. (2004). They are commonly used metrics in control theory-based adaptation methods. Given the dynamic nature of self-adaptive systems, which continually adjust in response to evolving contexts, calculating SASO metrics directly may not be feasible. Therefore, we design a specific experiment where context alterations occur at a distinct point in time, allowing us to test the system's adaptive capabilities both at initial setup (Situation 1) and upon context shifts (Situation 2). The SASO metrics are defined as follows Moreno et al. (2017):

- *Stability* indicates whether the system output stabilizes to a constant value over time, representing a binary evaluation of convergence.
- *Accuracy* pertains to how closely the system's performance indicators match the predefined setpoints upon stabilization. In the context of DARTSim, the accuracy for target detection is measured by the deviation from the ideal detection probability, calculated as $| \ 1 - y_{tar} \ |$, with the setpoint of y_{tar} being 1. Conversely, for the probability of threat detection, accuracy is measured by $| \ 0 - y_{thr} \ |$, aligning with the goal of minimizing threat detection.
- *Settling time* denotes the duration required for the system to achieve a stable state.
- *Overshoot* is linked to performance indicators and quantifies the peak deviation during the adjustment phase, employing the same error metrics as defined for Accuracy.

RQ3:. For RQ3, our objective is to evaluate the runtime applicability of our approach, focusing on its computational performance. We also investigate how the prediction horizon (h), the uncertainty horizon (h_u), and the number of possible values for the context variable impact the computational time required for execution.

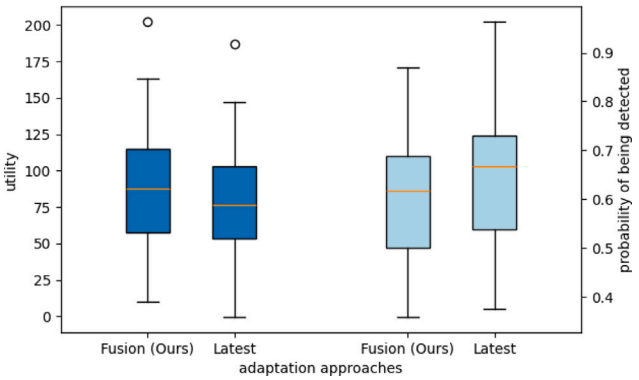

Fig. 3. Results of adaptation using different prediction methods.

7.3. Results

RQ1:. Our analysis for RQ1 regarding prediction accuracy, presented in Tables 3, 4, and 5, employs default settings with a 0.15 failure probability, predictor parameters $\mu = 0.8, \sigma = 0.01$, and a prediction horizon $h = 5$. For this evaluation, we generated random contexts over 5000 time steps for each scenario, documenting the mean and standard deviation of the prediction accuracy scores. The *Fusion* row highlights the performance of our method utilizing evidence theory for prediction fusion.

Table 3 illustrates a decline in prediction accuracy scores across all methods as the forecast period lengthens, attributable to the intrinsic decrease in predictor reliability over longer future intervals and the diminished availability of historical predictions for the *Average* and *Fusion* approaches. Notably, by the 5th time step ahead, a convergence in accuracy scores among all methods is observed, as a result of the scarcity of historical data for prediction fusion. This observation emphasizes the value of integrating extensive historical data in evidence theory-based prediction fusion to improve accuracy. Consequently, the data in Tables 4 and 5 are derived from one-step-ahead predictions to ensure sufficient historical information is available. Moreover, given the MPC's iterative recalculations at each subsequent time step to counteract future prediction inaccuracies, one-step-ahead forecasts are particularly crucial.

Table 4 reveals our method's superior performance in maintaining high prediction accuracy scores across different failure probabilities. Although accuracy diminishes as failure rates increase for all methods, our approach sustains higher accuracy levels. The averaging method, which directly incorporates erroneous predictions, yields the lowest accuracy. Sole reliance on the most recent prediction performs better than averaging but suffers from higher variability in accuracy, particularly when the latest prediction is incorrect.

Table 5 demonstrates that predictor configurations with higher μ and lower σ values achieve better accuracy, as expected. Our method particularly improves accuracy with predictors of lower inherent accuracy, showcasing its effectiveness in enhancing prediction reliability under various conditions.

To compare the utility values derived from the two approaches, namely prediction fusion and the latest predicted values, we generated 50 sets of contexts randomly. Each set spans a duration of 20 time steps. The results of this comparison are illustrated in Fig. 3. We conduct the Wilcoxon signed-rank test on the results U_{Fusion} and U_{Latest}, which is a non-parametric statistical hypothesis test for paired non-normally distributed data. The null hypothesis H_0 is that the two paired utility data come from the same distribution, $U_{Fusion} - U_{Latest}$ is symmetric about zero. One-sided alternative hypothesis H_1 is that they come from the different distributions and $U_{Fusion} - U_{Latest}$ is stochastically greater than a distribution symmetric about zero. The statistic of the Wilcoxon

Table 3
Prediction accuracy score for different time steps ahead.

Method	Timestep 1		Timestep 2		Timestep 3		Timestep 4		Timestep 5	
	avg	std	avg	std	avg	std	avg	std	avg	std
Average	0.662	**0.074**	0.654	**0.079**	0.647	**0.088**	0.641	**0.103**	0.633	0.141
Latest	0.693	0.194	0.677	0.181	0.660	0.166	0.649	0.150	0.633	0.141
Fusion	**0.881**	0.139	**0.828**	0.149	**0.762**	0.148	**0.687**	0.138	0.633	0.141

Table 4
Prediction accuracy score of different failure rate.

Method	Failure Rate 0.05		Failure Rate 0.1		Failure Rate 0.15		Failure Rate 0.2		Failure Rate 0.3	
	avg	std	avg	std	avg	std	avg	std	avg	std
Average	0.706	**0.048**	0.683	**0.065**	0.662	**0.074**	0.639	**0.085**	0.638	**0.083**
Latest	0.747	0.121	0.715	0.172	0.693	0.194	0.668	0.219	0.666	0.220
Fusion	**0.946**	0.062	**0.912**	0.108	**0.881**	0.139	**0.837**	0.182	**0.836**	0.178

Table 5
Prediction accuracy score of different predictor parameters μ, σ.

Method	$\mu, \sigma = (0.8, 0.05)$		$\mu, \sigma = (0.8, 0.01)$		$\mu, \sigma = (0.9, 0.05)$		$\mu, \sigma = (0.9, 0.01)$	
	avg	std	avg	std	avg	std	avg	std
Average	0.659	**0.099**	0.662	**0.074**	0.720	**0.119**	0.724	**0.103**
Latest	0.696	0.204	0.693	0.194	0.763	0.267	0.761	0.266
Fusion	**0.859**	0.185	**0.881**	0.139	**0.908**	0.172	**0.921**	0.145

signed-rank test is 1240 with a p-value of 3.82e−12. Hence, we would reject the null hypothesis at a confidence level of 0.05, concluding that there is an improvement in the utility. We did the same test on the probability of being detected results, which again showed an improvement. This demonstrates the ability of prediction fusion to help mitigate prediction uncertainty and thus improve the effectiveness of adaptation.

RQ2:. Furthermore, to evaluate the general effectiveness of our approach in contrast to the classical MPC-based method, we again randomly generated 50 sets of contexts with a duration of 20 time steps. The results obtained from these simulations are summarized in Table 6. The table outlines the utility values and probability of being detected associated with our method, the approach excluding stochastic MPC, the method omitting the latency-aware system model, and the classical MPC method devoid of both enhancements. This structured analysis aims to elucidate the respective contributions of stochastic modeling and latency awareness to the overall performance and reliability, thereby validating the efficacy of our method under a broad spectrum of operational conditions.

Consistent with our prior experimental procedures, we conducted the Wilcoxon signed-rank test to compare the three methodologies. Employing identical null (H_0) and alternative hypotheses (H_1), the tests conclusively rejected the null hypothesis at a 0.05 confidence level, with p-values of 1.81e−12, 8.88e−16, and 2.93e−14, respectively. This statistically significant outcome underscores the substantial improvement in utility values achieved through our approach.

Analysis of the experimental data reveals that the median utility value is notably higher when employing the planning method with the latency-aware system model. And regarding reliability, our method also achieves the lowest probability of being detected. This improvement is attributed to the model's capacity to more accurately predict future system states, facilitating more effective control decisions. Since in the default experimental setup we assign a higher reward for detecting a target compared to the penalty for being detected, the system's behavior tends to be more adventurous in detecting more targets. Furthermore, the distribution of utility values from our method exhibited the lowest standard deviation, with the least favorable utility values considerably surpassing those derived from alternative methods. This superior performance is not only due to the latency-aware system model but also to the stochastic MPC's comprehensive scenario consideration, ensuring that planning decisions remain robust against uncertainties, including erroneous predictions.

To further investigate the robustness of our findings against experimental conditions, we varied the utility model's parameters—specifically, the rewards for target detection and penalties for threat detection—and compared the effects of our method against the classical MPC approach. This examination was conducted within the same experimental framework of 50 randomly generated context sets. As illustrated in Fig. 4, we assess the average differential in utility values between our method and the classical MPC approach. The utility metrics remained positive across all configurations, with the discrepancy expanding in scenarios where the rewards/penalties were more substantial. And in the Table 7, we show the results for the probability of being detected under different utility settings. The values in the header row $(5, 30), \ldots, (30, 5)$ represent target revenue and threat penalty respectively. Again, our approach achieves better results, i.e., a lower probability of being detected. Moreover, we can see that as the penalty increases, the probability of being detected decreases, suggesting that the value of the utility model can be adjusted to allow the system to choose more conservative behaviors to improve the success rate in the specific task. These results affirm that variations in the experimental setup do not detract from our initial conclusions, reinforcing the efficacy of our approach across a diverse range of utility settings.

As for the second part, Fig. 5 delineates the experimental outcomes in a specific context, comparing our planning method (denoted as "1" with star notation) against the classical MPC-based planning method (denoted as "2" with triangle notation). The scenario assumes that the first ten regions are conditioned with $target = 0, threat = 1$, and transitions to $target = 1, threat = 0$ in the 11th region and remains until the end. The SASO properties utilizing our method are detailed in Table 8. Through the course of the experiment, it is observed that the SMPC-based planning adeptly aligns the system's performance indicators with the predefined setpoints. Nonetheless, it is imperative to recognize that MPC problems commonly encompass multiple outputs, each with distinct reference setpoints, necessitating a balanced trade-off in the overall objective function that may not achieve optimization across all performance indicators.

A notable observation was that the settling time metric under Situation 1 did not exhibit optimal performance, attributable to the prediction inaccuracies. Within the stochastic MPC computation, the

Table 6
Statistics on results of 4 approaches.

Method	Utility					Probability	
	Avg	Std	Max	Median	Min	Avg	Std
Ours	**86.06**	**41.22**	**211.40**	**88.86**	**−0.18**	**0.63**	**0.11**
Without SMPC	64.94	47.35	206.91	75.19	−48.98	0.81	0.15
Without latency-aware	60.97	45.18	190.05	64.44	−38.88	0.76	0.14
Classical MPC	56.09	45.48	180.89	57.84	−28.36	0.79	0.12

Table 7
Statistics on probability of being detected under different utility settings.

Method	(5,30)		(10,25)		(15,20)		(20,15)		(25,10)		(30,5)	
	avg	std	avg	std	avg	std	avg	std	avg	std	avg	std
Ours	**0.04**	**0.03**	**0.10**	**0.06**	**0.20**	**0.10**	**0.63**	**0.11**	**0.75**	0.10	**0.83**	**0.07**
Classical MPC	0.08	0.05	0.17	0.09	0.28	0.13	0.79	0.12	0.87	**0.09**	0.93	**0.07**

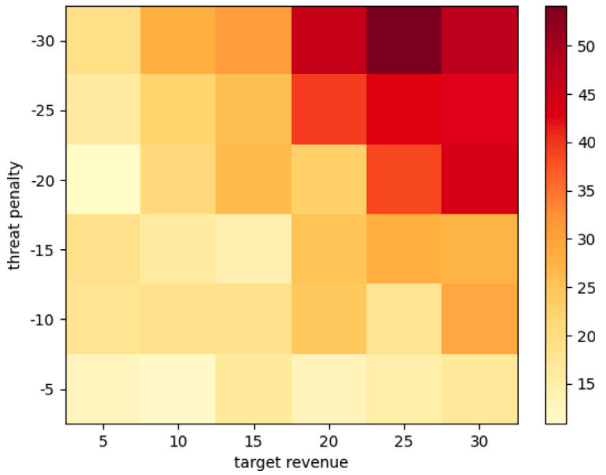

Fig. 4. Utility difference under different utility settings.

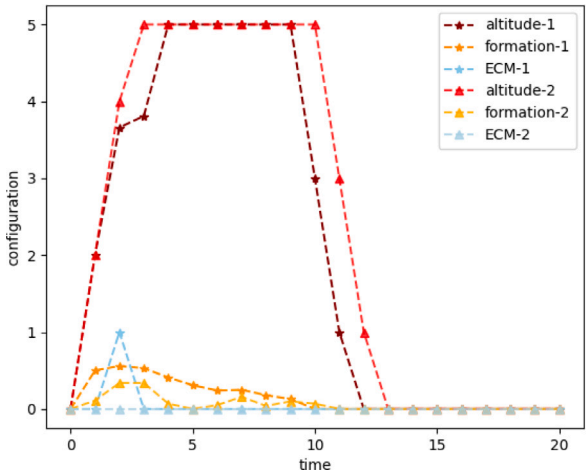

Fig. 5. Running detail of the specific context.

Table 8
SASO properties for DARTSim.

Metric	Situation 1	Situation 2
Stability	True	True
Accuracy (target)	0	0
Accuracy (threat)	0	0
Settling time	4 steps	3 steps
Overshoot (target)	0	0.4
Overshoot (threat)	0.53	0

inclusion of various potential outcomes from the predictions necessitates a cautious adaptation approach. For instance, even when the context remains at $target = 0$, the SMPC still considers the possibility of $target = 1$. This cautious planning strategy, while potentially conservative, ensures that adaptations in the face of prediction errors do not severely compromise the system, thereby enhancing reliability.

A closer examination reveals a distinct operational difference between the two methods in Situation 2, where our approach begins to decrease altitude a step earlier than the classical MPC method. This preemptive action is informed by the experimental setup, where the advantages of target detection surpass the risks of threat exposure, and an early descent yields greater overall utility. The incorporation of a latency-aware system model enables our method to anticipate and adapt prior to target arrival, in contrast to the classical MPC approach, which, lacking latency consideration, adjusts with a delay. This case exemplifies the timeliness and proactive nature of our adaptation strategy, demonstrating its superiority in scenarios requiring anticipatory adjustments.

RQ3:. In our experimentation, the time for prediction fusion via evidence theory proved to be minimal. Even with an expanded prediction horizon of 10, the process consumed merely about 0.01 s. However, the computational overhead escalates substantially with an increase in the potential values for a context variable, a challenge recognized within the domain of evidence theory research, as discussed by Voorbraak (1989).

We also analyzed the planning time in the experiments for RQ1 and RQ2, finding it to average around 400 ms with peaks up to 500ms—a duration deemed acceptable for most applications.

Given the direct correlation between planning time, prediction horizon, and uncertainty horizon, we adjusted these parameters to evaluate their impact on planning duration, as illustrated in Fig. 6. It was observed that planning time exhibits a roughly linear increase with the extension of the prediction horizon. This increment correlates with the linear growth of the solution $AS(t, h)$ as the prediction horizon h extends. Conversely, the planning time escalates more drastically with an enlarged uncertainty horizon due to the exponential rise in possible context scenarios. With the uncertainty horizon reaching 5, planning durations exceeded 30 s, a threshold considered impractical for most scenarios, hence its exclusion from the figure.

Further adjustments involved varying the number of possible values for the context variable, with the planning times documented in Fig. 7. The data indicate that planning durations extend with an increase in the context variable's potential values, particularly when the uncertainty horizon is broader. This trend is attributed to the polynomial relationship between the number of context scenarios and both the context variable's potential values and the uncertainty horizon.

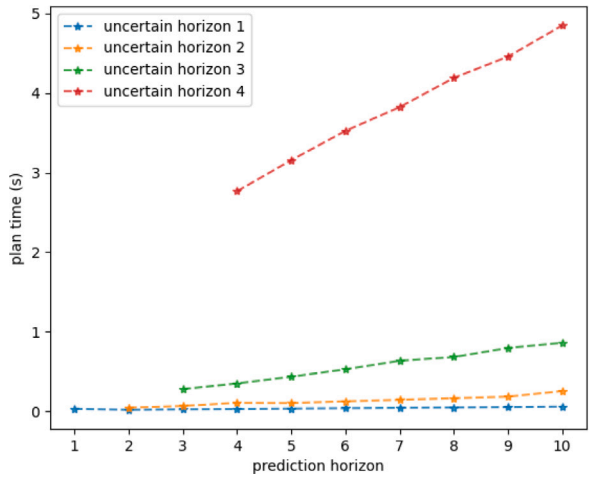

Fig. 6. Planning time under different prediction horizon and uncertain horizon.

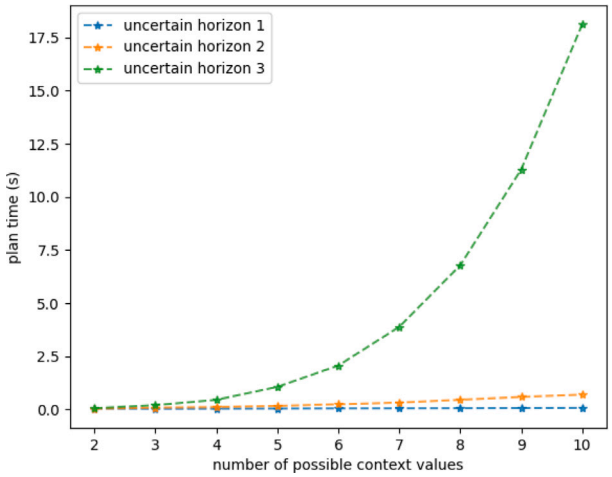

Fig. 7. Planning time under different number of possible context values.

Thus, in complex contextual scenarios, it becomes essential to curtail the uncertainty horizon to ensure planning times remain within acceptable bounds.

It is important to note that planning duration is also influenced by other factors, including the intrinsic properties of the problem and the capabilities of the solver, aspects that fall beyond the scope of this paper.

7.4. Evaluation summary and threats to validity

Our experimental findings underscore the effectiveness of our method in facilitating a reliable proactive adaptation strategy. Leveraging evidence theory to amalgamate historical predictions significantly improves prediction accuracy, thereby enhancing the adaptation utility. Furthermore, the deployment of an SMPC-based planning approach, enhanced by a latency-aware system model and a utility model grounded in the technical debt metaphor, enables the identification of more effective adaptation strategies in comparison to conventional MPC approaches.

Internal Validity: The primary challenge to internal validity arises from the influence of parameter configurations on the outcomes of our experiments. In addressing RQ1, we varied predictor parameters, such as failure probabilities, and for RQ2, we adjusted the parameters of the utility model, such as revenue figures. We conducted experiments under varied configurations to confirm the robustness of our approach. Additionally, to counteract the potential bias from specific contextual scenarios, contextual data were randomly generated for utility comparisons, aiming to neutralize the experimental bias towards particular scenarios.

External Validity: External validity revolves around the applicability and generalizability of our results. Our use of the DARTSim case study, a simplified model of a Cyber–Physical System (CPS), presents a scenario with system models that are well-documented in literature. Nonetheless, extending our approach to a broader range of software systems might require domain-specific knowledge or the application of system identification techniques to obtain accurate system models.

Construct Validity: Construct validity concerns whether the experiment's metrics accurately reflect the theoretical constructs they are intended to measure. In addressing RQ1, we devised specific accuracy scores tailored to our methodology rather than relying on generic metrics such as accuracy and recall. This decision was driven by the need to assess the performance of methods when predictions are probabilistic. For RQ2, the evaluation focused on utility values and the probability of being detected, aiming to establish the efficacy of our approach through the analysis of average or median utility values. Additionally, the reliability of the adaptation strategy was assessed based on the worst-case utility value.

8. Conclusion

In this study, we introduce a novel framework for proactive self-adaptation, aiming to derive reliable strategies under uncertain contexts. Our methodology integrates a prediction fusion mechanism utilizing evidence theory with a planning approach based on extended SMPC. Through prediction fusion, we improve context prediction accuracy and bolster the reliability of responding to erroneous predictions by incorporating uncertainty into decision-making via SMPC. Furthermore, we extend SMPC by integrating a latency-aware system model and a utility model grounded in technical debt. This expansion facilitates enhanced problem modeling and further refines the effectiveness of adaptation strategies. Through empirical analysis using the DARTSim scenario, we demonstrate the efficacy of our framework in augmenting the reliability of software systems.

A notable limitation of our approach is the prediction fusion method's design, which is tailored for discrete context variables, requiring the discretization of continuous variables for application. Moreover, the deployment of model predictive approaches necessitates the availability of an accurate system model, achievable through either direct system modeling or system identification techniques, a prerequisite that may not always be feasible. This is one of the main barriers to applying control theory to practical software systems. Additionally, the stochastic MPC faces challenges with computational complexity, especially in scenarios characterized by extensive value spaces or intricate contexts, leading to prolonged planning duration. However, several optimization strategies can be employed to adapt our method to more complex application scenarios. For complex environments, we can reduce the planning horizon or prune context combinations with a relatively lower occurrence probability. Moreover, utilizing more advanced solvers or algorithms can significantly reduce planning time. For example, the HSL MA27 solver will have a significant speed boost compared to the MUMPS solver that our method currently uses.

Moving forward, our research will delve into advanced evidence theory techniques and diverse prediction fusion techniques to refine the precision and versatility of our prediction fusion approach. Furthermore, we aim to explore alternative methods for representing the utility of adaptation strategies. In certain applications, users may exhibit varying preferences regarding risk and reliability. Consequently, alternative metrics for evaluating adaptation strategies may better align with user needs. For instance, optimizing the worst-case scenario using the Min–max form could be more effective in scenarios with lower risk acceptance. Finally, we plan to expand our experimental validations across a wider array of scenarios and use cases, thereby enhancing the generalizability and impact of our contributions to the field of proactive self-adaptation.

CRediT authorship contribution statement

Zhengyin Chen: Writing – original draft, Software, Methodology, Conceptualization. **Jialong Li:** Writing – review & editing, Investigation. **Nianyu Li:** Writing – review & editing, Supervision, Investigation. **Wenpin Jiao:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

https://github.com/easton-chen/DART-SMPC.git.

Acknowledgments

The research was supported by National Key R&D Program of China, Granted No. 2023YFC3502900, and Natural Science Foundation of China, Granted No. 62192730 and No. 62192731.

References

Ali, R., Dalpiaz, F., Giorgini, P., 2010. A goal-based framework for contextual requirements modeling and analysis. Requir. Eng. 15 (4), 439–458. http://dx.doi.org/10.1007/s00766-010-0110-z.

Andersson, J.A.E., Gillis, J., Horn, G., Rawlings, J.B., Diehl, M., 2019. CasADi – a software framework for nonlinear optimization and optimal control. Math. Program. Comput. 11 (1), 1–36. http://dx.doi.org/10.1007/s12532-018-0139-4.

Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M., 2007. Web services agreement specification (WS-agreement). In: Open Grid Forum. 128, (1), Citeseer, p. 216.

Angelopoulos, K., Papadopoulos, A.V., Silva Souza, V.E., Mylopoulos, J., 2016. Model predictive control for software systems with CobRA. In: Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. ACM, Austin Texas, pp. 35–46. http://dx.doi.org/10.1145/2897053.2897054.

Angelopoulos, K., Papadopoulos, A.V., Souza, V.E.S., Mylopoulos, J., 2018. Engineering self-adaptive software systems: From requirements to model predictive control. ACM Trans. Auton. Adapt. Syst. 13 (1), 1–27. http://dx.doi.org/10.1145/3105748.

Ayala, I., Papadopoulos, A.V., Amor, M., Fuentes, L., 2021. ProDSPL: Proactive self-adaptation based on dynamic software product lines. J. Syst. Softw. 175, 110909. http://dx.doi.org/10.1016/j.jss.2021.110909.

Cámara, J., Moreno, G.A., Garlan, D., 2014. Stochastic game analysis and latency awareness for proactive self-adaptation. In: Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. ACM, Hyderabad India, pp. 155–164. http://dx.doi.org/10.1145/2593929.2593933.

Chen, T., Bahsoon, R., Wang, S., Yao, X., 2018. To adapt or not to adapt?: technical debt and learning driven self-adaptation for managing runtime performance. In: Proceedings of the 2018 ACM/SPEC International Conference on Performance Engineering. ACM, Berlin Germany, pp. 48–55. http://dx.doi.org/10.1145/3184407.3184413.

Cooray, D., Kouroshfar, E., Malek, S., Roshandel, R., 2013. Proactive self-adaptation for improving the reliability of mission-critical, embedded, and mobile software. IEEE Trans. Softw. Eng. 39 (12), 1714–1735. http://dx.doi.org/10.1109/TSE.2013.36.

Cunningham, W., 1993. The WyCash portfolio management system. ACM Sigplan Oops Messenger 4 (2), 29–30. http://dx.doi.org/10.1145/157710.157715.

Dempster, A.P., 1967. Upper and lower probabilities induced by a multivalued mapping. Ann. Math. Stat. 38 (2), 325–339. http://dx.doi.org/10.1214/aoms/1177698950.

Fan, J., Tong, Y., Qin, Y., Ma, X., 2020. Overwhelming uncertainty in self-adaptation: an empirical study on PLA and CobRA. In: 12th Asia-Pacific Symposium on Internetware. ACM, Singapore Singapore, pp. 250–259. http://dx.doi.org/10.1145/3457913.3457943.

Fiedler, F., Karg, B., Lüken, L., Brandner, D., Heinlein, M., Brabender, F., Lucia, S., 2023. Do-mpc: Towards FAIR nonlinear and robust model predictive control. Control Eng. Pract. 140, 105676. http://dx.doi.org/10.1016/j.conengprac.2023.105676.

Gerostathopoulos, I., Raibulet, C., Alberts, E., 2022. Assessing self-adaptation strategies using cost-benefit analysis. In: 2022 IEEE 19th International Conference on Software Architecture Companion. ICSA-C, pp. 92–95. http://dx.doi.org/10.1109/ICSA-C54293.2022.00023.

Hellerstein, J.L., Diao, Y., Parekh, S.S., Tilbury, D.M., 2004. Feedback Control of Computing Systems. Wiley, http://dx.doi.org/10.1002/047166880X, URL.

Incerto, E., Tribastone, M., Trubiani, C., 2017. Software performance self-adaptation through efficient model predictive control. In: 2017 32nd IEEE/ACM International Conference on Automated Software Engineering. ASE, IEEE, pp. 485–496.

Keller, C., Mann, Z.Á., 2020. Towards understanding adaptation latency in self-adaptive systems. In: Yangui, S., Bouguettaya, A., Xue, X., Faci, N., Gaaloul, W., Yu, Q., Zhou, Z., Hernandez, N., Nakagawa, E.Y. (Eds.), Service-Oriented Computing – ICSOC 2019 Workshops. vol. 12019, Springer International Publishing, Cham, pp. 42–53. http://dx.doi.org/10.1007/978-3-030-45989-5_4.

Kephart, J., Chess, D., 2003. The vision of autonomic computing. Computer 36 (1), 41–50. http://dx.doi.org/10.1109/MC.2003.1160055.

Kouvaritakis, B., Cannon, M., 2016. Model predictive control, vol. 38, Springer International Publishing, Switzerland.

Kumar, S., Bahsoon, R., Chen, T., Buyya, R., 2019. Identifying and estimating technical debt for service composition in SaaS Cloud. In: 2019 IEEE International Conference on Web Services. ICWS, pp. 121–125. http://dx.doi.org/10.1109/ICWS.2019.00030.

Kusic, D., Kephart, J.O., Hanson, J.E., Kandasamy, N., Jiang, G., 2009. Power and performance management of virtualized computing environments via lookahead control. Cluster Comput. 12 (1), 1–15. http://dx.doi.org/10.1007/s10586-008-0070-y.

Lemos, R.d., Giese, H., Müller, H.A., Shaw, M., Andersson, J., Litoiu, M., Schmerl, B., Tamura, G., Villegas, N.M., Vogel, T., et al., 2013. Software engineering for self-adaptive systems: A second research roadmap. In: Software Engineering for Self-Adaptive Systems II. Springer, pp. 1–32.

Li, N., Cámara, J., Garlan, D., Schmerl, B., Jin, Z., 2021. Hey! preparing humans to do tasks in self-adaptive systems. In: 2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems. SEAMS, pp. 48–58. http://dx.doi.org/10.1109/SEAMS51251.2021.00017.

Ljung, L., 1998. System Identification: Theory for the User. Pearson Education.

Maggio, M., Papadopoulos, A.V., Filieri, A., Hoffmann, H., 2017. Automated control of multiple software goals using multiple actuators. In: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering. ACM, Paderborn Germany, pp. 373–384. http://dx.doi.org/10.1145/3106237.3106247.

Metzger, A., Neubauer, A., Bohn, P., Pohl, K., 2019. Proactive process adaptation using deep learning ensembles. In: Giorgini, P., Weber, B. (Eds.), In: Advanced Information Systems Engineering, vol. 11483, Springer International Publishing, Cham, pp. 547–562. http://dx.doi.org/10.1007/978-3-030-21290-2_34.

Moreno, G.A., Cámara, J., Garlan, D., Schmerl, B., 2015. Proactive self-adaptation under uncertainty: A probabilistic model checking approach. In: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering. ACM, Bergamo Italy, pp. 1–12. http://dx.doi.org/10.1145/2786805.2786853.

Moreno, G.A., Cámara, J., Garlan, D., Schmerl, B., 2018. Flexible and efficient decision-making for proactive latency-aware self-adaptation. ACM Trans. Auton. Adapt. Syst. 13 (1), 1–36. http://dx.doi.org/10.1145/3149180.

Moreno, G., Kinneer, C., Pandey, A., Garlan, D., 2019. DARTSim: An exemplar for evaluation and comparison of self-adaptation approaches for smart cyber-physical systems. In: 2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. SEAMS, IEEE, Montreal, QC, Canada, pp. 181–187. http://dx.doi.org/10.1109/SEAMS.2019.00031.

Moreno, G.A., Papadopoulos, A.V., Angelopoulos, K., Camara, J., Schmerl, B., 2017. Comparing model-based predictive approaches to self-adaptation: CobRA and PLA. In: 2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. SEAMS, IEEE, Buenos Aires, Argentina, pp. 42–53. http://dx.doi.org/10.1109/SEAMS.2017.2.

Neufelder, A.M., et al., 2016. IEEE recommended practice on software reliability. IEEE Standard 1633–2016.

Palmerino, J., Yu, Q., Desell, T., Krutz, D., 2019. Improving the decision-making process of self-adaptive systems by accounting for tactic volatility. In: 2019 34th IEEE/ACM International Conference on Automated Software Engineering. ASE, pp. 949–961. http://dx.doi.org/10.1109/ASE.2019.00092.

Salehie, M., Tahvildari, L., 2009. Self-adaptive software: Landscape and research challenges. ACM Trans. Autonomous Adapt. Syst. (taas) 4 (2), 1–42.

Shafer, G., 1976. A Mathematical Theory of Evidence. Princeton University Press.

Shin, Y.-J., Cho, E., Bae, D.-H., 2021. PASTA: An efficient proactive adaptation approach based on statistical model checking for self-adaptive systems. In: Guerra, E., Stoelinga, M. (Eds.), In: Fundamental Approaches To Software Engineering, vol. 12649, Springer International Publishing, Cham, pp. 292–312. http://dx.doi.org/10.1007/978-3-030-71500-7_15.

Tanabe, M., Tei, K., Fukazawa, Y., Honiden, S., 2017. Learning environment model at runtime for self-adaptive systems. In: Proceedings of the Symposium on Applied Computing. pp. 1198–1204.

Voorbraak, F., 1989. A computationally efficient approximation of Dempster-Shafer theory. Int. J. Man-Mach. Stud. 30 (5), 525–536. http://dx.doi.org/10.1016/S0020-7373(89)80032-X.

Wang, C., Pazat, J.-L., 2012. A two-phase online prediction approach for accurate and timely adaptation decision. In: 2012 IEEE Ninth International Conference on Services Computing. IEEE, Honolulu, HI, USA, pp. 218–225. http://dx.doi.org/10.1109/SCC.2012.26.

Wang, H., Wang, L., Yu, Q., Zheng, Z., Yang, Z., 2018. A proactive approach based on online reliability prediction for adaptation of service-oriented systems. J. Parallel Distrib. Comput. 114, 70–84. http://dx.doi.org/10.1016/j.jpdc.2017.12.006.

Wang, L., Xu, J., Duran-Limon, H.A., Zhao, M., 2015. Qos-driven cloud resource management through fuzzy model predictive control. In: 2015 IEEE International Conference on Autonomic Computing. IEEE, pp. 81–90.

Weyns, D., 2020. An Introduction to Self-adaptive Systems: A Contemporary Software Engineering Perspective. John Wiley & Sons.

Zhang, J., Ma, M., Wang, P., 2020. Lead time-aware proactive adaptation for service-oriented systems. In: 2020 IEEE International Conference on Web Services. ICWS, IEEE, Beijing, China, pp. 481–488. http://dx.doi.org/10.1109/ICWS49710.2020.00071.

Zhengyin Chen received the B.Sc. degree in computer science and technology from Peking University. He is currently pursuing the Ph.D. degree in computer software and theory under the supervision of Prof. Wenpin Jiao at the School of Computer Science, Peking University. His research interests include self-adaptive systems, proactive software systems, control systems, and context prediction.

Jialong Li is now a Research Associate at Waseda University in Tokyo, Japan. He received B.E and M.E degrees in Computer Science from Waseda University in 2019 and 2021, respectively. His research interests are in self-adaptive systems, software engineering, and human–computer interaction. He is a member of the IEEE Computer Society (IEEE CS), and Association for Computing Machinery (ACM).

Nianyu Li received her doctorate in Computer Software and Theory from Peking University under the guidance of Prof. Zhi Jin and Prof. Wenpin Jiao. Additionally, She had the opportunity to serve as a Visiting Research Intern at the National Institute of Informatics (NII) and Carnegie Mellon University (CMU). Her primary research area is human-involved self-adaptive systems, where she focuses on applying rigorous modeling and analysis techniques, frameworks, and control paradigms. She has a particular interest in human-in-the-loop systems, software design, requirements modeling, specification and verification, system safety, security, and cyber–physical systems.

Wenpin Jiao received the B.S. and M.S. degrees in computer science from the East China University of Science and Technology, in 1991 and 1997, respectively, and the Ph.D. degree in computer science from the Institute of Software, Chinese Academy of Sciences, in 2000. He is currently a Full Professor with the School of Computer Science, Peking University. His research interests include software engineering, multiagent systems, and intelligent software technology.