Timely and Accurate Detection of Model Deviation in Self-Adaptive Software-Intensive Systems

Yanxiang Tong
State Key Lab for Novel Software Technology, Nanjing University
Nanjing, China
tongyanxiang@gmail.com

Yi Qin*
State Key Lab for Novel Software Technology, Nanjing University
Nanjing, China
yiqincs@nju.edu.cn

Yanyan Jiang
State Key Lab for Novel Software Technology, Nanjing University
Nanjing, China
jyy@nju.edu.cn

Chang Xu
State Key Lab for Novel Software Technology, Nanjing University
Nanjing, China
changxu@nju.edu.cn

Chun Cao*
State Key Lab for Novel Software Technology, Nanjing University
Nanjing, China
caochun@nju.edu.cn

Xiaoxing Ma
State Key Lab for Novel Software Technology, Nanjing University
Nanjing, China
xxm@nju.edu.cn

ABSTRACT

Control-based approaches to self-adaptive software-intensive systems (SASs) are hailed for their optimal performance and theoretical guarantees on the reliability of adaptation behavior. However, in practice the guarantees are often threatened by model deviations occurred at runtime. In this paper, we propose a Model-guided Deviation Detector (MoD2) for timely and accurate detection of model deviations. To ensure reliability, a SAS can switch a control-based optimal controller for a mandatory controller once an unsafe model deviation is detected. MoD2 achieves both high timeliness and high accuracy through a deliberate fusion of parameter deviation estimation, uncertainty compensation, and safe region quantification. Empirical evaluation with three exemplar systems validated the efficacy of MoD2 (93.3% shorter detection delay, 39.4% lower FN rate, and 25.2% lower FP rate), as well as the benefits of the adaptation-switching mechanism (abnormal rate dropped by 29.2%).

CCS CONCEPTS

• **Software and its engineering** → **Software verification and validation**; • **Social and professional topics** → Software selection and adaptation.

KEYWORDS

Self-Adaptive Software, Control Theory, Model Deviation

ACM Reference Format:
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 European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '21), August 23–28, 2021, Athens, Greece.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3468264.3468548

*Corresponding author.

1 INTRODUCTION

Control theory has been increasingly adopted in developing self-adaptive software-intensive systems [56, 59]. In designing a control-based self-adaptive system (control-SAS for short), developers first identify a nominal model (e.g., a linear time-invariant model) of the subject system (a.k.a. the managed system, or the *plant* in control jargon), and then design a managing system (a.k.a. the *controller* in control jargon) based on the identified model with some (feedback) control mechanisms such as Proportional-Integral-Derivative control or Model Predictive Control. The advantage of this approach is that the control mechanisms have well-established mathematical foundations that theoretically guarantee the optimality and reliability of the resulting SAS [25, 47, 65].

However, software-intensive systems behave differently from physical devices to which control theory is usually applied. The behavioral patterns of the former are more dynamic and uncertain than the latter. During its execution, a managed software-intensive system's behavior often deviates from the identified nominal model [10, 25, 49]. This *model deviation*, if goes beyond a certain region, will invalidate the theoretical guarantees and threaten the safety of the whole system [9, 17, 46, 65].

Such problem of model deviation has been acknowledged in the literature, but to our knowledge no satisfactory solution was given [10, 11, 23, 44, 49, 60]. Using robust controllers [11, 44] can mitigate the problem by tolerating slight deviations at the cost of less optimality and more complexity in controller design, but the problem itself remains. Some authors proposed to monitor system outputs and rebuild the controller through system re-identification once the outputs violate some specified criteria [10, 23, 49]. However, this approach is less sensitive in that model deviations could have happened much earlier than their manifestation as abnormal outputs. It is crucial to report dangerous deviations as early as possible before they cause disastrous consequences. Sliding window-based model deviation detection approaches [12, 23, 35] require extensive domain knowledge and human efforts to tune their parameters. The emerging learning-based approaches [17, 57] reduce the dependence on domain knowledge and human efforts, but are not speedy and accurate enough, as to be shown in Section 5.

An alternative way is to monitor the managed system's model parameter values directly, which could save the time that a model deviation propagates to the managed system's abnormal output. Since the nominal model's parameter values are unobservable, one have to calculate these values based on the observable inputs and outputs. However, both internal uncertainty (e.g., process noise) and external uncertainty (e.g., measurement error) [19, 64] would lead to an inaccurate calculation. Sliding window could alleviate the impact of measurement error to some extent, but it is less effective in handling slight model deviation, and still faces the problem of manually tuned parameters.

In this paper, we propose the Model-guided Deviation Detector (MoD2) to support timely and accurate model deviation detection for control-SASs. The key intuition of MoD2 is to *estimate* the nominal model's parameter values. Unlike most existing works that treat the nominal model's parameter as a deterministic variable, MoD2 describes the parameter as a stochastic variable. The identified value is now considered as the parameter's mean value, and we additionally identify the parameter's variance. By leveraging the knowledge of the parameter's distribution, MoD2 uses Bayesian estimation to achieve an effective estimation of its value. MoD2 also integrates two lightweight techniques, namely uncertainty compensation and safe region quantification, to further improve its accuracy without sacrificing too much timeliness.

Based on MoD2, an adaptation-supervision mechanism is implemented to alleviate the impact of model deviation with a dual-track adaption strategy. The mechanism uses a supervision loop to guard the adaptation loop of a control-SAS. Once MoD2 detects model deviation, our mechanism will switch the control theory-based optimal controller for a mandatory controller that ensures the mandatory requirements but may scarify some system utilization. When model deviation disappears, the SAS can switch back to the optimal controller again.

We evaluate our approach on three representative exemplar SAS systems, namely, SWaT [8], RUBiS [18], and video encoder [50]. We compare MoD2's performance with two baseline approaches (i.e., a sliding window-based detector [23], and an SVM-based detector [17]). The results show the effectiveness of MoD2 which achieves 93.3% shorter detection delay, 39.4% lower FN rate, and 25.2% lower FP rate, as well as the usefulness of MoD2-based adaptation-supervision mechanism by reporting a 29.2% lower abnormal rate.

In the remainder of this paper, section 2 gives the necessary background and discusses the motivation of our work. Then, section 3 introduces the adaptation-supervision mechanism, and section 4 details our MoD2 with three main techniques. Next, section 5 shows the experimental evaluation of our approach. Finally, section 6 discusses the related work, and section 7 concludes the paper.

2 BACKGROUND AND MOTIVATION

In this section, we first introduce the background of control-SASs, with a motivating example of Secure Water Treatment testbed (SWaT for short) [8]. Then we discuss the impact of model deviation on self-adaptive systems as the limitation of existing approaches in handling model deviation. Next, we introduce Bayesian estimation and Kalman filter for estimating model parameter values. We then discuss our scope and assumptions in addressing the problem of model deviation.

2.1 Control-based self-adaptive systems

In recent years, control-based self-adaptive systems [16, 23, 25, 26, 56, 59, 63] became a research hotspot for its less burden on the developers' mathematical and software knowledge to design ad-hoc self-adaptation solutions. Filieri et al. first propose a methodology (i.e., the push-button method [23]) to implement control-SASs, which includes a systematic data collection and model fitting procedure (a.k.a., *system identification* [48]) and a control theory-guided controller designing procedure (a.k.a., *controller synthesis*). The identification procedure enables the automatic construction of an approximate model for the managed system, and the synthesized controller provides theoretical guarantees to the derived managing system's behavior in controlling adaptation.

Specifically, in system identification, control-SASs assume that the managed system's behavior can be captured by a quantitative *nominal model* to improve productivity and cope with infinite kinds of environmental dynamics. A *nominal model* describes the relationship between the managed system's output, the controller's output, and the environmental input. Many types of nominal models [52] have been proposed to support self-adaptation in various scenarios, including linear time-invariant system [4, 35, 61], linear time-varying system[37], and nonlinear time-invariant system [11].

In controller synthesis, control-SASs leverage various control-theoretical techniques to design the optimal controllers for various scenarios. Two of the most prevailing techniques are proportional-integral-derivative (PID) control [24, 44, 60], and model predictive control [4, 5, 49, 51]. A controller's workflow resembles the conventional self-adaptation mechanism that consists of continuous *adaptation loops* of monitoring, analyzing, planning, and executing. A controller's behavior should be subject to the control properties, for example, the SASO properties (i.e., stability, accuracy, settling time, and overshoot) [36].

Here, we use the SWaT system to explain the concepts of control-SASs. SWaT is a fully operational scaled-down water treatment plant that produces doubly-filtered drinking water. SWaT consists of five water-processing tanks, as well as the pipes connecting those tanks. The in-coming valve and out-going valve of each tank can be controlled remotely. The objective of a self-adaptive SWaT system is to enable safe (e.g., no overflow or underflow in any of the tanks) and efficient (e.g., maximum clean water production) water filtering under different environmental situations (e.g., the initial water level of the five tanks and the in-coming water flow of the first tank).

To build such a self-adaptive SWaT system following the control-SASs methodology, we can use the following linear time-invariant system to describe the SWaT in system identification.

$$\begin{cases} \vec{x}(k) = A \cdot \vec{x}(k-1) + B \cdot \vec{u}(k-1) \\ y(k) = C \cdot \vec{x}(k) \end{cases} \tag{1}$$

where (k) denotes a serial number of the adaption loop. For the variables, $\vec{x}(k)$ describes SWaT's current running state (i.e., the in-coming and out-going water of the five tanks), $\vec{x}(k-1)$ describes the historical state, $y(k)$ describes the system's output value (i.e., the measured water levels of five tanks), and $\vec{u}(k-1)$ describes its received control signal (i.e., the valves' opening time in an adaptation loop).

The parameters in Equation 1, including A, B, and C, are identified for the system since their values cannot be measured directly. Each of these parameters represents a specific dynamical feature of SWaT. Specifically, A represents the system's delay property, which describes the interval between the time when a controller requests to turn on/off a valve and the time when the valve is turned on/off. B represents the system's controllability, which describes the influence of turning on/off a valve upon the system's running state. C represents the system's observability, which describes the mapping between each tank's in-coming and out-going water flow and the measured water level.

Based on the identified parameter values, SWaT's developers provide a suite of well-designed controllers to enable the system's adaptation to different environmental situations. These controllers leverage both control-theory and rule-based domain knowledge to guide the control strategies of various valves. Concretely, there are a total of six controllers. Four control the in-coming and out-going valves of one or two tanks, respectively. Two controllers control the general water treatment procedure and coordinate with the other four.

Control theory guarantees the self-adaptive SWaT system subjecting to the control properties. For example, the water in all tanks should neither overflow nor underflow (i.e., subject to no overshoot property). Meanwhile, the water level in all tanks should reach the desired level quickly (i.e., subjecting to low settling time property), in order to maximize SWaT's clean water production.

2.2 Model deviation in control-based self-adaptive systems

Model deviation undermines control-SASs. Model deviation is the mismatch between the nominal model's identified parameter values and its actual values at runtime. The occurrence of model deviation may invalidate the theoretical guarantees provided by control theory and could potentially cause the abnormal behavior of a control-SAS.

Let us consider two real-world cases of model deviation in SWaT. The first one is reported by Adepu et al. that network attacks could cause SWaT's abnormal behavior [1]. For instance, if the controller's signal of turning on a valve is blocked or tampered with, the valves' opening time will be changed accordingly. Then, when the controlled tank's water level exceeds an alarming level, the corresponding controller will still control the valves according to the unchanged B value and cause tank overflow.

In the second case, the physical condition of SWaT's pipes and valves will also lead to model deviation. For example, physical abrasion [21] may wear the valves and result in the change of water flow rate (i.e., system's controllability feature B). When a controller still controls the valves based on the unchanged B value, the water flow into the tank will be smaller than what the controller expected and may cause the tank to underflow.

Lots of self-adaptive system researchers acknowledge the existence of model deviation, and propose different solutions. In push-button-method [23], Filieri et.al first identify model deviation, address it by monitoring system outputs, and rebuild the controller through system re-identification. Some other works [10, 24, 49] also rely on re-identification to handle model deviation. Some researchers focus on improving the controller's robustness to tolerate slight deviations [11, 44].

These solutions have their own limitations. For the robust strengthening approaches, model deviation does cause the controller designed by field experts to behave abnormally, in our previous motivating example of SWaT. We cannot simply assume that developer-designed controllers could overcome model deviation as in [11, 44]. For the re-identification approaches, it is too slow to be effective since model deviations could have happened much earlier than their manifestation as abnormal outputs (Kang et.al report that it takes around 5 minutes for a detected attack to fail SWaT's execution [42]). The latency between model deviation's occurrence and its observable consequences make it crucial to report dangerous deviations as early as possible.

As we discussed in Section 1, the challenge is to balance the detection's timeliness and accuracy. Existing works fail to balance this in detecting model deviation. For the sliding window-based approaches that have been exploited long by self-adaptive researchers [12, 23, 35], their performance largely depends on manual or empirical settings (e.g., size of the sliding window-based and the threshold for monitored system's output values). It often requires extensive domain and mathematical knowledge for an appropriate setting, which is impossible for non-experts. For those newly-proposed learning-based approaches [17, 57], they usually combine multiple test results for improving their detection accuracy, which prolongs their detection delay. What's more, these approaches often require binary training sets, the positive instances of which can hardly be prepared since the managed system's behavior under model deviation is unpredictable.

Our MoD2 addresses the challenge above following the guidance of the nominal model. First, MoD2 fundamentally decreases its detection delay by deriving the nominal model's parameter values with Bayesian estimation. Second, MoD2 improves its estimation accuracy and detection accuracy with two supporting techniques, namely, uncertainty compensation and safe region quantification. Third, MoD2 further reduces the time overhead of its estimation and detection by Kalman filter and probability-based quantification, respectively.

2.3 Bayesian estimation and Kalman filter

Bayesian estimation [29] enables one to estimate an unobservable parameter value (the value of $B(k)$ in our case). To improve its accuracy, the approach requires a priori information about the parameter's distribution (i.e., mean value and variance of all collected B values in system identification). Bayesian estimation works in an iterative manner. In each iteration, we first compute a *prior distribution* of the concerned parameter that matches the past observations and the identified distribution. Then, the prior is combined with the current observations to obtain a *posterior distribution* which is reported as the current estimation value.

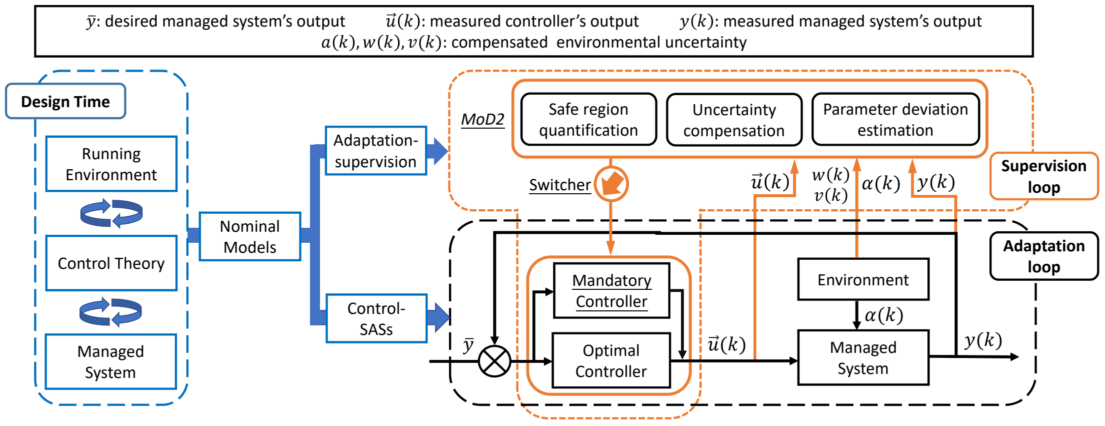

Figure 1: Overview of MoD2-based adaptation-supervision mechanism

In Bayesian estimation, one has to calculate the prior and posterior distributions in every iteration, which brings a considerable overhead. Kalman filter reduces this overhead by deriving the current prior and posterior distributions through updating the previous distributions. Specifically, Kalman filter uses a recursive factor K (also called Kalman gain) to approximate the relationship of the prior/posterior distributions from two or multiple consecutive iterations. When such relationship is linear or can be approximate linearly, Kalman filter could produce an accurate and fast estimation of the concerned parameter.

2.4 Our scope and assumptions

Due to the various application scenarios of self-adaptive systems, the proposed MoD2 has its scope and assumptions for ease of discussion and presentation. We delineate our scope based on existing work and inherit pre-existing assumptions from other self-adaptation or control theory research. Notice we only give our assumptions and clarify their realistic here. We will discuss MoD2's limitations brought by these assumptions in Section 4.4.

Nominal model. In this work, we focus on linear time-invariant system (LTI system for short). LTI system is one of the most widely-used models among self-adaptation researchers [35, 49, 60], and has been successfully applied in different subjects [4, 61].

Model deviation. We focus on two types of model deviation, discrete behavior and inaccurate environmental interaction, following the two reported cases in our motivating example of SWaT. One occurs when a controller produces its control signals (e.g., block or tamper with the instruction to open the valves) and the other occurs when the managed system receives the control signals (e.g., water flow changing due to physical abrasion).

Based on the two types of model deviation, in this work, we focus on the deviation of the managed system's controllability features (i.e., the parameter B) and assume that the managed system's delay and observability features (i.e., parameter A and C) are comparatively stable. Recent literatures [1, 2, 41] report and validate the deviation of the controllability features.

We also assume that $B(i)$ is a unary parameter for ease of presentation. Nevertheless, our approach can be applied to a multinary parameter.

Environmental uncertainty. Environmental uncertainty is the *known unknown* information at runtime [4, 20]. As a result, one has to model the target uncertainty first before addressing it. In this paper, we assume that the environmental uncertainty can be described by a linear time-invariant model or a normal distribution. The former is a widely used approach in describing uncertain environmental input by the control theory [13, 62], and the latter is a typical setting for describing uncertainty's influence among self-adaptive systems [19, 64].

3 AN ADAPTATION-SUPERVISION MECHANISM

To address the problem of model deviation, this paper proposes an adaptation-supervision mechanism to supervise the execution of control-SASs as shown in Figure 1. Intuitively, we added a new parallel *supervision loop* to a control-SAS. The supervisor continuously monitors the adaptation loop. When model deviation is detected, the supervisor immediately "falls back" to a safer (but less efficient) mandatory controller and recovers to the main control loop after successful model re-identification.

The supervisor consists of three major components: a model-guided deviation detector (MoD2, which is this paper's primary focus), a mandatory controller, and a switcher. Underlined blocks in Figure 1 denote these newly added components.

In the adaption-supervision mechanism, we first require the system designer to provide a *mandatory controller* that guarantees minimal system functionality (i.e., mandatory requirements) even on nominal-model deviation. Mandatory controllers have been extensively studied by the community of control-SASs, e.g., by adopting architectural-guided [27] or goal-driven rules [58].

With both control theory-based optimal controller and mandatory controller, the *switcher* coordinates them on the MoD2. The switcher immediately changes the adaptation loop from using the optimal controller to using the mandatory controller on model deviation to reduce the unpredictable influences to a minimum. The switcher is also responsible for consistently conducting model re-identification and turning back to the optimal controller when the abnormal situation has disappeared (e.g., the network attack of

SWaT was blocked by a firewall). The switcher can be implemented following the push button method [23].

Therefore, the design of an efficient and effective model-deviation detector (our MoD2) should be the key point.

4 MOD2: MODEL-GUIDED DEVIATION DETECTOR

The three major technical designs (and contributions) of the proposed MoD2 are:

(1) MoD2 *directly estimates the nominal model's parameter values* by modeling the nominal model's parameter B as a *time-variant stochastic variable* $B(k)$. In contrast with existing work, which is mostly based on the observable outputs $y(k)$, this white-box approach accelerates the model-deviation detection and improves the accuracy.

(2) MoD2 *compensates environmental uncertainty* in the nominal model by introducing compensation terms to reduce the consequences of measurement errors, yielding improved model-deviation estimation accuracy. MoD2 adopts the Kalman filter, a fast Bayesian estimator, to accelerate its estimation on the parameter values.

(3) MoD2 *directly quantifies the nominal model's theoretical safe region* without the need for calibrating an empirical threshold based on execution traces of the managed system. By using the necessary conditions under which model deviation can make the controller behave abnormally, MoD2 directly (and thus timely) reports model deviation with sufficient probabilistic confidence.

These technical details are elaborated as follows.

4.1 Parameter deviation estimation

We first describe the concerned model parameter B as a time-variant stochastic variable $B(k)$:

$$B(k) = B(k-1) + \omega \cdot (\tilde{B} - B(k-1)) + q(k) \tag{2}$$

where: (1) The parameter's deviation is described as the sudden changes of centripetal force ω that drive $B(k)$ away from identified $\tilde{B}$. (2) The parameter's inherent fluctuation is described as a random walk procedure, the step of which subjects to a normal distribution $q(k)$. If the system is free from model deviation (i.e., $\omega = 0$), the variance of $q(k)$ equals to the variance of $B(k)$, which can be acquired along with the identification of $\tilde{B}$.

Combining Equation 1 and Equation 2 enables MoD2 to estimate the value of $B(k)$ in a Bayesian approach. Specifically, in the i-th adaptation loop, a prior distribution of $B(i)$ is computed that fits the previous posterior distribution $B(i-1)$ and identified variance of $q(k)$, according to Equation 2. Then, MoD2 calibrates the posterior distribution of $B(i)$ by updating the prior distribution of $B(i)$ with the current observation (i.e., $\vec{x}(i-1)$, $\vec{u}(i-1)$, and $y(i)$), according to Equation 1.

Note that we might not be able to observe the value of system state $\vec{x}(i)$ directly in some cases. We also use Bayesian estimation to estimate $\vec{x}(i)$'s value. In this situation, each iteration of Bayesian estimation has two steps. MoD2 first estimates $\vec{x}(i-1)$'s value by treating the previous estimated $B(i-1)$ as an observation. Then, the obtained $\vec{x}(i-1)$ is used to estimate the current parameter value $B(i)$.

4.2 Uncertainty compensation

Due to various environmental uncertainties, a direct estimation of $B(i)$'s value is usually inaccurate. To reduce uncertainty-based errors in the estimations of $B(i)$, we refine the nominal model by compensating for environmental uncertainty and accelerate our estimation with the Kalman filter.

Using compensation terms to refine a nominal model is a widely-used technique in the control theory community [30]. However, one has to carefully design the effective compensation terms. Here, we introduce three different compensation terms. One is the measured environmental input $a(k)$, which can be used to posterior calibrate our estimated $B(i)$'s value. We use a linear time-invariant model to describe the influence brought by environmental input. The other two compensation terms, $w(k)$ and $v(k)$, capture the influences of measurement errors on the managed system's internal state and external behavior, respectively. According to our assumptions, we depict $w(k)$ and $v(k)$ as normal distribution of variance W and V.

Then, the nominal model (Equation 1) is refined with three compensation terms, as follows.

$$\begin{cases} \vec{x}(k) = A \cdot \vec{x}(k-1) + B(k) \cdot \vec{u}(k-1) + \gamma \cdot a(k) + w(k) \\ y(k) = C \cdot \vec{x}(k) + v(k) \end{cases} \tag{3}$$

where γ is the coefficient of the linear model that describes $a(k)$. The value of γ which is acquired during system identification along with other model parameters.

Based on Equation 3, MoD2 further alleviates the impact of measurement error by transforming the nominal model to difference equations. In the original nominal model, the measurement error on parameter values (e.g., A) is multiplied by other variables (e.g., $\vec{x}(k-1)$). As a result, a large variable value would amplify the measurement error and make the following estimation less accurate. MoD2 uses the difference equation to address the impact of this multiplier effect. If the managed system's behavior is free from severe changes, a variable's values in two consecutive adaptation loops should not differ a lot [14, 55], and the difference equation can thus alleviate the multiplier effect. Otherwise, it could be easier for us to detect. In this paper, we use $\Delta var(k)$ to denote the difference between variable var's values in the k-th and $(k-1)$-th adaptation loop (i.e., $\Delta var(k) = var(k) - var(k-1)$). Thus the difference equations of the nominal model are listed as follows.

$$\begin{cases} \Delta\vec{x}(k) = A \cdot \Delta\vec{x}(k-1) + B(k) \cdot \Delta\vec{u}(k-1) + \gamma \cdot \Delta a(k) + w(k) \\ \Delta y(k) = C \cdot \Delta\vec{x}(k) + v(k) \end{cases} \tag{4}$$

Notice that if the controller's control signal keeps unchanged, the difference equations are ineffective since the estimated B will be eliminated by multiplying zero ($\Delta\vec{u}(k-1) = \vec{u}(k) - \vec{u}(k-1) = 0$). In this situation, we use the original nominal model to estimate $B(k)$'s value instead of the difference equations.

With those compensation terms introduced, it would be extremely time-consuming to perform the original Bayesian estimation. MoD2 accelerates its estimation by using Kalman filter [43, 45], a more efficient linear quadratic estimation approach. Conceptually, Kalman filter works by updating the prior distribution of $B(i)$ with the latest observation only (i.e., $\vec{x}(i-1)$, $\vec{u}(i-1)$, and $y(i)$), instead of re-calculating the priori with all historical observations (i.e., $\vec{x}(0)$–$\vec{x}(i-1)$, $\vec{u}(0)$–$\vec{u}(i-1)$, and $y(0)$–$y(i)$). MoD2's Kalman filter parameter estimation also follows an iterative manner as the original Bayesian estimation. However, it can give a more accurate normal distribution of the estimated parameters by filtering the effects of compensation terms (i.e., $a(i)$, $w(i)$, and $v(i)$).

4.3 Safe region quantification

Once the distribution of $B(i)$ is estimated, MoD2 has to check whether it suffers model deviation or not. We combine the nominal model listed in Equation 4 with a theoretical safe region and use a cumulative distribution function to calculate the probability that $B(i)$'s actual value exceeds the safe region. Unlike existing works that use a manual or empirical threshold, MoD2's safe region represents the necessary condition under which the controller's behavior is guaranteed by control theory. As a result, MoD2 would distinguish the normal executions from the abnormal ones even though the executions have not been empirically explored and makes its detection more accurate.

It is the control theory, which helps developers design the original controllers, that shapes the safe regions for the derived controllers. A controller's safe region can be captured by either formal analysis or experimental study. For PID and MPC controllers, their safe region can be theoretically derived by frequency response analysis [38, 40] and linear matrix inequality [31], respectively. How to capture a theoretical safe region is out of our scope. If it is extremely hard to capture the theoretical safe region of a complex controller, we will turn back to empirical study or statistic analysis on the execution traces. Formally, parameter B's safe region Θ_B can be defined as the interval between a lower bound Θ_B^L and an upper bound Θ_B^U.

Given a controller's safe region Θ_B, $B(i)$'s cross-border detection can still be inaccurate due to the estimation error of $B(i)$'s value. Most existing works combine possible results in different adaptation loops to improve their accuracy, which inevitably delays their detection. MoD2's estimated distributions of the parameter value naturally combines the past observations and avoid using multi-time detections.

MoD2 uses a probability-based approach to detect B's deviation with a confidence interval CI. The probability that $B(i)$'s value falls within the safe region $[\theta_B^L, \theta_B^U]$ can be calculated by a cumulative distribution function, as shown in Equation 5, where $f_{B(k)}(x)$ is $B(k)$'s probability density function. Particularly, the estimated $B(i)$'s distribution in our case is subject to a normal distribution (i.e., $B(i) \sim N(\mu(i), P(i))$). If the derived probability $\hat{p}(i)$ exceeds a confidence interval CI, MoD2 will report model deviation and trigger the switching of adaptation.

$$\hat{p}(k) = \int_{\theta_B^U}^{\theta_B^L} f_{B(k)}(x)dx \tag{5}$$

MoD2 also uses an active detector to handle the situation when the optimal controller produces no control signal by using hypothesis testing to reveal possible model deviation. We use normal distribution to depict the variation properties (e.g., residuals) of the managed system's output when no control signal comes and detect model deviation by checking whether the measured variation is consistent with the pre-identified distribution.

4.4 Discussions on MoD2

Assumptions and limitations. In Section 2.4 we listed our assumptions in MoD2 design. Now, we discuss the limitations brought by these assumptions, as well as the possible solutions to free our approach from these assumptions.

First, we assume that the managed system's nominal model is described as a LTI model. Our approach can be directly extended to other types of nominal models since the three major technical designs of MoD2 are independent of the types of the nominal model. Other types of models might make Kalmar filter less effective due to their more complicated forms. However, extended Kalman filter [43] would alleviate this by supporting non-linear or time-variant nominal models.

Second, we assume that the deviated parameter is the controllability parameter (i.e., B) only, and the parameter has a unary description. Since our parameter estimation on $B(k)$ requires no more than the identified value of the other two parameters (i.e., A and C), MoD2 can also be extended to detect model deviation on one of these two parameters. However, detecting model deviation on multiple parameters is challenging and remains open. As we explained previously that B's unary description is for the ease of discussion only, MoD2 can be applied to a multi-unary parameter. In fact, in our later experiments, we already run MoD2 to detect model deviation on binary parameters (with subject SWaT [8]).

Third, we assume that the compensated uncertainty terms can be described in normal distributions. Though normal distribution can be applied to most types of uncertainties reported by recent literature [19, 64], our approach can adapt to other types of uncertainty models by estimating model parameter value using other filters, such as particle filter.

Parameter as stochastic variable. To the best of our knowledge, all existing works treat the nominal model's parameter as a deterministic variable. Different from these pieces of works, MoD2 considers the concerned parameter as a stochastic variable, which is the key to the estimation in our approach. Besides the parameter's mean value derived through system identification, we additionally use the parameter's variance to make our estimation robust in handling the parameter's inherent fluctuation.

Leveraging the parameter's variance does not bring a significant burden to our approach. In fact, during system identification, such variance information is once collected but eventually overlooked. In system identification, the parameter's value is derived based on a collection of execution traces of the managed system. Since different traces could infer different candidate parameter values, linear regression is introduced to predict a parameter value that best fits all execution traces. Conventional system identification only reports this fitted value, and the candidate values are simply discarded. However, when the nominal model's parameter is treated as a stochastic variable, those candidates actually reflect the inherent inflection of the parameter. As such, in our MoD2, the previous fitted value is considered as the mean value of the parameter, and

configs	$\mathbf{env}_i$	$\mathbf{para}_i$	deviation
SWaT$^-$ & SWaT^+_D	input extracted from a real-world testbed [8]	original settings of the valves from existing literature [8, 17]	injecting manipulation disturbance to the system's valves according to [21]
	Example: friction factor value of the valves	Example: in-coming or out-going water rates of the valves	Example: $\mathbf{env}_d$ that changes the friction factor value from 0.0 to a random value in (0.0, 0.35)
SWaT^+_A	input extracted from a real-world testbed [8]	original settings from existing literature [8, 17]	injecting network attacks to the system's internal communications according to [17]
	Example: friction factor value of the valves	Example: in-coming or out-going water rates of the valves	Example: $\mathbf{para}_d$ that tampers with a valve's control signal from 0 (closed) to 1 (open)
RUBiS$^-$ & RUBiS$^+$	user request stream generated based on two real-world workloads [6, 7]	original settings in RUBiS from existing literature [54]	injecting attacks to the system's settings that mismatches the running environment according to [3]
	Example: an array of the time intervals of user requests	Example: working state of the servers	Example: $\mathbf{para}_d$ that changes servers' working state's value from "lowFidelity" to "highFidelity"
Encoder$^-$ & Encoder$^+$	video streams based on published dataset [23]	original settings in video encoder from existing literature [23]	introducing different types of video streams
	Example: road traffic video streams	Example: parameter values of the image compression algorithm	Example: $\mathbf{env}_d$ that replaces road traffic video streams with advertisement video streams

Table 1: Test configurations

the distances between the candidate values and the fitted value are quantized as the parameter's variance (i.e., Q).

5 EXPERIMENTS

We evaluate the effectiveness and usefulness of MoD2 and MoD2-based adaptation-supervision mechanism (and a Python implementation[1]) on three representative subject systems.

Our comparison baselines are SWDetecter [12] (the model deviation detection approach in the push-bottom method [23]) and LFM [17], which is one of the latest abnormal detection approaches for self-adaptive systems.

Particularly, we study the following two research questions:

RQ1 *Can MoD2 timely and effectively detect model deviation for self-adaptive systems?*

RQ2 *How useful is MoD2-based adaptation-supervision mechanism in terms of avoiding abnormal self-adaptation behavior?*

5.1 Experiment setup

5.1.1 Experimental subjects. We selected three prevalent subject systems from the community of self-adaptive systems. Each subject is provided with a control theory-derived controller and a mandatory controller, both of which are either given by the subject's developers or implemented following existing works.

- *SWaT*, a water treatment testbed we described in Section 2. The original control theory-derived controller is inherited from the programmable logic controllers in [17]. The mandatory controller (developed following [27, 58]) keeps the tank water level between upper and lower alarm levels locally.
- *RUBiS*, a web auction system (studied in [11, 44, 54]) that can adapt to workload/network changes by adjusting the number of servers to satisfy quality-of-service levels. The original control theory-derived controller (from the push-button method [23]) is tuned for high system utilization and low response time. The mandatory controller from [44] guarantees in-time response and maintains affordable system utilization.
- *Video encoder*, a video compression and streaming system (studied in [49–51]) that can adaptively change compression parameters to balance the throughout and video quality. The original control theory-derived controller (also from [23]) adjusts the compression parameters to achieve smooth and high-resolution video streams. The mandatory controller (also developed following [27, 58]) keeps a no-lagging video stream by decreasing the compression parameter values accordingly.

5.1.2 Test configurations. We conducted the experiments on predefined test configurations and compare MoD2 to SWDetecter and LFM over a series of *test configurations* (configurations for short). Given a subject system, each configuration is a simulated system run that consists of at most one model deviation. A configuration is *negative* if it contains no model deviation. Otherwise, a configuration that contains one model deviation is categorized as *positive*.

Since the subject's controllability features (i.e., the value of B) cannot be directly manipulated, we simulate model deviation resulted from discrete behavior by changing the subject's parameter values, and inaccurate environmental interaction by changing its environmental inputs.

Specifically, a configuration is denoted by a quintuple of (env_i, env_d, $para_i$, $para_d$, t), in which env_i and env_d denote the initial and deviated environmental input, $para_i$ and $para_d$ denote the initial and deviated model parameters, and t denotes the time point of model deviation. Initially, the subject and its running environment are reset with the initial parameters $para_i$ and env_i. At time point t, the model parameter is changed to $para_d$ and the environmental input is changed to env_d. For a negative configuration, $para_i$ =

[1]https://github.com/tongyanxiang/MoD2

	SWaT⁻ & SWaT⁺			RUBiS⁻ & RUBiS⁺			Encoder⁻ & Encoder⁺		
	MTD(s)	FN(%)	FP(%)	MTD(s)	FN(%)	FP(%)	MTD(s)	FN(%)	FP(%)
MoD2	40.11	0.0	0.0	0.00	0.0	0.3	0.00	2.0	3.0
SWDetector (θ=3σ)	274.92	94.0	93.3	365.00	4.0	10.0	0.00	40.0	46.5
SWDetector (θ=6σ)	OT	100.0	0.0	355.74	1.5	5.5	0.01	1.0	2.5

Table 2: Comparison of MoD2 and SWDetector

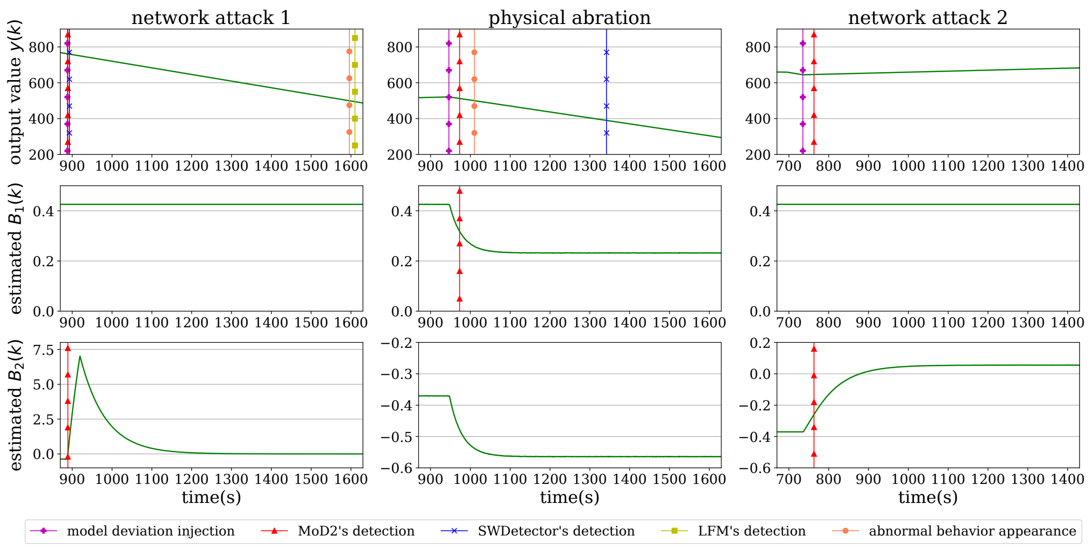

Figure 2: A case study on three positive configurations

$para_d$ and $env_i = env_d$ for the entire execution. For a positive configuration, $para_i/env_i$ is changed to a different $para_d/env_d$ at time t.

Our criteria for creating configurations followed existing works. The negative configurations for SWAT (SWaT$^-$) and RUBiS (RUBiS$^-$) cover most settings in existing works [8, 54], and the negative configurations for VideoEncoder (Encoder$^-$) are created based on [23] (we cannot cover it since the original datasets are not available). The positive configurations cover most reported model-deviated settings in SWAT (SWaT^+_D and SWaT^+_A) [17] and RUBiS (RUBiS$^+$) [3]. VideoEncorder's positive configurations (Encoder$^+$) were designed by our-own since no existing settings available. Each group includes 200 different configurations. Table 1 describes our configurations in detail.

5.1.3 Evaluation criteria. We measure the timeliness by the mean time delay (MTD, i.e., the average interval between the deviation point and the detection point of a positive configuration). Notice that we only consider the correctly-detected positive configurations in MTD.

We measure the accuracy by the false-negative rate (FN rate, i.e., the percentage of positive configurations that are falsely detected to be negative) and the false-positive rate (FP rate, i.e., the percentage of negative configurations that are falsely detected to be positive). Noticing that each positive configuration can be divided into a negative part (i.e., execution before the time point of model deviation) and positive part (i.e., execution after the time point of model deviation). FP also accounts for the positive configurations that are falsely detected in their negative parts.

We measure the usefulness by the positive configuration's abnormal rate (i.e., the ratio of a subject system's abnormal operation time to its suffered model deviation time). The abnormal operation time is measured as the time during which the subject system violates any control properties (e.g., overshoot of the water level in SWaT) or fails to fulfill its mandatory requirements (e.g., overflow or underflow in SWaT). The model deviation time is measured as the subject system's execution time after we inject model deviation. Notice that a positive configuration may never violate any control property, as we discussed in Section 2, model deviation is a necessary condition, not a sufficient condition, to the system's abnormal behavior.

5.1.4 Experiment procedure. We conducted all the experiments on an ECS server of Alibaba Cloud with 8 CPUs and 16GB of memory.

To answer **RQ1**, we compare three approaches' achieved MTD, FN rate, and FP rate on different configuration sets. We first compare the performance of MoD2 against SWDetector on SWaT$^-$, SWaT^+_D, RUBiS$^-$, RUBiS$^+$, Encoder$^-$, Encoder$^+$. We then compare the performance of MoD2, SWDetector, and LFM on SWaT^+_A, since LFM is proposed to address the challenges of attack-induced abnormal behavior only. The SWDetector is set with two different thresholds ($\theta = 3\sigma$ or $\theta = 6\sigma$), which are two of the suggested settings in [53].

We also study the impact of system identification, which could produce various identification values required by MoD2 (i.e., γ, W, V). We compare the performance of MoD2 with the identified values from different amounts of data (20–100%, step of 20%).

	SWaT$^-$ & SWaT$^+$				**RUBiS$^-$ & RUBiS$^+$**				**Encoder$^-$ & Encoder$^+$**			
	identification error	MTD(s)	FN(%)	FP(%)	identification error	MTD(s)	FN(%)	FP(%)	identification error	MTD(s)	FN(%)	FP(%)
20%	[-2.5%, 3.8%]	40.21	0.0	0.0	[-26.6%, 11.7%]	0.00	0.0	0.3	[1.2%, 6.4%]	0.00	2.0	3.0
40%	[-1.3%, 2.5%]	40.17	0.0	0.0	[-20.2%, 6.4%]	0.00	0.0	0.3	[-3.4%, 0.5%]	0.00	2.0	3.0
60%	[0.0%, 1.3%]	40.21	0.0	0.0	[-4.5%, 19.9%]	0.00	0.0	0.3	[0.2%, 6.1%]	0.00	2.0	3.0
80%	[0.0%, 1.3%]	40.21	0.0	0.0	[-1.3%, 7.4%]	0.00	0.0	0.3	[-0.2%, -0.2%]	0.00	2.0	3.0
100%	[0.0%, 0.0%]	40.11	0.0	0.0	[0.0%, 0.0%]	0.00	0.0	0.3	[0.0%, 0.0%]	0.00	2.0	3.0

Table 3: Comparison of MoD2's performance with different amounts of identification traces

	original	**MoD2-based**	**SWDetector-based (θ=3σ)**	**SWDetector-based (θ=6σ)**
SWaT	14.0% (162.45s)	0.0% (0.00s)	10.6% (114.35s)	7.1% (71.32s)
RUBiS	61.1% (1953.00s)	2.0% (60.00s)	11.1% (398.10s)	10.6% (386.40s)
video encoder	16.2% (2.92s)	0.4% (0.08s)	6.6% (1.19s)	0.1% (0.02s)

Table 4: Comparison of the abnormal rates w/wo adaptation-supervision mechanisms

To answer **RQ2**, we compare three subjects' abnormal rate on different configuration sets with and without MoD2-based adaptation-supervision. We also implemented and evaluated two SWDetector-based adaptation-supervision mechanisms to study the role of our proposed MoD2. We measure the abnormal rate on SWaT^+_A, RUBiS$^+$, and Encoder$^+$ respectively.

5.2 Experiment Results

RQ1 (effectiveness) Table 2 shows the performance of MoD2 and SWDetector for the three subjects. Considering detection timeliness, MoD2's detecting time for model deviation varies in different subjects. The reported MTD is 40.11 seconds for SWaT, 0.00 seconds for both RUBiS and video encoder. When comparing with SWDetectors, MoD2's achieved MTD is 185.76s smaller on average for the subjects. We notice that the two SWDetectors report similar detection delay for subject video encoder. This is because our injected model deviation would produce a severe change of the managed system's output, which favors the SWDetector's monitoring on the output values.

As for detection accuracy, MoD2 achieves good FN and FP rate in detecting model deviation. Generally, the average FN rate of MoD2 is 0.7% (0.0%–2.0%) and the average FP rate of MoD2 is 1.1% (0.0%–3.0%). We notice that for SWaT, the reported FN and FP rate are zero. For RUBiS, MoD2's reported FN rate is zero and FP rate is 0.3% which is caused by the active detector. MoD2 reports a 3.0% FN rate and a 2.0% FP rate for video encoder. This is caused by the difference between the selected initial estimate variance of model parameter value and its true value.

Comparing with SWDetectors of different settings, MoD2's FN rate is averagely 39.4% lower (−1.0%–100.0%), and its FP rate is averagely 25.2% lower (−0.5%–93.3%). When the SWDetector uses the best setting only (i.e., the window size ws is 28, and the threshold θ is 6σ), MoD2's FN rate is averagely 33.5% lower (−1.0%–100.0%), and its FP rate is averagely 1.6% lower (−0.5%–5.2%). We notice that MoD2's accuracy is slightly lower than SWDetectors on the subject of VedioEncoder. In this subject, the injected model deviation would immediately cause severe changes in the managed system's output, which favors the sliding window-based approach. MoD2 sacrifices its accuracy in detecting such noticeable model deviations slightly (1.0% lower FN rate and 0.5% lower FP rate) in exchange for its accuracy in detecting covert model deviations (50.8% higher FN rate and 2.7% higher FP rate).

	SWaT^+_A		
	MTD(s)	FN(%)	FP(%)
MoD2	11.15	0.0	0.0
LFM	886.45	1.0	0.5
SWDetector (θ=3σ)	84.60	85.0	84.0
SWDetector (θ=6σ)	1.06	37.5	0.0

Table 5: Comparison results of MoD2, SWDetector and LFM

Table 5 gives the experimental results on dataset SWaT^+_A (i.e., the positive configurations by injecting attacks to SWaT system). Basically, MoD2 outperforms the other three approaches (including the SWDetectors of different settings) in terms of detection timeliness and detection accuracy. MoD2 successfully detect all model deviations in a quite short time (averagely 11.15 seconds) with no false alarm. The learning-based LFM approach also achieves a good detection accuracy. However, it requires much longer time (886.45 seconds, 78.5 times longer than MoD2's) to work. The reason for LFM's large detection delay is that the accuracy of the trained classifier should be above a threshold of 85% to avoid false alarms. For SWDetector with a better setting (i.e., $\theta = 6\sigma$), although it reports the shortest detection time (1.06 seconds) and no false alarm, it fails to detect 37.5% positive configurations, which is the worst among the compared approaches.

To better demonstrate MoD2's effectiveness in detecting model deviation, we also perform a detailed study on the positive configurations that the baseline approaches fail to give a timely and accurate detection. We list three of those positive configurations in Figure 2. The first and the third configurations are gained by injecting network attacks to SWaT, and the second one is derived by injecting manipulation disturbance to SWaT's valves. For each of the configurations, we list the measured output value $y(k)$ (in the first line chart) and MoD2's estimated parameter values $B(k)$ (in the last two line charts, corresponding to the two arguments in $B(k)$). We use different marked lines to denote the time of model deviation injection, the time of MoD2's detection, the time of SWDetector's detection, the time of LFM's detection, and the time abnormal behavior appeared respectively. For the time of MoD2's detection, we also mark it on the corresponding $B_i(k)$'s line chart, indicating which argument exceeds its safe region.

For the first configuration, all of the three approaches successfully detect model deviation. However, LFM requires longer time since it has to gather multiple testing results to give an accurate detection. Such high detection delay makes LFM's detection unfeasible since SWaT's execution has already broken the control property (i.e., stability property) by the time it reports. For the second configuration, SWDetector's detection is too slow to be effective since the monitored output value shows no difference from the negative configuration in the early stages. For the third configuration, both SWDetector and LFM fail to detect model deviation. Their missing detections are reasonable since this model deviation has not caused severe consequences in SWaT's execution till the end of the execution trace. However, according to [17], this model deviation will eventually cause SWaT's abnormal behavior if the execution is prolonged from 30 minutes to 60 minutes.

Table 3 compares MoD2's performance with different amounts of traces for identification. We also listed the variation of the identified values (i.e., identification error) used by MoD2. Generally, changing the amount of identification traces has a limited impact on MoD2's effectiveness. The mean detection delay fluctuates in a small range (40.17s–40.21s for SWaT and 0.00s–0.00s for both RUBiS and video encoder) as the used amount of identification traces changes. MoD2 with different amounts of identification traces reports exactly the same FN and FP rate. Table 3 reveals MoD2's stable performance.

In summary, *MoD2 can detect model deviation with low detection delay (average* 13.37 *seconds) and good accuracy (*0.8% *FN rate and* 1.0% *FP rate). Comparing with the baseline approaches, MoD2 achieves a smaller detection delay (average* 185.76 *seconds or* 93.3% *smaller), as well as a better accuracy (*39.4% *lower FN rate and* 25.2% *lower FP rate).*

RQ2 (usefulness) Table 4 compares the three subjects' abnormal rates on different positive configuration sets with different adaptation-supervision mechanisms. For the subjects with our MoD2-based mechanism, the average abnormal rate is 1.2% (0.0%–2.0%). Comparing with the subjects without our mechanisms (denoted as *original*), the abnormal rates drop 29.2% (14.0%–59.1%) on average. This result validates the usefulness of our approach in alleviating the impact of model deviation. Particularly, our mechanism achieves a zero abnormal rate for SWaT system (i.e., prevent all model-deviation-caused severe consequences), while the abnormal rate without our mechanism is 14.0%. So even if SWaT's optimal controllers are carefully designed and implemented by the field experts, one cannot assume that the controllers' robustness could handle all cases of model deviation [17].

We also list each subjects' abnormal operation time along with their suffered abnormal rates. Each subject averagely suffers 706.12 seconds abnormal operation time in each configuration without MoD2. In other words, the corresponding control-SASs fail to provide any guarantees on the subject systems' behavior in nearly 11.77 minutes. Notice that the aforementioned abnormal operation time could be longer if model deviation has not been appropriately addressed, since the execution we collected only lasts for 30 minutes.

Comparing with the two SWDetector-based mechanisms, the results reflect the importance of our MoD2 in guarding the subjects' adaptation. Specifically, MoD2-based mechanism achieves 6.5% lower abnormal rate (3.0%–8.9%), as well as 141.87 seconds shorter abnormal operation time. We believe that MoD2 is more effective in protecting control-SASs.

The result displayed in Table 4 can also partly reflect the correctness of detecting model deviation based on estimated parameter values. The abnormal rate with supervision indicates the output abnormalities prevented by our detection (1.2% on average), which echoes the low FN rate of MoD2. The abnormal rate without supervision would imply the low FP rate of MoD2, and that average 30.4% of the experimental subjects' execution time suffers abnormalities.

In summary, *Our MoD2-based adaptation-supervision mechanism can alleviate the impact of model deviation on the managed system. With the support of our mechanism, the abnormal rate averagely drops by* 29.2% *comparing the original control-SASs and averagely drops by* 6.5% *comparing the SWDetector-based mechanism.*

5.3 Threats to Validity

One major concern on the validity of our empirical conclusions is the selection of evaluation subjects. We only use three subjects as the managed systems. This might harness the generalization of our conclusions. A comprehensive evaluation requires a full understanding of the managed systems, as well as their suitable control theory-derived controllers. This requirement restricts our choice of possible experimental subjects. Nevertheless, we believe that our selected subjects are representative of their different platforms (including network systems and cyber-physical systems) and architectures (including single controller and multiple controllers). Moreover, all of the selected subjects are widely used by other self-adaptation researchers as their experimental subjects or motivating systems.

Another concern is about injecting model deviations in the positive configurations. Since we cannot directly manifest the subject system's controllability features, we can only modify its parameters or inputs to simulate model deviation. This might make our experimental settings less realistic. To address this problem, we carefully design the injected model deviations. For SWaT, the model deviation is based on the reported physical abrasion [8] and network attacks [17]. For RUBiS, the model deviation is designed according to reported failures in web service systems [3]. And for video encoder, we use real-world video streams to simulate the model deviation.

6 RELATED WORK

Control theory has been widely exploited to implement self-adaptive systems for their theoretical guarantees. Some pieces of related work use control-theoretical techniques to refine their architecture-based system model for the managed system. For example, Checiu et al. use a server request model to describe the behavior of an adaptive web service system [15]. Filieri et al. use discrete-time Markov chain model to depict service-oriented applications [22]. Some others use control-theoretical techniques to guide the design of the managing system. Konstantinos et al. [4] propose the CobRA framework for designing self-adaptive web-services, which uses model predictive control to guide the adaptation strategy of the managing system. Gabriel et al. [54] combine control-SASs and traditional architecture-based SASs by introducing discrete-time Markov chain in their PLA adaptation framework. Different from these pieces of work, the focus of MoD2 is not designing the managing system but providing self-adaptation assurances in the presence of model deviation. In other words, our work can be regarded as an complementary for them by alarming their managing system the occurrence of model deviation.

Model deviation has received the attention of self-adaptation researchers since control-SASs emerged. According to control theory, the precision of the nominal model in control-SASs directly determines the effectiveness of the derived managing systems. Baresi et al. propose using a grey-box discrete-time feedback controller to support robust self-adaptation that can overcome slight model deviation [11]. Filieri et al. use continuous learning mechanisms to keep the nominal model updating at runtime [24]. Maggio et al. use Kalman filter to revise the identified nominal model by updating its state values [49]. Comparing with these pieces of works, our MoD2-based mechanism concentrates on the model deviation that would cause the managing system to violate control properties, as well as the managed system behaving abnormally. Together with these works that address slight deviation of the model parameters, we can achieve model-deviation-free self-adaptation for control-SASs.

The key component of our proposed mechanism is a timely and accurate detector for model deviation, which is similar to the works on abnormal detection. Many research efforts have been contributed to both control theory community and self-adaptation community. Window-based approach is the most-widely used abnormal detection approach in control theory area [32], which is similar to the compared SWDetector approach in Section 5. Ozay et al. propose a set-membership approach to detect property violations occurring in the control system [33, 34]. For self-adaptation researchers, Jiang et al. derive invariants by observing messages exchanged between system components for robotic systems [39]. Chen et al. propose an SVM-based method to detect network attacks to SWaT system [17]. Qin et al. use context information to refine the derived invariants and combine multiple testing results to improve the accuracy of abnormal detection [57].

As we discussed in Section 2, the major difference between our MoD2 and these approaches is that we directly estimate the values of nominal model's parameters instead of monitoring the system's output values. By doing so, we reduce the delay time for model deviation detection.

Environmental uncertainty has always been a challenge for self-adaptation researchers. Most of these works focus on uncertainty's impact on the managed system's state. Esfahani et al. identify internal uncertainty and external uncertainty and propose a probability-based approach to assess both the positive and negative consequences of uncertainty [20]. Ghezzi et al. propose an adaptation framework to manifest non-functional uncertainty via model-based development [28]. Angelopoulos et al. also use Kalman filter to alleviate uncertainty's impact on the estimated states of the managed system's nominal model [4]. MoD2's handling of uncertainty follows these works. However, the focus of our MoD2 is on the uncertainty's impact on the nominal model's parameters, which is addressed by our parameter deviation estimation technique.

7 CONCLUSION

In this paper, we present an adaptation-supervision mechanism with the addition of *supervision loop* for alleviating the impact of model deviation in control-SASs. The key to our mechanism is a novel detector, MoD2, which combines different techniques, including parameter deviation estimation, uncertainty compensation, and safe region quantification, to balance the detector's timeliness and accuracy. We conduct experiments to show the effectiveness of MoD2 and the usefulness of MoD2-based adaptation-supervision mechanism.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their comments. This work is supported by Key-Area Research & Development Program of Guangdong Province (Grant #2020B010164003), Natural Science Foundation of China (Grants #62025202, #61932021, #61902173), and Natural Science Foundation of Jiangsu Province (Grants #BK20190299).

REFERENCES

[1] Sridhar Adepu and Aditya Mathur. 2016. An investigation into the response of a water treatment system to cyber attacks. In *Proceedings of 17th IEEE International Symposium on High Assurance Systems Engineering*. IEEE, 141–148.

[2] Cristina Alcaraz and Stephen Wolthusen. 2014. Recovery of structural controllability for control systems. In *International Conference on Critical Infrastructure Protection*. Springer, 47–63.

[3] 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.

[4] 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.

[5] 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.

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

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

[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] 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.

[11] 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.

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

[13] Mogens Blanke, Michel Kinnaert, Jan Lunze, Marcel Staroswiecki, and Jochen Schröder. 2006. *Diagnosis and fault-tolerant control*. Vol. 2. Springer.

[14] Yuriy Brun, Giovanna Di Marzo Serugendo, Cristina Gacek, Holger Giese, Holger Kienle, Marin Litoiu, and et al. 2009. Engineering self-adaptive systems through feedback loops. In *Software Engineering for Self-Adaptive Systems*. Springer, 48–70.

[15] Laurentiu Checiu, Bogdan Solomon, Dan Ionescu, Marin Litoiu, and Gabriel Iszlai. 2011. Observability and controllability of autonomic computing systems for composed web services. In *Proceedings of the 6th IEEE International Symposium on Applied Computational Intelligence and Informatics*. IEEE, 269–274.

[16] Xing Chen, Shihong Chen, Yun Ma, Bichun Liu, Ying Zhang, and Gang Huang. 2019. An adaptive offloading framework for Android applications in mobile edge computing. *Science China Information Sciences* 62, 8 (2019), 1–17.

[17] 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.

[18] OW2 Consortium et al. 2008. Rubis: Rice university bidding system. *URL http://rubis. ow2. org* (2008).

[19] 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.

[20] 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.

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

[22] Antonio Filieri, Carlo Ghezzi, Alberto Leva, and Martina Maggio. 2011. Self-adaptive software meets control theory: A preliminary approach supporting reliability requirements. In *Proceedings of 26th IEEE/ACM International Conference on Automated Software Engineering*. IEEE, 283–292.

[23] 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.

[24] 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.

[25] Antonio Filieri, Martina Maggio, Konstantinos Angelopoulos, Nicolás D'Ippolito, Ilias Gerostathopoulos, Andreas Berndt Hempel, and et al. 2015. Software engineering meets control theory. In *Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE Press, 71–82.

[26] Antonio Filieri, Martina Maggio, Konstantinos Angelopoulos, Nicolás D'ippolito, Ilias Gerostathopoulos, Andreas Hempel, and et al. 2017. Control strategies for self-adaptive software systems. *ACM Transactions on Autonomous and Adaptive Systems* 11, 4 (2017), 1–31.

[27] 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.

[28] Carlo Ghezzi, Leandro Sales Pinto, Paola Spoletini, and Giordano Tamburrelli. 2013. Managing non-functional uncertainty via model-driven adaptivity. In *Proceedings of the 35th International Conference on Software Engineering*. IEEE, 33–42.

[29] Jayanta K Ghosh, Mohan Delampady, and Tapas Samanta. 2007. *An introduction to Bayesian analysis: theory and methods*. Springer Science & Business Media.

[30] Graham Goodwin, Stefan Graebe, and Mario Salgado. 2001. *Control system design*.

[31] 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.

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

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

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

[35] 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.

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

[37] Emilio Incerto, Mirco Tribastone, and Catia Trubiani. 2017. Software performance self-adaptation through efficient model predictive control. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*. IEEE, 485–496.

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

[39] Hengle Jiang, Sebastian Elbaum, and Carrick Detweiler. 2013. Reducing failure rates of robotic systems though inferred invariants monitoring. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 1899–1906.

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

[41] Raphael Jungers, Atreyee Kundu, and Maurice Heemels. 2017. Observability and controllability analysis of linear systems subject to data losses. *IEEE Trans. Automat. Control* 63, 10 (2017), 3361–3376.

[42] Eunsuk Kang, Sridhar Adepu, Daniel Jackson, and Aditya Mathur. 2016. Model-based security analysis of a water treatment system. In *Proceedings of the 2nd International Workshop on Software Engineering for Smart Cyber-Physical Systems*. IEEE, 22–28.

[43] Youngjoo Kim and Hyochoong Bang. 2018. Introduction to Kalman filter and its applications. *Introduction and Implementations of the Kalman Filter* 1 (2018), 1–16.

[44] 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.

[45] Qiang Li, Ranyang Li, Kaifan Ji, and Wei Dai. 2015. Kalman filter and its application. In *Proceedings of the 8th International Conference on Intelligent Networks and Intelligent Systems*. IEEE, 74–77.

[46] 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.

[47] Marin Litoiu, Mary Shaw, Gabriel Tamura, Norha Villegas, Hausi Müller, Holger Giese, and et al. 2017. What can control theory teach us about assurances in self-adaptive software systems? In *Software Engineering for Self-Adaptive Systems III. Assurances*. Springer, 90–134.

[48] Lennart Ljung. 1999. System identification. *Wiley encyclopedia of electrical and electronics engineering* (1999), 1–19.

[49] 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.

[50] Martina Maggio, Alessandro Vittorio Papadopoulos, Antonio Filieri, and Henry Hoffmann. 2017. Self-adaptive video encoder: Comparison of multiple adaptation strategies made simple. In *Proceedings of 12th IEEE/ACM International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, 123–128.

[51] 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.

[52] Claudio Menghi, Shiva Nejati, Lionel Briand, and Yago Isasi Parache. 2020. Approximation-refinement testing of compute-intensive cyber-physical models: An approach based on system identification. In *Proceedings of the 42nd International Conference on Software Engineering*. IEEE, 372–384.

[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] Alexander Palm, Andreas Metzger, and Klaus Pohl. 2020. Online reinforcement learning for self-adaptive information systems. In *Proceedings of the 32nd International Conference on Advanced Information Systems Engineering*. Springer, 169–184.

[56] Tharindu Patikirikorala, Alan Colman, Jun Han, and Liuping Wang. 2012. A systematic survey on the design of self-adaptive software systems using control engineering approaches. In *Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, 33–42.

[57] 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.

[58] 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.

[59] 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.

[60] 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.

[61] 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.

[62] Silvio Simani, Cesare Fantuzzi, and Ronald Jon Patton. 2003. Model-based fault diagnosis techniques. In *Model-based Fault Diagnosis in Dynamic Systems Using Identification Techniques*. Springer, 19–60.

[63] 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.

[64] 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.

[65] 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.