Active Monitoring Mechanism for Control-based Self-Adaptive Systems

ANONYMOUS AUTHOR(S)

Control-based self-adaptive systems (control-SAS) are susceptible to deviations from their pre-identified nominal models. If this model deviation exceeds a threshold, the optimal performance and theoretical guarantees of the control-SAS can be compromised. Existing approaches detect these deviations by locating the mismatch between the control signal of the managing system and the response output of the managed system. However, *vague observations* may mask a potential mismatch where the explicit system behavior does not reflect the implicit variation of the nominal model. In this paper, we propose the Active Monitoring Mechanism (AMM for short) as a solution to this issue. The basic intuition of AMM is to stimulate the control-SAS with an active control signal when vague observations might mask model deviations. To determine the appropriate time for triggering the active signals, AMM proposes a stochastic framework to quantify the relationship between the implicit variation of a control-SAS and its explicit observation. Based on this framework, AMM's monitor and remediator enhance model deviation detection by generating active control signals of well-designed timing and intensity. Results from empirical evaluations on three representative systems demonstrate AMM's effectiveness (33.0% shorter detection delay, 18.3% lower FN rate, 16.7% lower FP rate) and usefulness (19.3% lower abnormal rates and 88.2% higher utility).

ACM Reference Format:
Anonymous Author(s). 2024. Active Monitoring Mechanism for Control-based Self-Adaptive Systems. 1, 1 (March 2024), 24 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Control-based self-adaptive systems (control-SAS for short) enable developers to optimize the performance of their software systems based on the mathematical foundations of control theory. Conceptually, a control-SAS consists of a managed system, which is the subject software system, and a managing system, which adapts the behavior of the managed system through producing control signals. Various control mechanisms, such as proportional–integral–derivative control [30, 47, 66] and model predictive control [3, 4, 51, 52], provide both practical guidelines on how to implement and configure a managing system, whose optimality and reliability are theoretically guaranteed by control theory [42].

However, software systems behave differently from physical systems to which control mechanisms are typically applied. The behavior of the former is discrete and more dynamic than that of the latter. During the execution of a control-SAS, the controllability of the managed system may change from the *nominal model* on which the managed system is configured. This change, which can be referred to as *model deviation*, could invalidate the managing system's controllability over the managed system [9, 22, 49, 78]. Therefore, monitoring the nominal model of a control-SAS and detecting its unsafe deviation is critical to the effectiveness of the system.

ACM XXXX-XXXX/2024/3-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

To detect model deviation, most existing approaches leverage control theory-based techniques, including the sliding window approach [15, 29, 36, 66], the autoregressive moving average approach [23, 41], and the parameter estimation approach [1, 21, 43]. The basic intuition of these techniques is that model deviation could cause a mismatch between the control signal of the managing system and the response of the managed system to the signal. As a result, once the managed system is stimulated by a control signal, one could first derive a theoretical value of the system output (i.e., the output of the managed system), and then compare it to the observed system output to detect potential model deviations. Despite its simplicity, this intuition has proven effective for conventional control systems. However, the direct application of it to control-SASs would face a critical problem that *a control-SAS may generate more zero-valued control signals than its control system counterpart*, especially when the system reaches the stable equilibrium. If the managed system is stimulated by a zero-valued control signal, it will simply not respond, making existing approaches unable to detect potential model deviations.

The reason that control-SASs produce more zero-valued control signals is twofold. First, control-SASs are designed to reject disturbances [29]. The managing system should not respond to short-lived external forces and stimulate the managed system with non-zero control signals accordingly. Second, most control-SASs use discrete-valued (e.g., the operating frequency of a CPU) or even integer-valued (e.g., the number of active servers) control signals. If the managing system produces a small-value control signal, it will most likely be rounded to a zero-valued signal to be received by the managed system. Disturbance rejection and signal discretization lead to a control error of the managing system. When the system is in its transient state, this control error is compensated with non-zero control signals. However, when the system is in its stable state, a compensated non-zero signal can only be triggered by the accumulation of the control error, which can take a long time and delay the detection of the model deviations.

We call this phenomenon *vague observation* in model deviation detection. Basically, vague observation describes the situation where one cannot distinguish the normal behavior of a control-SAS from a deviated one. To the best of our knowledge, there is no satisfactory solution for detecting model deviation with vague observations. The widely used control-theoretic approaches rely heavily on observing the response of the control-SAS stimulated by non-zero control signals [10, 38, 39, 77], which makes them more vulnerable to vague observations. For those approaches that do not use control signals [13, 29, 51], their accuracy and timeliness are less impressive due to the uncertainty faced by control-SASs.

In this paper, we propose an $\underline{\text{A}}$ctive $\underline{\text{M}}$onitoring $\underline{\text{M}}$echanism (*AMM* for short) as a solution for detecting model deviation in control-SASs. The intuition behind AMM is simple but effective: *AMM generates an active control signal u_A to reveal the potential mismatch between the managing system and the managed system.* The key challenge for AMM is to balance the effectiveness and risk of its active signal u_A. Given that u_A is injected into the running managed system, triggering it too often would increase the risk for system disruption. However, a long trigger interval will reduce the effectiveness of the active signal in handling vague observations.

To address this challenge, AMM proposes a stochastic framework to systematically evaluate the confidence that a model deviation has occurred. We not only quantify *the confidence that a model deviation has occurred with non-vague observations*, but also evaluate *the confidence that a model deviation is covered by vague observations.* To achieve this, the proposed framework leverages our investigation of reported model deviations suffered by different control-SASs. The framework describes the implicit variation (i.e., the change in model parameters) of model deviations from a software engineering perspective, and links the variation to its corresponding explicit manifestations (i.e., the change in observable variables) from a control theory perspective. Based on this framework, AMM uses a four-way monitor to evaluate its confidence in different types of model deviations with both non-vague and vague observations.

We implement AMM as a supervising mechanism for control-SASs, consisting of a monitor and a remediator. The remediator takes remedial action when the confidence reported by the monitor falls below a warning level: either it generates a well-designed active signal when model deviation might occur with vague observations, or it switches to a mandatory managing system when model deviation is detected with non-vague observations.

Based on the evaluation of three representative control-SASs and four baseline approaches, AMM reports lower detection delay (on average 33.0% shorter), higher accuracy (on average 18.3% lower FN rate and 16.7% lower FP rate). AMM-based supervising mechanism can also enhance the execution of control-SASs with lower abnormal rates (on average 19.3% lower) and higher utility ratio (on average 88.2% higher).

In the remainder of this paper, Section 2 provides the necessary background on control-SASs. Section 3 motivates the work, including an empirical investigation of the model deviation problem and a motivating example. Next, Section 4 presents the proposed AMM approach. Section 5 gives the experimental evaluation of our approach. Finally, Section 6 discusses related work, and Section 7 concludes the paper.

2 PRELIMINARIES

In this section, we first present the background of control-SASs. Then, we briefly describe the assumptions of our approach and clarify their realistic.

2.1 Control-based self-adaptive systems

Control-based self-adaptive systems are inspired by control theory. It reduces the burden on the developers' mathematical and software knowledge to design specific self-adaptive solutions [29, 64, 68]. The basic assumption of control-SASs is that the behavior of the managed system can be captured by a quantitative *nominal model*. The nominal model describes the relationship between the managed system's output y (a.k.a. system output), the output of the managing system u (a.k.a. the control signal, which is also the internal input of the managed system) and the external input of the managed system α (a.k.a. the environmental input). Basically, the nominal model θ includes the parameters representing the system's controllability (denoted as θ_C), latency (denoted as θ_L), observability (denoted as θ_O), and so on.

We use a self-adaptive system for the Body Sensor Network [34] (*BSN* for short) to further explain the details of a control-SAS. The original BSN is a real-time health monitoring system based on a wireless sensor network. It consists of six sensors that collect different types of vital signs (e.g., blood oxygen, heart rate, body temperature, etc.) and a data processing hub that provides an overall health assessment. The power consumption of each sensor is determined by its sampling rate (which can be adjusted remotely), and its sensing noise (which is determined by the environmental uncertainty). In a self-adaptive BSN, each of the sensors is controlled by a proportional–integral managing system that adjusts the sensor's power consumption y to a setpoint $\bar{y}$ by changing the sampling rate with a control signal u (e.g., a positive u means increasing the sampling rate, and a negative one means decreasing the sampling rate). Figure 1 describes the design of such a managing system. The variables of the model are explained as follows (t denotes a time stamp corresponding to the self-adaptation loop of the system).

- $u(t)$ describes the value of the control signal at time t, for example the sampling rate of the sensor.

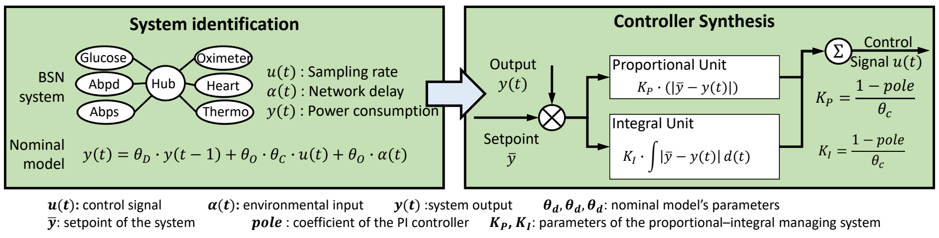
System identification
BSN system
Glucose
Abpd
Abps
Hub
Oximeter
Heart
Thermo
$u(t)$: Sampling rate
$\alpha(t)$: Network delay
$y(t)$: Power consumption
Nominal model
$y(t) = \theta_D \cdot y(t-1) + \theta_O \cdot \theta_C \cdot u(t) + \theta_O \cdot \alpha(t)$
Controller Synthesis
Output $y(t)$
Setpoint $\bar{y}$
Proportional Unit
$K_P \cdot (|\bar{y} - y(t)|)$
Integral Unit
$K_I \cdot \int |\bar{y} - y(t)|\, d(t)$
Σ
Control Signal $u(t)$
$K_P = \frac{1 - pole}{\theta_c}$
$K_I = \frac{1 - pole}{\theta_c}$
$u(t)$: control signal
$\alpha(t)$: environmental input
$y(t)$:system output
$\theta_d, \theta_d, \theta_d$: nominal model's parameters
$\bar{y}$: setpoint of the system
$pole$: coefficient of the PI controller
K_P, K_I: parameters of the proportional–integral managing system

Fig. 1. The motivating self-adaptive BSN system

- $y(t)$ describes the value of the system output at time t, for example the power consumption of the sensor.
- $\alpha(t)$ describes the value of the environmental input at time t, for example the measurement error that affects the quality of the sensed data.

In the first step (i.e., system identification), the values of the model parameters (θ_C, θ_L, and θ_O) are identified by systematically sampling the power consumption of the sensor at different sampling rates. Then, in the second step (i.e., controller synthesis), the proportional–integral-based managing system is derived by composting the proportional unit and the integral unit. The effectiveness and safety of the system depends on its coefficients (K_P and K_I), which are directly determined by the identified model parameters. As a result, once a model deviation occurs, the nominal model will deviate from its pre-identified parameter value, and the entire system will operate inefficiently and unsafely.

2.2 Our assumptions

Uncertainty distribution One of the major causes of model deviation is the uncertainty faced by control-SASs. Uncertainty consists of both measurement error, which affects one's observation of the system, and system noise, which fluctuates the internal state of the system. In this work, we assume that these two types of uncertainty are subject to a normal distribution, which is a typical assumption among researchers of self-adaptive systems [24, 27, 69, 70]. We assume that the system output y and the environmental input α are subject to normal distributions. We also assume that the model parameters (θ_C, θ_L, and θ_O) are subject to normal distributions that describe the variances of their identification values.

Linear system-based nominal model. In this paper, we will focus on linear systems. Linear systems are one of the most widely used models among both self-adaptation and control theory researchers, and have been successfully applied in different domains [3, 41, 51, 66, 67]. Linear model can also provide an approximate description to nonlinear system by using various linearization approaches, such as local linearization in phase space [32], global linear representation based on Lie series solution [11, 46], and topological indices-based global linearization [12].

3 MOTIVATION

In this section, we first describe the issue of model deviation and present our motivating examples. Next, we review the model deviations reported by existing literature.

3.1 Model deviation in Control-SASs

The model deviation problem. Given the dynamic and uncertain running environment of a control-SAS, there is an inevitable difference between the nominal model's identified parameter values θ, and its actual value $\theta(t)$ at runtime. If this difference lies within a certain safety range, the control mechanism's robustness ensures the control-SAS's efficacy. If the difference exceeds the safety range, *model deviation* occurs and undermines the system.

When a model deviation arises, the corresponding control-SAS doesn't instantly malfunction. Rather, the impact of this deviation is apparent when the managing system produces its control signals. The model deviation diminishes the effectiveness of these control signals, as well as the managing system's ability to adjust the managed system. When such inadequate adaptation cannot be compensated for by stronger control signals, the managed system's behavior would go beyond the control of the managing system, resulting in the failure of the control-SAS.

In the previous example of a self-adaptive BSN, if the controllability parameter $\theta_C(t)$ increases and the managing system decreases its sampling rate by u, then the power consumption of the system y will decrease more slowly than anticipated. Consequently, the self-adaptive BSN will continue to reduce its sampling rate to compensate for the increased value of θ_C. Once this compensation ceases (e.g., when the sampling rate reaches its lowest level), the managing system can no longer adjust the system's power consumption effectively.

Here we use three phases to describe a control-SAS's execution after model deviation:

- *Implicit phase* begins when model deviation occurs, invalidating the theoretical guarantees provided by control theory, without producing any explicit manifestation.
- *Low-controllability phase* begins when the managed system receives its first control signal following the occurrence of the deviation. During this phase, the system violates a few non-critical properties, like the optimizability of the managing system, leading to slow response time or lengthy settling time.
- *Abnormal phase* begins when the system violates crucial control properties, such as the SASO properties (i.e., stability, accuracy, settling time, and overshoot). Compared to the low-controllability phase, the behavior of the controlled system is no longer adjustable by the managing system, resulting in system saturation, oscillation, or other failures.

3.2 Detecting model deviation with vague observation

Several observation-based approaches have been proposed to address the issue of model deviation. As mentioned in Section 1, these approaches detect model deviation by identifying the mismatch between the control signal u generated by the managing system and the response output y of the managed system [15, 29, 41, 53, 69]. In other words, these approaches depend on linking the explicit manifestation of the deviation (i.e., how the system's operation is impacted) with the implicit variation (i.e., how the nominal model deviates). However, vague observations would weaken this connection, rendering these observation-based approaches ineffective.

Let us consider two exemplary executions of the self-adaptive BSN system as motivating examples.

Vague observation during a long-lasting gradual change deviation. Figure 2-(a) shows the first run. We introduce a deviation caused by changes in the sensor's battery capacity, as reported in [34].

In this example, the low-controllability phase begins when the system produces its first control signal $u(132)$ (as indicated by mismatch②). During this phase, the system's utility decreases as the deviation results in control error of the managing system, hindering its ability to adjust the system output to its setpoint (which is 0.9 in this case). As the mismatch between $u(t)$ and $y(t)$ grows (from mismatch②to mismatch④), so does the gap between the responded $y(t)$ and the setpoint, and finally results in system oscillation after mismatch⑤.

To detect this deviation, existing approaches rely on four mismatched pairs of $u(t)$ and $y(t)$. However, it would not be easy to distinguish the mismatched pairs (e.g., mismatch②to ⑤) from the

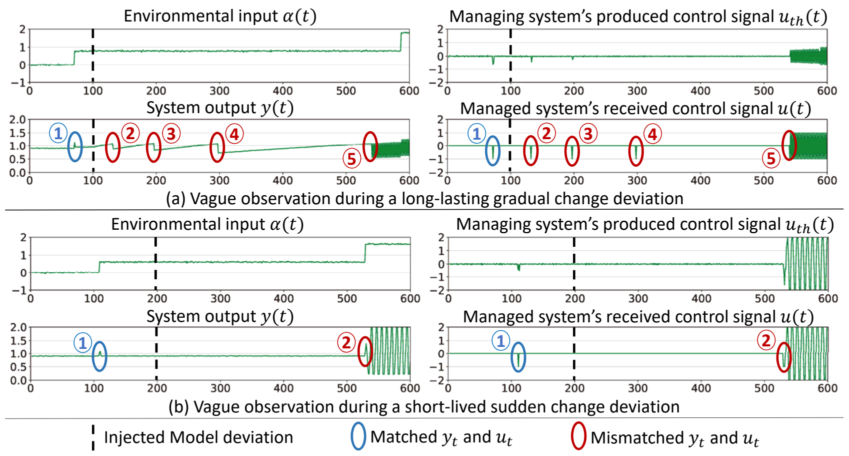

Fig. 2. Two exemplar executions of the self-adaptive BSN system suffering model deviation

matched pair (e.g., mismatch①) given the uncertainty suffered by control-SASs. Although certain mismatched pairs of $u(t)$ and $y(t)$ (such as mismatch⑤) may be detected easily, but identifying such apparent pairs would be meaningless since oscillation has already occurred.

The discrete control signal worsens the situation by rounding small-value control signals to zero. Figure 2-(a) depicts that the managing system produces non-zero control signals $u_{th}(t)$ in response to the fluctuating $y(t)$ from time 298 to 590. However, $u(t)$ received by the managed system remains zero at the same time. In other words, if a detection approach fails to identify mismatch④, then it has to wait until the last pair appears to detect the model deviation.

Vague observation during a short-lived sudden change deviation. Figure 2-(b) shows the second execution, wherein a deviation is injected simulating a network attack that causes the temporary reversal of the control signal (i.e., θ_C is changed to $-\theta_C$). Although this deviation is not directly reported by BSN researchers, but it is a common occurrence in control-SASs [22, 29, 65].

In this example, the system undergoes drastic oscillations soon after the managed system receives its first non-zero control signal $u(530)$ (as indicated by mismatch②). This means that the existing observation-based approach cannot prevent the loss of utility, nor can it prevent the system from entering an unsafe state. This rapid deterioration is caused by the intense $u(530)$, as a response to environmental fluctuations near time 530. The strength of this signal drives the system into an abnormal phase quickly, along with the severely-deviated θ_C. We note $u(t)$ remains zero between time 110 and 530, since the environmental input $\alpha(t)$ remains stable, and small-valued $u_{th}(t)$ is rounded off. If the control signal is not discretized, even a small control signal could potentially cause a reversed system output, and existing approaches can easily detect such a mismatch.

In this paper, we address the challenge of vague observation by generating active control signals to compensate for the rounded control signals. The proposed AMM approach determines the optimal moment for triggering the active signal to balance signal effectiveness and introduced disturbance. The technical design of AMM focuses on a stochastic framework that connects model deviations' implicit variation with explicit manifestations. The intuition is similar to software runtime monitoring, where the implicit variation reflects the target software defect patterns, and the explicit manifestations correspond to the observed system execution. However, consider the difference between control-SASs and conventional software systems, AMM uses the state-space model from a control theory perspective to formally describe the interaction between the variation and the manifestations.

3.3 Review of reported model deviations

To better design AMM's stochastic framework, we briefly review some recent literature on the safety and reliability of control-SASs. Our findings indicate that seven commonly used subject systems from the community have been reported to suffer from model deviations. We present the results of our investigation in Table 1. Noted that the types of model deviation reported by different researchers may vary for a given subject system. The table also shows the explicit manifestation (i.e., how the behavior of the system is affected) and implicit variation (i.e., how the nominal model is deviated) that we concluded from the literature.

Table 1. The investigated model deivations

System	Reported deviation	Explicit manifestation	Implicit variation
Encoder	System's garbage collector activating uncontrollably [31]	Short-lived sharp system oscillation	Unexpected change of controllability
	Encoding videos of different types with the one used for system identification [13]	Long-lasting great loss of system utility	Unexpected change of controllability
RUBiS	Delay of the deployment of new servers [14]	Long-lasting small loss of system utility	System noise in control delays
	Unexpected system hardware defects [47]	Long-lasting sharp system saturation	Unexpected change of controllability or observability
	Rapid growth of the service request [54]	Short-lived sharp system oscillation	Unexpected change of controllability
SWaT	Network attack on the smart valves [22]	Long-lasting great loss of system utility	Unexpected change of controllability
BSN	degradation of the sensor's battery performance [34]	Long-lasting slow decline of the system utility	System noise in controllability
ZNN	Change of network condition [16]	Long-lasting great loss of system utility	System noise in controllability and delay
TAS	Change of the service response time due to network condition fluctuations [73]	Long-lasting small loss of system utility	System noise in control delay
	Uncontrolled network jamming [65]	Long-lasting great loss of system utility	Unexpected change in controllability and dely
SDB	Infrastructure communication defects [29]	Long-lasting sharp system saturation	Unexpected change in observability
	Third-party API dependency exceptions [30]	Short-lived sharp system oscillation	Unexpected change in controllability

The explicit manifestation. We describe the explicit manifestation of the model deviation in terms of two aspects, its duration and its severity. *Duration* refers to the elapsed time period during which the model deviation persists, as some nominal models can recover from certain types of deviations. *Severity* refers to the rate of progression of the model deviation, e.g., how long it might take for a deviation to progress from the implicit phase to the abnormal phase. Note that the listed

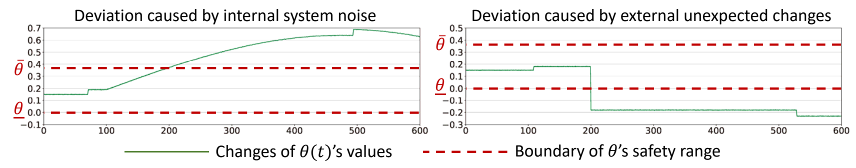

Fig. 3. Two types of implicit variations

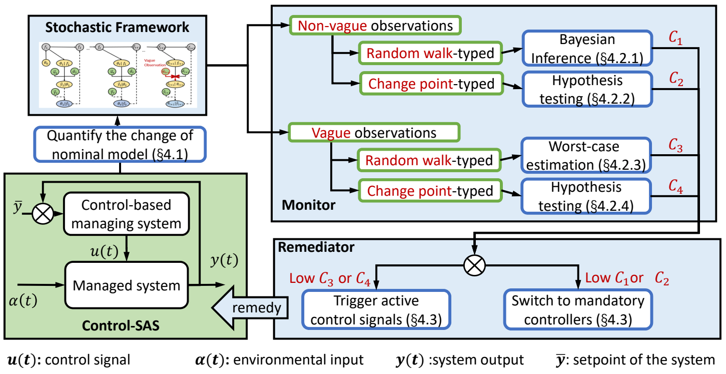
Stochastic Framework
Quantify the change of nominal model (§4.1)
Non-vague observations
Random walk-typed
Change point-typed
Bayesian Inference (§4.2.1)
Hypothesis testing (§4.2.2)
Vague observations
Random walk-typed
Change point-typed
Worst-case estimation (§4.2.3)
Hypothesis testing (§4.2.4)
C_1
C_2
C_3
C_4
Monitor
Remediator
Low C_3 or C_4
Low C_1or C_2
Trigger active control signals (§4.3)
Switch to mandatory controllers (§4.3)
$\bar{y}$
Control-based managing system
$u(t)$
$y(t)$
Managed system
$\alpha(t)$
remedy
Control-SAS
$\boldsymbol{u(t)}$: control signal
$\boldsymbol{\alpha(t)}$: environmental input
$\boldsymbol{y(t)}$:system output
$\bar{\boldsymbol{y}}$: setpoint of the system

Fig. 4. The proposed AMM approach

explicit manifestation is typically observed during the abnormal phase and should therefore be combined with the implicit variation to guide the detection of model deviations.

The implicit variation. The model deviation's implicit variation provides insight into the patterns of deviations within subject systems. This paper identifies two distinct varieties of implicit variations, namely *external unexpected changes* and *internal system noise*, which are demonstrated in Figure 3. External unexpected changes arise from the environment or domain in which the control-SAS is deployed. Due to their potential to surpass the system's maximum capacity, unexpected changes often trigger a dramatic response in the system, including a considerable loss of system utility and severe oscillations. In contrast, internal system noise arises from the difficulty of determining the effect of the control signal on the behavior of the managed system. Since system noise can only hinder the system's ability to adapt, it may cause a gradual decrease in system utility rather than a severe response.

The two categories are inspired by a well-known category of uncertainty faced by self-adaptive systems [27]. We believe that these categories could guide our detection of model deviations since uncertainty is one of the primary causes of model deviation.

4 AN ACTIVE MONITORING MECHANISM FOR MONITORING CONTROL-SASS

In this section, we present our Active Monitoring Mechanism for detecting model deviation in control-SASs. We begin by presenting a stochastic framework that connects the implicit variation of model deviation with its explicit manifestation. Following this, we describe two components of AMM: a four-way monitor that assesses the confidence value of whether model deviation occurs, and a remediator that takes remedial action once the monitor reports low confidence. The monitor and remediator form a supervising mechanism for detecting a control-SAS's model deviation and preventing its damage, as shown in Figure 4.

4.1 A stochastic framework

We propose a stochastic framework to facilitate AMM's design of its monitor and remediator. The framework, presented in Equation 1, is a time-varying state-space model. The model connects a control-SAS's observable variables (i.e., $u(t)$, $y(t)$, and $\alpha(t)$) to its implicit model parameter $\theta(t)$. The variables are describes in a stochastic manner to address the uncertainty issue.

$$\begin{cases} \theta(t) \sim \mathcal{N}(f_t(\theta(t-1)), Q) & (a) \\ f_t(\theta) = \theta + \varepsilon(t) + \omega \cdot \Theta & (b) \\ y(t) \sim \mathcal{N}(\mathcal{LS}(y(t-1), \theta(t), u(t), \alpha(t)), R) & (c) \end{cases} \tag{1}$$

Basically, the framework consists of two parts. Equations 1-(a) and (b) describe the implicit variation of a model deviation, while Equation 1-(c) illustrates its explicit manifestation.

In its first part, the framework represents the implicit variation as the transition of the model parameter $\theta(t)$. We define $\theta(t)$ as a stochastic variable subject to a normal distribution. The variable's mean value is set by a transition function f_t. The variance of $\theta(t)$, Q, remains unchanged in the function. This is because that the variance represents the noise of θ introduced during system identification. The value of Q is set as the variance of the parameter's identified values during system identification.

The transition function f_t, as shown in Equation 1-(b), takes into account the two types of implicit variations of model deviation. The external unexpected change is described as a low probability change point $\omega \cdot \Theta$. Considering the various factors that can cause an unexpected change, it could be very difficult to set the probability ω and intensity Θ appropriately. However, our investigation finds that unexpected changes lead to drastic consequences. Thus, AMM employs hypothesis testing to identify change point-typed deviations. We compare our observations to a prior prediction to determine whether the system is free from unexpected changes.

The internal system noise is described by a random walk procedure that captures the inherent fluctuations of the system. The step of the random walk $\varepsilon(t)$ is a stochastic variable that follows a normal distribution. We set the mean of the variable to zero and the variance to the system noise. Note that the actual system noise is difficult to infer. AMM approximates the variance of $\varepsilon(t)$ with the identification error Q. An iterative Bayesian process is introduced to calibrate the variance during deviation detection.

In its second part (Equation 1-(c)), the framework quantifies the explicit manifestation using the nominal model, $\mathcal{LS}$. More specifically, $\mathcal{LS}$ is a recursive equation connecting $\theta(t)$, $u(t)$, $\alpha(t)$, and $y(t)$. Figure 1 presents an example of $\mathcal{LS}$ for our motivating BSN system. The system output $y(t)$ is defined as a normal distributed variable. We consider the variance of $y(t)$, R, as the measurement error on observable variables. The value of R can be determined during system identification.

Based on the framework, we can further define vague observation as follows. *Vague observation* describes a combination of the values of the observable variables of a control-SAS, ($u(t)$, $\alpha(t)$, $y(t)$, $y(t-1)$). The values should satisfy Equation 1-(c) with any model parameter value $\theta(t)$. If the equation can only be satisfied with certain parameter values (taking into account system noise and measurement error), then we have non-vague observation instead. In the case of BSN, a vague observation would be a combination where the control signal $u(t)$ is zero. According to BSN's nominal model $\mathcal{LS}$, such a zero control signal would make an arbitrary $\theta(t)$ satisfy Equation 1-(c). Therefore, no existing approach can detect the potential deviation of the parameter.

4.2 Monitor

AMM proposes a four-way monitor to evaluate its confidence that a control-SAS experiencing a model deviation. The monitor combines the identified deviation categories and characteristics of control-SASs to introduce different techniques in different cases as follows.

- Case 1: non-vague observation, random walk-typed (C_1): we use *Bayesian inference* to estimate the current distribution of $\theta(t)$ and compare it with the safety range of θ;

- Case 2: non-vague observation, change point-typed (C_2): we use *Bayesian hypothesis testing* to predict a prior distribution of $y(t)|\theta(t)$ and compare it with the observed $y(t)$;
- Case 3: vague observation, random walk-typed (C_3): we use *worst-case estimation* to predict a prior distribution of $\theta(t)$ and compare it with the safety range of θ;
- Case 4: vague observation, change point-typed (C_4): we use *autoregressive hypothesis testing* to predict a prior distribution of $y(t)|\alpha(t)$ and compare it to the observed $y(t)$.

4.2.1 Estimating random-walk deviation with non-vague observation. When AMM receives a non-vague observation, it uses Bayesian inference to estimate the distribution of $\theta(t)$. Since the transition $f_t(\theta)$ (i.e., Equation 1-(b)) cannot be obtained directly, AMM approximates it using the standard Gaussian process function $GP(t)$ (i.e., i.e., $\theta(t) = GP(t)(\theta(t-1))$). Based on $GP(t)$, AMM derives $\theta(t)$'s prior estimation, $\theta(t)|GP(t)$. Recalling that $\theta(t)$ is a stochastic variable, $\theta(t)|GP(t)$ is also described as a distribution. The corresponding confidence C_1 is calculated as the cumulative probability that the distribution $\theta(t)|GP(t)$ falls within θ's safety range $[\underline{\theta}, \overline{\theta}]$.

We set the mean function of $GP(t)$ to zero, indicating that $\theta(t)$ is without any change points. We use the standard kernel function with a variance of Q as the initial variance function of $GP(t)$ (i.e., the variance function of $GP(1)$). The standard kernel function allows us to approximate the random walk process of $\theta(t)$. AMM also introduces a regression process that calibrates the variance function of $GP(t)$ with the most recent observations. Following Equation 1-(c), the regression process begins by deriving the posterior distribution of $\theta(t)$, denoted as $\theta(t)|y(t)$. Afterwards, a new covariance matrix that incorporates $\theta(t)|y(t)$ is constructed to update the kernel function of $GP(t)$.

4.2.2 Estimating change-point deviation with non-vague observation. Similar to C_1, evaluating C_2 begins with the estimation of the prior $\theta(t)|GP(t)$. Instead of directly calculating confidence, we use Bayesian hypothesis testing to detect potential model deviations. This is because that a single prior estimation $\theta(t)|GP(t)$ is not sufficient when considering a change-point type deviation, since $GP(t)$ can no longer approximate the transition of $\theta(t)$.

More specifically, our hypothesis is $\theta(t)$ being free of a change point. We first derive a prior prediction of $y(t)|GP(t)$ for $y(t)$ based on $\theta(t)|GP(t)$ using Equation 1-(c). Next, we compare the predicted $y(t)|GP(t)$ with the observed $y(t)$ to test our hypothesis. We calculate the confidence C_2 as the probability that $y(t)$ is subject to the predicted $y(t)|GP(t)$. If the hypothesis test passes, we will move on to the regression phase to calibrate $GP(t)$ and ready ourselves for the next self-adaptation loop.

4.2.3 Estimating random-walk deviation with vague observation. When estimating the confidence with vague observation, Equation 1-(c) becomes meaningless since any $\theta(t)$ would satisfy the equation. However, Equations 1-(a) and (b) remain intact, allowing us to derive the prior estimation $\theta(t)|GP(t)$. Thus, the primary challenge in assessing C_3 is how to calibrate our prior, considering that the approximation of $GP(t)$ may be excessively imprecise to be functional.

AMM calibrates the prior by considering the worst-case scenario. Given $\theta(t-1)$, we produce two posterior estimates of $\theta(t-1) + Q$ and $\theta(t-1) - Q$. These are two pessimistic estimation since we assume $\theta(t)$ reaches its maximum deviation. We update the kernel function of $GP(t)$ with these two estimations. This guarantees that no random walk-typed deviation would exceed our estimation.

4.2.4 Estimating change-point deviation with vague observation. When assessing C_4, we also use hypothesis testing to detect potential deviations. Since Equation 1-(c) no longer works, we employ an autoregressive model to predict $y(t)|\alpha(t)$ based on the environmental input $\alpha(t)$. To assess whether $\theta(t)$ has undergone unexpected changes, $y(t)|\alpha(t)$ is compared against the observed $y(t)$, similar to the approach for evaluating C_2.

The autoregressive model is a prominent tool for characterizing stable systems' behavior. We check the stability of involved variables as follows. First, $\theta(t)$ should be free of sudden changes, considering our hypothesi that no change-point deviation occurs. Second, $u(t)$ should also be free of sudden changes, considering the undergoing vague observation. Third, the sudden change of $\alpha(t)$ can be easily detected and will trigger non-vague observations. Since all variables remains remain free from unexpected changes, the autoregressive model is applicable to predicting $y(t)$.

To balance its efficiency and effectiveness, we have decided to set the order of our autoregressive model to two, in accordance with the existing literature [37]. As such, the model can be described as a linear model $\mathcal{AR}$ that $y(t) = \mathcal{AR}(y(t-1), y(t-2), \alpha(t), \alpha(t-1))$, since the involved variables should change gradually. The model's parameters can be determined through system identification by setting $u(t)$ to zero.

4.3 Remediator

AMM's remediator is designed to take different remedial actions in response to the monitor's output. We set a confidence interval of 0.997 (according to the prevailing 3σ rule [58]) for the four derived confidences. If C_1 or C_2 falls below the threshold, it indicates the presence of model deviation. If C_3 or C_4 falls within the warning range, it suggests that the model deviation could be masked by vague observations.

In the first situation, AMM switches from the control-based managing system to a mandatory rule-based managing system. This is a common strategy that sacrifices system efficiency for reliability [66, 69, 71]. The deviation between the nominal model and the control mechanism can be corrected by online identification [60] or runtime calibration techniques [72]. Considering that there is a wealth of established literature in this field, our AMM focuses in addressing the detection of potential model deviations through a well-designed active control signal.

Once C_3 or C_4 falls below the warning threshold, AMM triggers an active control signal u_A in the next self-adaptation loop. AMM balances u_A's effectiveness, cost and risk through optimizing its intensity with Equation 2. We set the optimization objective as a three-terms fitness function. Intuitively, the effectiveness term Fit_e demands that the subsequent system output $y(t+1)$ possess a greater absolute value. The cost term Fit_c requires a reduced intensity of u_A with a smaller absolute value. Finally, the risk term Fit_r mandates a larger distance between $y(t+1)$ and its safety limit.

$$
\begin{aligned}
& u_A = argmax \ \text{Fit}_e - \text{Fit}_c - \text{Fit}_r \\
& \text{Effectiveness: } \text{Fit}_e = y(t+1)^2 \\
& \text{Cost: } \text{Fit}_c = u(t)^2 \\
& \text{Risk: } \text{Fit}_r = \frac{1}{(y(t+1) - \underline{y})^2} + \frac{1}{(y(t+1) - \overline{y})^2} \\
& s.t. \quad \text{safety: } \begin{cases} \underline{y} \le y(t+1) \le \overline{y} \\ y(t+1) = \mathcal{LS}(\theta, u(t), \alpha(t)) \end{cases}
\end{aligned} \tag{2}
$$

The active signals would affect the execution of the control-SAS and inevitably destabilize the system. After the injection of an active signal $u(t)$, the output of the managed system $y(t+1)$ will be pushed away from its setpoint. Consequently, the managing system will produce control signals to pull the system back to its equilibrium point. So the question here is whether $u(t)$ would push $y(t+1)$ so hard that it exceeds the managing system's ability to pull the system back to steady state. Note that if the system is currently suffering from model deviation, then it is impossible to provide a theoretical guarantee that $u(t)$ will not undermine the safety of the system. This is because an adversarial deviation of $\theta(t)$ would amplify the side-effect of $u(t)$ and lead the system to unsafe executions.

AMM's solution to this safety problem is twofold. First, we theoretically avoid the unsafe signals for the situation when model deviation does not happen. We set a safety constraint on the derived $u(t)$ (as shown in Equation 2), which uses a safety range of the upcoming system output $y(t+1)$. This safety range $[\underline{y}, \overline{y}]$ is determined by the Lyapunov property of control-SASs. The property guarantees that the system can eventually converge to the equilibrium point. The derivation of the safety range has been studied by the control theory community [35, 44, 45] and is beyond the scope of this paper. Second, we introduce a heuristic optimization term Fit_r to reduce the risk of the active signal. Intuitively, this term requires an active signal that pushes the upcoming system output $y(t+1)$ away from its safety limits.

We use constraint solvers like Z3 [25] to derive the most suitable $u(t)$. The result will be rounded and employed as the intensity of u_A. In case the optimal value is zero, the intensity will be set to the minimum non-zero value. Once u_A is triggered, AMM's monitor receives a non-vague observation and could evaluate C_1 and C_2 accordingly. If these two confidence are acceptable, AMM will produce an additional active control signal u_A^{-1} with the opposite intensity of u_A to offset the side effects caused by u_A.

4.4 Design specificity for control-SASs

We leverage the knowledge of control-SASs to design the AMM approach from the following three perspectives.

First, *AMM's supervising mechanism is designed based on the specificity of the model deviation problem faced by control-SASs.* For these systems, model deviation reduces the reliability of the system and calls for a mechanism to recover the system from unsafe executions. Software engineers often rely on approximate nominal models to design control-SASs. It is nearly impossible to derive an accurate nominal model for control-SAS designing because a software system can hardly be described from a first principle. The gap between the approximate model and the actual behavior of the system results in model deviation and leads to unsafe execution.

For conventional control systems, however, model deviation actually helps optimize the system's control strategy. This is because these systems are governed by physical laws that allow precise nominal models to be identified. Control engineers not only describe the behavior of the system with precise nominal models, but also describe the potential deviations precisely [15]. At runtime, model deviation detection determines which nominal model (or model parameters) better fits the current behavior of the system and calls a monitor-switch loop to support optimized control strategies (e.g., the model reference adaptive control approach [56, 63, 76]).

Second, *AMM's effective monitor is designed based on the specific types of deviations suffered by control-SASs.* We classify the deviations faced by control-SASs into different categories. We propose the use of different techniques to detect different types of deviations without a precise description. When estimating the random walk deviation with non-vague observation, we use the standard Gaussian process function to approximate the random walk deviation. The standard Gaussian process works here because the normal distribution would represent the non-dramatic uncertainty suffered by self-adaptive systems in many cases [50, 75]. We introduce hypothesis testing when estimating the change-point deviation with non-vague observations. This is because such a deviation would cause the control-SAS to behave significantly abnormally, and hypothesis testing is effective in detecting such behavior for black-box software systems [18–20, 26].

When estimating the deviation with vague observations, the monitor compromises its accuracy for timeliness. In particular, we can tolerate the false alarms reported by the monitor, but we cannot afford missing detections. This is because active signals can compensate for our false alarms by providing non-vague estimation result, but cannot recover missing detections. We use two pessimistic techniques to estimate the deviation with vague observation, where they make a decision based on the worst case scenario. This makes the two techniques too imprecise to be useful for general control systems, but still effective for triggering our active control signals.

Finally, *AMM's required inputs are simplified based on the specific background of software engineers.* In addition to the variables used in the development of a control-SAS, AMM requires only the model parameter θ's variance Q and the system output y's variance R as inputs. Moreover, the values of these two variables can be obtained during the system identification procedure with minimal statistical knowledge. AMM does not require engineers to fine-tune its parameters, as many control system-oriented approaches do (e.g., the window size and threshold of the sliding-window approach).

5 EXPERIMENTS

In this section, we experimentally evaluate the effectiveness and usefulness of AMM [1] on three representative subject systems.

Our baselines for comparison include four approaches, three of which are originally proposed by control system researchers, including the sliding window approach ("SWD" for short) [15, 53], the autoregressive moving average approach ("ARMA" for short) [41], and the parameter estimation approach ("PED" for short) [43], and the remaining approach model-guided deviation detector ("MoD2" for short) is proposed by control-SASs researchers. We selected SWD and ARMA as our baselines because they are the most widely used approaches for monitoring control-SASs. We selected PED since it is often selected as a baseline together with SWD and ARMA by control theory researchers. We selected MoD2 [69] as our baseline because it is the state-of-the-art approach for monitoring control-SASs. It is recently proposed and takes into account the dynamic characteristics of control-SASs in detecting model deviations. MoD2 was implemented based on the provided open access repository. The authors implemented the other three baselines by closely following the literature descriptions [15].

The evaluation investigates the following two research questions:

RQ1 *Can AMM effectively detect a control-SAS's model deviations?*
RQ2 *How useful is AMM-based supervising mechanism in preventing a control-SAS's abnormal behavior and improving its utility?*
RQ3 *What is the cost of AMM's detection in terms of resource consumption and execution time?*

5.1 Experimental setup

5.1.1 Experimental subjects. We selected three subject systems from the community of self-adaptive systems as our managed systems.

BSN is a body sensor network system we described in Section 2. The corresponding managing system consists of six proportional–integral controllers that adjusts the sensors' sampling rates. These controllers are implemented following [34].

SWaT is an operational scaled-down water treatment plant emulator [8]. The system consists of five water-processing tanks, as well as the valves that control the in-coming and out-groining water of the each tank. The corresponding managing system consists of seven programmable logic controllers that control those vales remotely, following the methodology of on-off control [33, 62].

[1]The artifact and experimental data are available at https://github.com/supervisionAMM/AMM.

RUBiS is a web auction system that has been studied in [14, 47, 54]. The system receives remote network requests and delivers services of different levels of qualities. The corresponding managing system is a proportional–integral–derivative controller that can adjust the number of servers in order to adapt to workload or network changes, following the push-button method [29].

Table 2. The experimental configurations

subject	**environmental input** α	**deviation** dev
BSN	the measurement error that affects the quality of the sensed data [34]	sensor noise that cost the sensors more power to sample data [7]
SWaT	input extracted from a real-world testbed [8]	injecting manipulation disturbance of the valves according to [28]
RUBiS	user requests generated based on two real-world workloads [5, 6]	injecting attacks to the system's settings according to [2]

5.1.2 Experimental Configurations. Table 2 presents the configurations used in our evaluation. The environmental input α describes the input fed into the subject system. Deviation describes the model deviation injected into the subject system. These configurations either employ existing datasets or were implemented by the authors based on existing literature.

Note that Table 2 provides details on the preparation of configurations rather than the concrete (α, dev) pairs in our experiments. To conduct experiments, each subject system was prepared with a different number of (α, dev) pairs, 400 pairs for *BSN*, 400 pairs for *SWaT* and 160 pairs for *RUBiS*. If a configuration pair has insignificant deviation (not exceeds the safety range), then it is classified as a *negative configuration*. Conversely, if it exhibits significant deviation, it is labeled as a *positive configuration*.

5.1.3 Evaluation criteria. In **RQ1**, we are mainly concerned with the timeliness and accuracy of the compared approaches in detecting model deviations. We measure timeliness by the *mean time delay* (MTD, the average interval between the time when the model deviation *dev* is injected and the detection time of the approach). We measure accuracy by the *false-negative rate* (FN rate) and the *false-positive rate* (FP rate). The false negative rate concerns the missed detections, i.e. the detection approach classifies a positive execution as a negative one and lets the subject system run abnormally. A high false negative rate results in lower system safety. For the false positive rate, it concerns the false alarms, i.e. the detection approach classifies a negative execution as a positive one and triggers the remedial actions incorrectly. A high false positive rate results in lower system utility.

Note that it is straightforward to compute the false positive rate for negative configurations, since any reported model deviation is a false alarm. For positive configurations, however, the false positives only affect the negative execution of the configuration. Consider that the model derivation is injected at time t. If an approach reports a deviation before time t, then it should be considered a false alarm. If the deviation is reported after time t, then it should be considered a good detection and will not be counted in computing the false positive rate.

In **RQ2**, we measure usefulness by the system's abnormal rate and achieved utility ratio. The *abnormal rate* is calculated as the ratio of the time the system is in an abnormal phase (i.e., violating control properties such as "SASO") to the time the system suffered a model deviation. The *utility ratio* is calculated as the ratio of utility gained by the system with or without a supervising mechanism. The original subject system's utility ratio will be 100. To measure the utility gained by different

Table 3. Comparison of AMM with the baselines and AMM's alternatives

	BSN			SWaT			RUBiS		
	MTD	FN(%)	FP(%)	MTD	FN(%)	FP(%)	MTD	FN(%)	FP(%)
SWD	193.7	35.0	46.8	363.1	91.0	0.2	409.2	27.5	54.4
ARMA	321.6	1.0	0.0	204.0	51.0	25.5	290.8	36.3	68.1
PED	322.2	0.5	0.0	267.9	24.5	8.0	495.67	20.0	12.5
MoD2	314.6	0.0	0.0	266.3	0.5	0.0	488.9	22.5	4.4
AMM	17.1	0.0	0.0	195.8	0.0	0.0	488.3	22.5	5.0
	BSN$^+$			**SWaT$^+$**			**RUBiS$^+$**		
T_{random}	18.1	0.5	0.5	229.4	0.0	0.0	143.8	61.3	61.3
I_{opt1}	25.3	0.0	0.0	241.6	0.5	0.0	501.3	23.8	10.0
I_{opt2}	18.1	0.5	0.5	264.8	0.5	0.0	508.9	22.5	10.0
I_{random}	9.11	73.0	73.0	206.8	0.0	0.0	510.0	20.0	12.5
Summary: AMM achieves 33.0% *smaller MTD,* 18.3% *lower FN rate and a* 16.7% *lower FP rate*									

systems, we refer to existing work [55], which assigns different utility gains to different execution phases. More specifically, the system's utility gain is 1.0 per adaptation loop during its abnormal phase, 1.5 per adaptation loop in its low-controllability phase, and 2.0 per adaptation loop during its implicit phase and deviation-free execution. Additionally, we impose a utility loss for systems that fail to deliver the required function or violate crucial control properties, as noted in [17, 55].

In **RQ3**, we measure the resource consumption of the compared approaches by their increased CPU usage (denoted as "%CPU") and memory usage (denoted as "%MEM") from executing both detection approach and subject system, compared to executing only the subject system. The time cost of the compared approaches is measured by their execution time, which is the average time required for an approach to execute its algorithm in a self-adaptation loop.

5.1.4 *Experimental procedure.*

All experiments were conducted on a personal computer with 8 cores and 32 GB memory.

To answer **RQ1**, we compare the achieved MTD, FN rate, and FP rate of the five compared approaches on the three subject systems. We also compare the performance of AMM's alternatives with different active signal timings and intensities. The timing-based alternative T_{random} uses random timing to trigger the active signal. The intensity-based alternatives include I_{opt1}, which excludes Fit_e from its fitness function in Equation 2, I_{opt2}, which excludes Fit_r from its fitness function, and I_{random}, which uses random signal intensities. The comparison of AMM and its alternatives is conducted on the positive configurations only (i.e., BSN$^+$, SWaT$^+$, and RUBiS$^+$).

To answer **RQ2**, we compare the abnormal rate and utility ratio achieved by three subjects with and without a supervising mechanism. Additionally, we implement and evaluate four different mechanisms, by using the four baselines as their monitors. All the mechanism uses the same remediator. We only use the positive configurations here.

To answer **RQ3**, We compare the CPU usage, memory usage, and detection time of our AMM with the baselines (i.e. SWD, PED, ARMA and MoD2) on the three subject systems. To focus on the cost of the approaches, we used only negative configurations as input for the compared approaches (i.e., BSN$^-$, SWaT$^-$, and RUBiS$^-$). In addition, we did not stop the execution of the approach when it reported a positive result. In other words, in this part of the evaluation, the compared approaches kept detecting potential model deviation till the end of each experimental configuration.

5.2 Experiment Results

RQ1(Effectiveness) Table 3 first compares the performance of AMM with the baselines regarding their effectiveness in detecting model deviations. In terms of detection timeliness, AMM's mean delay time is the smallest for subjects *BSN* and *SWaT* (on average 93.8% smaller for *BSN* and 25.9% smaller for *SWaT*). However, for *RUBiS*, AMM's achieved MTD is higher than SWD and ARMA. This is because AMM requires a closed-loop identification procedure to determine its coefficient of variation, namely $\theta(t)$'s variance Q and $y(t)$'s variance R. For *RUBiS*, such an identification only used two workloads (which are mentioned in Table 2). The absence of sufficient workloads prevents us from systematically sampling the executions and obtaining accurate identification values for Q and R. This, in turn, undermines the timeliness of AMM. It's worth noting that the other two estimation approaches (PED and MoD2) also demand a closed-loop identification process, which accounts for their remarkably higher MTD than SWD and ARMA.

With regards to accuracy, AMM achieves satisfactory FN and FP rates in detecting model deviations. On average, AMM's FN rate is 7.5%, and its comparable FP rate is 1.67%. Except for *RUBiS*, in which AMM presents a marginally higher FN rate than PED (2.5% higher) and a slightly higher FP rate than MoD2 (0.6% higher), its FN and FP rates are the lowest compared to other approaches. There is a comparable discrepancy found in AMM's performance between *BSN/SWaT* and *RUBiS*. Again, the imprecise identification of AMM's coefficients contributes to this gap. However, the identification accuracy of AMM is significantly better than that of SWD and ARMA, in contrast to the timeliness results. This reflects the challenge of model deviation detection [69], which is to balance timeliness and accuracy. By sacrificing accuracy, one could significantly decrease the detection delay, but this would also lead to reporting a large number of false positives (54.4% FP rate for SWD and 68.1% for ARMA).

We note that AMM and MoD2 achieve the better performance among the five approaches compared. In some cases, their difference is small (e.g., in the case of subject *RUBiS*). Both AMM and MoD2 consider the characteristics of control-SASs instead of conventional control systems, which results in their high detection accuracy. The main difference between AMM and MoD2 is in their timeliness, where AMM uses active signals to reduce its detection delay (on average 34.5% shorter). Therefore, if the subject system rarely suffers from vague observations (e.g., in the case of a web service system like RUBiS), it may not make a difference whether AMM or MoD2 is used. If the subject system usually suffers from vague observations (e.g., in the case of cyber-physical systems like BSN and SWaT), AMM will be a better choice.

Table 3 also compares the performance of AMM with its alternatives, which involve active control signals with different timings and intensities. The results indicate that AMM outperforms all of its alternatives, which supports the claim stated in Section 1 that the main challenge of AMM is to balance the effectiveness and risk of its active signals. Some alternatives may have shorter MTD or lower FP/FN rates, but they cannot strike an optimal balance between timeliness and accuracy. For example, I_{random} achieves an impressive MTD of 9.11 in *BSN*, but its reported FN and FP rate are excessively high to be effective.

When comparing AMM's alternatives with the baselines, it is evident that some of the alternatives do not surpass the performance of the baselines. For example, T_{random} reports much higher FN and FP rate than PED and MoD2 in *RUBiS*, while I_{opt1} and I_{opt1}'s MTD at the same level as PED and MoD2 in *SWaT*. These findings suggest that active signals may not always contribute positively to model deviation detection; rather, such signals may, at times, prove counterproductive. It is worth noting that, in certain scenarios, the alternatives yield identical FN and FP rates. This is because only positive configurations are used here, and a false alarm prior to the deviation injection will also result in a false negative outcome.

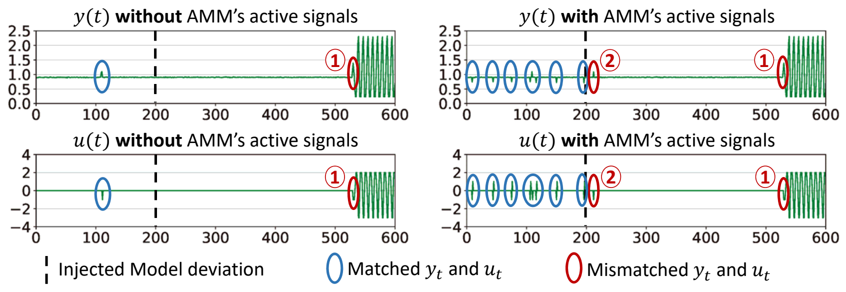

Fig. 5. Detecting model deviation w/wo AMM's active signals

To better demonstrate the effectiveness of AMM's active signals in detecting model deviation, we analyze the motivating execution illustrated in Figure 2-(b), in which the baselines fail to detect the deviation in time (the observable mismatch pair appears at time 530, as indicated by mismatch①). As shown in Figure 5, without the support of active signals, vague observation makes it impossible to detect model deviation before the system enters oscillation, since all observed $y(t)$ and $u(t)$ are consistent. However, when AMM introduces its active signals, a mismatch between the active signal $u(211)$ and $y(212)$ (as indicated by mismatch②) enables it to detect the introduced deviation timely and accurately.

In summary, the answer to our research question RQ1 is as follows. *AMM can detect model deviation with low detection delay and high accuracy. Comparing with the baselines, AMM achieves a smaller detection delay (on average* 33.0% *smaller) as well as better accuracy (with an* 18.3% *lower FN rate and a* 16.7% *lower FP rate).*

RQ2 (usefulness) Table 4 displays the abnormal rates and utility ratio of the subject systems with different supervising mechanisms. We first focus on the abnormal rate. The result shows that our AMM-based supervising mechanism greatly reduces the abnormal rates of the subject systems (19.3% on average), as compared to the original unsupervised system. Furthermore, our mechanism showcases a superior outcome with the lowest average abnormal rates at 2.4%, as compared to other supervising mechanisms. This outcome reflects AMM's performance in *RQ1*, where it exceeds all baseline approaches in achieving a balance between timeliness and accuracy.

Table 4. Comparison of the abnormal rates (Abn. rate) and utility ratio (Uti. ratio) with different supervising mechanism

		Original	**SWD**	**ARMA**	**PED**	**MoD2**	**AMM**
BSN	Abn. rate	19.6%	7.6%	1.0%	0.7%	0.0%	0.0%
	Utt. ratio	100.0	143.0	152.2	152.9	155.4	188.0
SWaT	Abn. rate	3.4%	1.9%	1.3%	0.1%	0.0%	0.0%
	Uti. ratio	100.0	104.7	117.9	125.2	130.7	130.8
RUBiS	Abn. rate	42.1%	14.7%	22.9%	10.0%	7.8%	7.2%
	Uti. ratio	100.0	214.1	204.0	229.8	233.1	245.9
Summary: AMM decreases abnormal rate by 19.3% *and increase utility by* 88.2%							

It is worth noting that AMM performs better on *BSN* and *SWaT* than it does on *RUBiS*, which is similar to the finding in *RQ1*. The imprecise identification of the variance coefficient makes it challenging for estimation-based approaches, like PED, MoD2, and AMM, in effectively leveraging their advantages to monitor changes in nominal models online. Although AMM performs better than other methods, it still has limitations that we aim to address in future work.

The reduced abnormal rates demonstrate what the supervising mechanism is able to avoid, while the utility shows the benefits it provides to the system. Results indicate that AMM enhances the utility of control-SASs by an average of 88.2% (30.8%–145.9%), compared to the original systems without a supervising mechanism. We observe that AMM obtains the highest utility under all cases, which can provide some insight into the impact of AMM's active control signals on the execution of the subject systems. Due to space constraints, we cannot engage in theoretical discourse on this topic. However, the high utility achieved by AMM illustrates that its active signals have minimal side effects. In other words, the utility gain brought by the signals outweighs the potential utility loss caused by their activation.

According to Table 4, AMM's leading advantages in utility is larger than it in abnormal rates, especially when compared to the other two parameter estimation-based approaches. On average, AMM outperforms PED and MoD2 by 0.7% in the lower abnormality rate, and its lead in higher utility has increased to 17.1%. This outcome better highlights the usefulness of AMM's active control signals. While previous methods may identify the model deviation before a failure occurs, they must wait for a non-vague observation that leads the system into the low-controllability phase and reduces the utility. In contrast, the active control signal of AMM permits the detection of model deviation in the implicit phase, thereby preserving utility during the low-controllability phase.

In summary, the answer to our research question *RQ2* is as follows. *Our AMM-based supervising mechanism can alleviate the impact of model deviation. When compared to a system without the supervision mechanism, the abnormal rate on average decreased by* 19.3% *with the use of the AMM-based mechanism. Additionally, the utility of the system improved by* 88.2% *on average.*

Table 5. Comparison of AMM's resource consumptions (cpu usage %CPU and memory usage %MEM) and time cost (ms) with the baseline approaches

	BSN⁻			**SWaT⁻**			**RUBiS⁻**		
	%CPU	%MEM	time	%CPU	%MEM	time	%CPU	%MEM	time
SWD	+0.8	+0.9	0.08	+4.2	+0.1	0.02	+0.5	+0.6	0.01
ARMA	+0.9	+1.90	4.8	+3.7	+1.60	1.17	+0.4	+2.0	0.74
PED	+0.3	+1.8	0.72	+3.4	+1.1	0.23	+0.5	+1.6	0.11
MoD2	+1.6	+1.80	0.87	+5.4	+1.10	1.47	+1.3	+1.6	0.19
AMM	+2.1	+2.3	12.65	+5.7	+1.6	3.01	+1.4	+2.1	1.63

RQ3 (cost) Table 5 compares the resource consumption of the approaches. The result shows that when the AMM approach is activated, CPU usage increases by an average of 3.1% (1.4%–5.7%) and memory usage increases by an average of 2.0% (1.6%–2.3%). We note that AMM consumes more resources than the baselines due to its more complicated design and structure, but the difference is still in the same order of magnitude. Taking into account the resource consumption of the subject control-SAS, the overhead of the detection approach is quite small. The CPU consumption for the three subject systems is 48.1%, 83.5%, and 58.6%, respectively, and the memory consumption is 27.5%, 0.6%, and 1.8%, respectively. Given the usefulness of AMM in terms of saving system utility from model deviations, we believe that the resource consumption of AMM is affordable.

Table 5 also shows the time cost of the compared approaches. Similar to the resource consumption result, our AMM takes the longest time to execute its detection algorithm (ranging from 1.62 ms to 12.65 ms). However, this "long" execution time has a limited impact on the timeliness of the AMM, since the execution time of the detection approach is insignificant compared to the self-adaptation loop of the control-SAS (which would range from one second to one minute). AMM's well-designed monitor allows it to make accurate detections with fewer observations. In other words, AMM saves time waiting for the subject systems to iterate their self-adaptation loops and improves its timeliness.

In summary, the answer to our research question *RQ3* is as follows. *Our AMM costs more resources and execution time than the baseline approaches, and the difference is in the same order of magnitude. Given AMM's leading performance in terms of effectiveness and usefulness, we believe the cost of AMM is acceptable.*

5.3 Threats to Validity

A major concern about the validity of our experimental conclusion is that we have conducted experiments with environment simulators. In other words, we use simulators to generate the environmental inputs to the subject managed systems, and receive their produced outputs. This is due to efficiency issues, considering that executing these control-SASs within the real environment would take an incredibly long time. For example, it would take us 30 minutes to run an experimental execution of the SWaT system on the real-life water treatment plant. As a result, it would take us over 1000 hours to complete the SWaT-related experiments. By using a simulator, we speed up the execution of SWaT to one minute per configuration, which allows us to cover more different configurations in our experiments. However, to reduce the bias introduced, all simulators are either provided by or suggested by the developers of the target system. We believe that this has greatly reduced the side effects of using simulators.

In our evaluation, we implemented the control theory-based baselines ourselves. This may threaten the validity of the experimental results. Incorrect implementations can report lower detection accuracy metrics. Less efficient implementations can report longer detection delays. To ensure the correctness of the implementations, we carefully selected the guidelines and strictly followed their descriptions. We followed [15] as our guideline, which is a classic textbook in control theory. We also thoroughly tested our implementations to avoid programming errors. Regarding efficiency, we made efforts to optimize our implementations, but it is difficult to claim that they achieved optimal efficiency. However, when comparing the execution time (in milliseconds) of the baselines with the detection delay (in seconds), improving the implementation's efficiency by a few milliseconds may be insignificant in reducing the detection delay. Therefore, we believe that our experimental results on the baselines are convincing.

6 RELATED WORK

The deviation of the control-SAS's nominal model during runtime has been studied by self-adaptive systems researchers since the emerging of control-SASs. Some researchers have proposed different remedies that allow a control-SAS to overcome the deviation of its nominal model. Filieri et al. use continuous learning mechanisms to keep the nominal model updated at runtime [30]. Maggio et al. use Kalman filters to revise the identified nominal model by updating its state values [51]. Wang et al. introduce an additional monitor-reconfigure loop that sacrifices the managing system's statistical guarantees in return for the whole system's robustness [72]. Compared to these works, our AMM focuses on the detection of the nominal model's deviation. Tong et al. propose a Kalman filter-based mechanism, MoD2, that can automatically switch the control logic of the system when the nominal model has changed [69]. MoD2 takes into account the dynamic characteristics of control-SASs when tuning its Kalman filter-based detector, but it cannot overcome the problem of vague observations.

The problem of model deviation detection has been studied by the control theory community. The most widely-used approaches including the sliding window approach [15, 36, 53], the autoregressive moving average approach [23, 41, 61], and the parameter estimation approach [1, 21, 43]. These approaches follow the similar intuition that model deviation could cause a mismatch between the control signal and the system's response. Specifically, the sliding window approach uses a sliding window to overcome the measurement error. The auto-regressive moving average approach defines the mismatch using the auto-regressive model. The parameter estimation approach uses a state space model to estimate the value of the deviated parameters, while mitigating the effects of measurement error and system noise. The limitation of these control theory-based approaches is that they require a precise model of the deviation, which is unlikely in the context of control-SASs. In addition, these approaches cannot handle the problem of vague observations, since this phenomenon is control-SASs-specific.

AMM's supervising mechanism is similar to the structure of model reference adaptive control ("MRAC" for short) [56, 63, 76] in control engineering. In MRAC, a reference model is introduced to describe the mismatch between the theoretical behavior of a control system and its observable behavior. The reference model is used to calibrate the nominal model of the control system online [74]. Some MRAC approaches also use stochastic frameworks to improve their effectiveness [40, 48]. Our AMM differs significantly from MRAC approaches in the following two ways. First, MRAC approaches require a reference model to specify the desired system output, while AMM requires only the nominal model of the system. Second, MRAC approaches detect the difference between the theoretical output of the system and the actual output, while AMM focuses on the mismatch between the control signal and the system output to detect model deviation.

Runtime monitoring has been exploited by software engineering researchers to identify the abnormal behavior of the system. Chen et al. propose an SVM-based method to detect network attacks on SWaT systems [22]. Qin et al. use context information to refine the derived invariants and combine multiple test results to improve the accuracy of anomaly detection [59]. Pastore et al. use statement coverage information during program execution to improve the accuracy of anomaly detection during runtime monitoring [57]. The main difference between our AMM and these approaches is that we combine passive monitoring with well-designed active control signals that reduce the time delay passive approaches have to wait for sufficient observations.

7 CONCLUSION

In this work, we propose an active monitoring mechanism to detect model deviation for control-SASs. AMM uses active control signals to stimulate the control-SAS when vague observations might mask model deviations. To determine the appropriate time for triggering the active signals, AMM proposes a stochastic framework to quantify the relationship between the implicit variation of a control-SAS and its explicit observation. Based on this framework, AMM's monitor and remediator enhance model deviation detection by generating active control signals of well-designed timing and intensity. Results from empirical evaluations with three representative systems demonstrate AMM's effectiveness (33.0% shorter detection delay, 18.3% lower FN rate and 16.7% lower FP rate) and usefulness (19.3% lower abnormal rates and 88.2% higher utility).

The AMM approach has limitations, as it assumes a normal distribution for uncertainty and a linear system-based nominal model for the subject managed system. Further research could explore relaxing these assumptions. For instance, the normal distribution can be extended to an exponential distribution when considering service time in a micro-service system, or a Cauchy distribution when considering network transmission time. Introducing new distribution types requires updating AMM's monitor to estimate the model parameters of additional types of deviations. When faced with a complex managed system that needs to be decomposed into sub-modules, a nonlinear system-based model can be considered. This model can describe the interdependent behavior of the submodules and enable the monitoring mechanism to detect potential deviations.

REFERENCES

[1] Ryan Prescott Adams and David JC MacKay. 2007. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742* (2007).

[2] Nadia Alshahwan and Mark Harman. 2011. Automated web application testing using search based software engineering. In *Proceedings of the 26th IEEE/ACM International Conference on Automated Software Engineering*. IEEE, 3–12.

[3] Konstantinos Angelopoulos, Alessandro Papadopoulos, Vítor Silva Souza, and John Mylopoulos. 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*. IEEE, 35–46.

[4] Konstantinos Angelopoulos, Alessandro Papadopoulos, Vítor Silva Souza, and John Mylopoulos. 2018. Engineering self-adaptive software systems: From requirements to model predictive control. *ACM Transactions on Autonomous and Adaptive Systems* 13, 1 (2018), 1–27.

[5] Martin Arlitt and Tai Jin. 2000. A workload characterization study of the 1998 world cup web site. *IEEE Network* 14, 3 (2000), 30–37.

[6] Martin Arlitt and Carey Williamson. 1996. Web server workload characterization: The search for invariants. *ACM SIGMETRICS Performance Evaluation Review* 24, 1 (1996), 126–137.

[7] Gabriel E Arrobo and Richard D Gitlin. 2011. Improving the reliability of wireless body area networks. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2192–2195.

[8] Kaung Myat Aung. 2015. Secure water treatment testbed (SWaT): an overview. *Singapore University of Technology and Design* (2015).

[9] Abhijit Badwe, Ravindra Gudi, Rohit Patwardhan, Sirish Shah, and Sachin Patwardhan. 2009. Detection of model-plant mismatch in MPC applications. *Journal of Process Control* 19, 8 (2009), 1305–1313.

[10] Taposh Banerjee, Miao Liu, and Jonathan P How. 2017. Quickest change detection approach to optimal control in markov decision processes with model changes. In *2017 American control conference (ACC)*. IEEE, 399–405.

[11] Stephen P Banks. 2002. Three-dimensional stratifications, knots and bifurcations of two-dimensional dynamical systems. *International Journal of Bifurcation and Chaos* 12, 01 (2002), 1–21.

[12] Stephen P Banks and Song Yi. 2006. Elliptic and automorphic dynamical systems on surfaces. *International Journal of Bifurcation and Chaos* 16, 04 (2006), 911–923.

[13] Saeid Barati, Ferenc Bartha, Swarnendu Biswas, Robert Cartwright, Adam Duracz, Donald Fussell, and et al. 2019. Proteus: Language and runtime support for self-adaptive software development. *IEEE Software* 36, 2 (2019), 73–82.

[14] Luciano Baresi, Sam Guinea, Alberto Leva, and Giovanni Quattrocchi. 2016. A discrete-time feedback controller for containerized cloud applications. In *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. 217–228.

[15] Michele Basseville and Igor Nikiforov. 1993. *Detection of abrupt changes: theory and application*. Vol. 104.

[16] Javier Cámara and Rogério De Lemos. 2012. Evaluation of resilience in self-adaptive systems using probabilistic model-checking. In *2012 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, 53–62.

[17] Javier Cámara, Gabriel A Moreno, and David Garlan. 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*. 155–164.

[18] Matteo Camilli, Carlo Bellettini, Angelo Gargantini, and Patrizia Scandurra. 2018. Online model-based testing under uncertainty. In *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 36–46.

[19] Matteo Camilli, Angelo Gargantini, and Patrizia Scandurra. 2020. Model-based hypothesis testing of uncertain software systems. *Software Testing, Verification and Reliability* 30, 2 (2020), e1730.

[20] Matteo Camilli, Angelo Gargantini, Patrizia Scandurra, and Catia Trubiani. 2021. Uncertainty-aware exploration in model-based testing. In *14th IEEE Conference on Software Testing, Verification and Validation*. IEEE, 71–81.

[21] Hao Chen. 2019. Sequential change-point detection based on nearest neighbors. *The Annals of Statistics* 47, 3 (2019), 1381–1407.

[22] Yuqi Chen, Christopher Poskitt, and Jun Sun. 2018. Learning from mutants: Using code mutation to learn and monitor invariants of a cyber-physical system. In *Proceedings of the 39th IEEE Symposium on Security and Privacy*. IEEE, 648–660.

[23] Martial Coulon, Marie Chabert, and Ananthram Swami. 2008. Detection of multiple changes in fractional integrated ARMA processes. *IEEE Transactions on Signal Processing* 57, 1 (2008), 48–61.

[24] Mário de Castro and Ignacio Vidal. 2019. Bayesian inference in measurement error models from objective priors for the bivariate normal distribution. *Statistical Papers* 60, 4 (2019), 1059–1078.

[25] Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems: 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings 14*. Springer,

337–340.

[26] Jens Ehlers, André van Hoorn, Jan Waller, and Wilhelm Hasselbring. 2011. Self-adaptive software system monitoring for performance anomaly localization. In *Proceedings of the 8th ACM International Conference on Autonomic Computing*. 197–200.

[27] Naeem Esfahani, Ehsan Kouroshfar, and Sam Malek. 2011. Taming uncertainty in self-adaptive software. In *Proceedings of the 19th ACM SIGSOFT Symposium on the Foundations of Software Engineering and 13th European Software Engineering Conference*. 234–244.

[28] Oreste Fecarotta, Riccardo Martino, and Cristina Morani. 2019. Wastewater pump control under mechanical wear. *Water* 11, 6 (2019), 1210.

[29] Antonio Filieri, Henry Hoffmann, and Martina Maggio. 2014. Automated design of self-adaptive software with control-theoretical formal guarantees. In *Proceedings of the 36th International Conference on Software Engineering*. 299–310.

[30] Antonio Filieri, Henry Hoffmann, and Martina Maggio. 2015. Automated multi-objective control for self-adaptive software design. In *Proceedings of the 10th Joint Meeting on Foundations of Software Engineering*. 13–24.

[31] Antonio Filieri, Martina Maggio, Konstantinos Angelopoulos, Nicolas d'Ippolito, Ilias Gerostathopoulos, Andreas Berndt Hempel, Henry Hoffmann, Pooyan Jamshidi, Evangelia Kalyvianaki, Cristian Klein, et al. 2015. Software engineering meets control theory. In *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, 71–82.

[32] John Franks. 1985. Nonsingular Smale flows on S3. *Topology* 24, 3 (1985), 265–282.

[33] David Garlan, S-W Cheng, A-C Huang, Bradley Schmerl, and Peter Steenkiste. 2004. Rainbow: Architecture-based self-adaptation with reusable infrastructure. *Computer* 37, 10 (2004), 46–54.

[34] Eric Bernd Gil, Ricardo Caldas, Arthur Rodrigues, and et al. 2021. Body Sensor Network: A Self-Adaptive System Exemplar in the Healthcare Domain. In *16th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, 224–230.

[35] Jorn Gruber, Daniel Ramirez, Teodoro Alamo, and Eduardo Camacho. 2011. Min–Max MPC based on an upper bound of the worst case cost with guaranteed stability. *Journal of Process Control* 21, 1 (2011), 194–204.

[36] Fredrik Gustafsson and Fredrik Gustafsson. 2000. *Adaptive filtering and change detection*. Vol. 1. Citeseer.

[37] Edward J Hannan and Barry G Quinn. 1979. The determination of the order of an autoregression. *Journal of the Royal Statistical Society: Series B (Methodological)* 41, 2 (1979), 190–195.

[38] Farshad Harirchi, Zheng Luo, and Necmiye Ozay. 2016. Model (in) validation and fault detection for systems with polynomial state-space models. In *2016 American Control Conference (ACC)*. IEEE, 1017–1023.

[39] Farshad Harirchi and Necmiye Ozay. 2018. Guaranteed model-based fault detection in cyber–physical systems: A model invalidation approach. *Automatica* 93 (2018), 476–488.

[40] Michiel Hazewinkel and Jan C Willems. [n. d.]. Stochastic Systems: The Mathematics of Filtering and Identification and Applications. ([n. d.]).

[41] Zhijian He, Yao , Enyan Huang, Qixin Wang, Yu Pei, and Haidong Yuan. 2019. A system identification based oracle for control-cps software fault localization. In *Proceedings of the 41st International Conference on Software Engineering*. IEEE, 116–127.

[42] Joseph Hellerstein, Yixin Diao, Sujay Parekh, and Dawn Tilbury. 2004. *Feedback control of computing systems*. Wiley Online Library.

[43] Rolf Isermann. 1990. Estimation of physical parameters for dynamic processes with application to an industrial robot. In *1990 American control conference*. IEEE, 1396–1401.

[44] Rolf Isermann. 2013. *Digital control systems*. Springer Science & Business Media.

[45] Michael Johnson and Mohammad Moradi. 2005. *PID control*. Springer.

[46] Jürgen Jost. 2005. *Dynamical systems: examples of complex behaviour*. Springer Science & Business Media.

[47] Cristian Klein, Martina Maggio, Karl-Erik Årzén, and Francisco Hernández-Rodriguez. 2014. Brownout: Building more robust cloud applications. In *Proceedings of the 36th International Conference on Software Engineering*. 700–711.

[48] YD Landau. 1981. Deterministic and Stochastic Model Reference Adaptive Control: A unified presentation of model reference adaptive controllers and stochastic self-tuning regulators. In *Stochastic Systems: The Mathematics of Filtering and Identification and Applications: Proceedings of the NATO Advanced Study Institute held at Les Arcs, Savoie, France, June 22–July 5, 1980*. Springer, 387–419.

[49] Dan Ling, Ying Zheng, Hong Zhang, Weidong Yang, and Bo Tao. 2017. Detection of model-plant mismatch in closed-loop control system. *Journal of Process Control* 57 (2017), 66–79.

[50] Tao Ma, Shaukat Ali, Tao Yue, and Maged Elaasar. 2019. Testing self-healing cyber-physical systems under uncertainty: a fragility-oriented approach. *Software Quality Journal* 27 (2019), 615–649.

[51] Martina Maggio, Alessandro Vittorio Papadopoulos, Antonio Filieri, and Henry Hoffmann. 2017. Automated control of multiple software goals using multiple actuators. In *Proceedings of the 11th Joint Meeting on Foundations of Software*

Engineering. 373–384.

[52] Claudio Mandrioli and Martina Maggio. 2020. Testing self-adaptive software with probabilistic guarantees on performance metrics. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1002–1014.

[53] Douglas Montgomery. 2020. *Introduction to statistical quality control.* John Wiley & Sons.

[54] Gabriel A Moreno, Javier Cámara, David Garlan, and Bradley Schmerl. 2015. Proactive self-adaptation under uncertainty: a probabilistic model checking approach. In *Proceedings of the 10th Joint Meeting on Foundations of Software Engineering*. 1–12.

[55] Gabriel A Moreno, Alessandro Vittorio Papadopoulos, Konstantinos Angelopoulos, Javier Cámara, and Bradley Schmerl. 2017. Comparing model-based predictive approaches to self-adaptation: CobRA and PLA. In *12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, 42–53.

[56] Nhan T Nguyen and Nhan T Nguyen. 2018. *Model-reference adaptive control.* Springer.

[57] Fabrizio Pastore and Leonardo Mariani. 2015. ZoomIn: Discovering failures by detecting wrong assertions. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 1. IEEE, 66–76.

[58] Friedrich Pukelsheim. 1994. The three sigma rule. *The American Statistician* 48, 2 (1994), 88–91.

[59] Yi Qin, Tao Xie, Chang Xu, Angello Astorga, and Jian Lu. 2019. CoMID: Context-Based Multiinvariant Detection for Monitoring Cyber-Physical Software. *IEEE Transactions on Reliability* 69, 1 (2019), 106–123.

[60] Sayan Basu Roy, Shubhendu Bhasin, and Indra Narayan Kar. 2017. Combined MRAC for unknown MIMO LTI systems with parameter convergence. *IEEE Trans. Automat. Control* 63, 1 (2017), 283–290.

[61] Farhana Sadia, Sarah Boyd, and Jonathan M Keith. 2018. Bayesian change-point modeling with segmented ARMA model. *Plos one* 13, 12 (2018), e0208927.

[62] Mazeiar Salehie and Ladan Tahvildari. 2012. Towards a goal-driven approach to action selection in self-adaptive software. *Software: Practice and Experience* 42, 2 (2012), 211–233.

[63] Amritash Shekhar and Abhijeet Sharma. 2018. Review of model reference adaptive control. In *2018 international conference on information, communication, engineering and technology (ICICET)*. IEEE, 1–5.

[64] Stepan Shevtsov, Mihaly Berekmeri, Danny Weyns, and Martina Maggio. 2017. Control-theoretical software adaptation: A systematic literature review. *IEEE Transactions on Software Engineering* 44, 8 (2017), 784–810.

[65] Stepan Shevtsov, M Usman Iftikhar, and Danny Weyns. 2015. SimCA vs ActivFORMS: comparing control-and architecture-based adaptation on the TAS exemplar. In *Proceedings of the 1st international workshop on control theory for software engineering*. 1–8.

[66] Stepan Shevtsov and Danny Weyns. 2016. Keep it simplex: Satisfying multiple goals with guarantees in control-based self-adaptive systems. In *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. 229–241.

[67] Stepan Shevtsov, Danny Weyns, and Martina Maggio. 2019. SimCA*: A Control-theoretic approach to handle uncertainty in self-adaptive systems with guarantees. *ACM Transactions on Autonomous and Adaptive Systems* 13, 4 (2019), 1–34.

[68] Hui Song, Amit Raj, Saeed Hajebi, Aidan Clarke, and Siobhán Clarke. 2013. Model-based cross-layer monitoring and adaptation of multilayer systems. *Science China Information Sciences* 56, 8 (2013), 1–15.

[69] Yanxiang Tong, Yi Qin, Yanyan Jiang, Chang Xu, Chun Cao, and Xiaoxing Ma. 2021. Timely and accurate detection of model deviation in self-adaptive software-intensive systems. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 168–180.

[70] Guojun Wang and Siva Sivaganesan. 2013. Objective priors for parameters in a normal linear regression with measurement error. *Communications in Statistics-Theory and Methods* 42, 15 (2013), 2694–2713.

[71] Qiang Wang, Xinlei Zheng, Jiyong Zhang, and Joseph Sifakis. 2022. A hybrid controller for safe and efficient longitudinal collision avoidance control. *Journal of Systems Architecture* 125 (2022), 102432.

[72] Shu Wang, Henry Hoffmann, and Shan Lu. 2022. AgileCtrl: a self-adaptive framework for configuration tuning. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 459–471.

[73] Danny Weyns and Radu Calinescu. 2015. Tele assistance: A self-adaptive service-based system exemplar. In *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, 88–92.

[74] H Philip Whitaker, Joseph Yamron, and Allen Kezer. 1958. *Design of model-reference adaptive control systems for aircraft.* Massachusetts Institute of Technology, Instrumentation Laboratory.

[75] Wenhua Yang, Chang Xu, Yepang Liu, Chun Cao, Xiaoxing Ma, and Jian Lu. 2014. Verifying self-adaptive applications suffering uncertainty. In *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*. 199–210.

[76] Dan Zhang and Bin Wei. 2017. A review on model reference adaptive control of robotic manipulators. *Annual Reviews in Control* 43 (2017), 188–198.

[77] Wanrong Zhang and Yajun Mei. 2022. Bandit change-point detection for real-time monitoring high-dimensional data under sampling control. *Technometrics* (2022), 1–11.

[78] Xiaoyun Zhu, Mustafa Uysal, Zhikui Wang, Sharad Singhal, Arif Merchant, Pradeep Padala, and et al. 2009. What does control theory bring to systems research? *ACM SIGOPS Operating Systems Review* 43, 1 (2009), 62–69.