Model-based Testing under Parametric Variability of Uncertain Beliefs

Matteo Camilli[0000−0003−2491−5267] and Barbara Russo[0000−0003−3737−9264]

Faculty of Computer Science,
Free University of Bozen-Bolzano, Bolzano Italy
{mcamilli,brusso}@unibz.it

Abstract. Modern software systems operate in complex and changing environments and are exposed to multiple sources of uncertainty. Considering uncertainty as a first-class concern in software testing is currently on an uptrend. This paper introduces a novel methodology to deal with testing under uncertainty. Our proposal combines the usage of parametric model checking at design-time and online model-based testing algorithms to gather runtime evidence and detect requirements violations. As modeling formalism, we adopt parametric Markov Decision Processes where transition probabilities are not fixed, but are possibly given as a set of uncertain parameters. The design-time phase aims at analyzing the parameter space to identify the constraints for requirements satisfaction. Then, the testing activity applies a Bayesian inference process to identify violations of pre-computed constraints. An extensive empirical evaluation shows that the proposed technique is effective in discovering violations and is cheaper than existing testing under uncertainty methods.

Keywords: Model-based Testing · Parametric Markov Decision Processes · Uncertainty analysis · Bayesian inference

1 Introduction

Modern software-intensive systems are often situated in complex ecosystems that can be hard or even impossible to fully understand and precisely describe at design-time. Nevertheless, unreliable or unpredictable software behavior cannot be tolerated as society increasingly depends on it. For this reason, there exists the increasing need for systematic approaches to deal with incomplete knowledge and sources of uncertainty while engineering complex systems. Endowing conventional software engineering methodologies with techniques and practices able to model, quantify, and mitigate uncertainty is becoming increasingly crucial [1]. In particular, research effort in techniques that explicitly consider uncertain expected behavior in software testing is currently on an evident uptrend (e.g., see [2,3,4,5]).

This paper introduces a novel methodology to deal with uncertainty quantification by combining parametric model checking [6] at design-time and online (or on-the-fly) Model-based Testing (MBT) algorithms [7,8]. MBT is a software testing technique where run-time behavior of a software System Under Test (SUT) is checked against a formal model describing the system's behavior [7]. Our MBT approach allows the design-time probabilistic model to be underspecified. Namely, the modeler can explicitly represent partial knowledge on the SUT by means of uncertain model parameters in a Markov Decision Process (MDP). Parametric model checking is used to verify design-time requirements under variability of uncertain model parameters. The outcome of this stage is a mapping from regions of these parameters to truth values encoding the verification conditions to be evaluated at runtime. Thus, our MBT approach leverages these conditions to drive the testing activity by maximizing the probability to hit the uncertain components of the SUT multiple times. The objective of the MBT phase is to gather runtime evidence over uncertain model parameters using a Bayesian inference approach [9]. Thus, the MBT spots requirements violations by comparing the incremental posterior knowledge and the pre-computed verification conditions on uncertain parameters. The whole methodology is supported by a software toolchain whose core component is a MBT module which integrates test case generation, execution and evaluation. The MBT module makes use of fine grained characteristics of the uncertain model parameters to reduce the effort required by testing. The design-time analysis of the region space of uncertain parameters (i.e., the outcome of the parametric model checker) is leveraged to detect requirements violations and decide over termination.

To illustrate our approach we make use of an existing open-source case study called SafeHome. It represents an exemplar of Cyber Physical Systems (CPSs) borrowed from [5]. We conducted an empirical evaluation to study the cost-effectiveness of our testing method by varying the number of uncertain parameters and the distance between actual values and verification conditions. We also compared our approach with selected existing MBT methods, pointing out advantages and threats to validity.

The major contribution of this paper can be summarized as follows:

i description of our methodology to MBT under parametric variability of uncertain beliefs;
ii extensive evaluation to assess the cost-effectiveness of our approach and comparison with existing testing methods.

Our empirical evaluation shows that the whole methodology is effective to spot requirements violations with bounded effort. Furthermore, the developed MBT method outperforms existing MBT strategies.

The remainder of this article is structured as follows. Section 2 introduces a preview of our methodology. Section 3 recalls the necessary background concepts. Section 4 presents a running example (i.e., the SafeHome system), used throughout the article to illustrate the main phases of the methodology. Section 5 introduces a formal treatment of our approach. Section 6 reports our evaluation and discusses threats to validity. Section 7 describes related work. Section 8 concludes the paper.

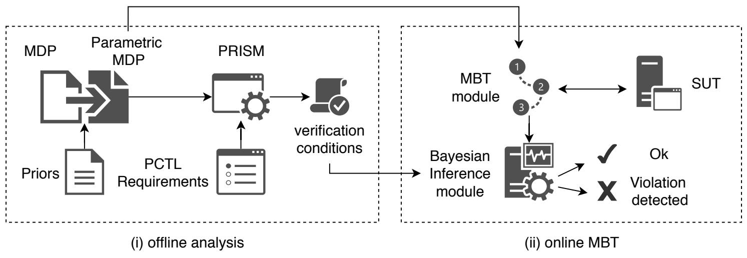


Fig. 1: High-level schema of our approach.

2 Preview of the Approach

In this work we focus on systems modeled as MDPs and quantitative requirements expressed using Probabilistic Computation Tree Logic (PCTL) [10]. MDPs represent a widely accepted formalism to model and verify software system dependability (e.g., reliability, availability, safety) [11]. Recent research activities show also the effective usage of MDPs in testing probabilistic systems [2]. As described in [12], models developed at design-time are often subject to sources of uncertainty. Namely, certain behaviors of the system itself and the surrounding environment are hard to predict. For instance, the success probability of a software task, or the failure rate of a hardware device (represented by transitions in the MDP) may be hard to specify in a complete and accurate way. So, to deal with uncertainty, our approach gives the modeler the ability of representing partial knowledge (i.e., beliefs) on transition probabilities by means of Prior probability density functions [9], or simply Priors. On top of these assumptions, we informally introduce here the two main phases of the approach (see Figure 1): (*i*) offline analysis; and (*ii*) online MBT.

The offline analysis (or pre-computation) aims at studying the parameter space. We leverage the parametric model checking functionality of PRISM [6] to analyze how parameters affect the satisfaction of PCTL requirements. The Prior density functions are used to mechanically build a parametric MDP model and the search space of each individual uncertain parameter. The result of the model checker is a mapping from regions of these parameters to truth values (i.e., either true or false with respect to requirements satisfaction). It is worth noting that model checking is computationally expensive and requires exhaustive exploration of the model's state space to analyze arbitrarily complex properties [13]. Since the computational cost of model-checking may be prohibitive for online usage, we keep this pre-computation separated as an offline phase, where we can execute demanding activities without interfering with the system operation. The outcome of the pre-computation encodes verification conditions to be satisfied to meet the requirements. Thus, the online MBT phase performs a controlled exploration of the SUT by using an uncertainty-aware test case generation strategy. Such a strategy leverages the structural characteristics of the model to direct the effort towards transitions associated with uncertain parameters. In other words, we aim at concentrating on those components of the SUT whose behavior is subject to sources of uncertainty. The MBT feeds a Bayesian inference process that computes the actual value of uncertain model parameters based on the evidence gathered during testing. The actual values are then checked against the (pre-computed) verification conditions to detect violations of design-time requirements.

3 Background

This section recalls required background notions to understand the formal aspects of the approach we developed. In the following we briefly revisit parametric MDPs, the quantitative temporal logic PCTL, and Bayesian Inference. For a complete treatment we let the reader refer to [11,14,9].

3.1 Parametric Markov Decision Processes

Let θ be a finite set of variables. Let $\mathbb{Q}[\theta]$ denote the set of all rational-coefficient polynomial functions (i.e., a sum of terms, where each term is given by a coefficient and a monomial). A parametric Markov Decision Process (pMDP) $\mathcal{M}$ is a tuple $(S, \theta, A, s_0, \delta, AP, L)$ where S is a (finite) set of states, θ is a finite set of parameters, A is an alphabet of actions, $s_0 \in S$ is the initial state, and $\delta : S \times A \times S \to \mathbb{Q}[\theta] \cup [0, 1]$ is the partial probabilistic transition function, AP is a set of atomic propositions, $L : S \to 2^{AP}$ is a labeling function that associates to each state the set of atomic propositions that are true in that state. State transitions occur in two steps: a nondeterministic choice among available actions; and a stochastic choice of the successor state according to δ. In the rest of the paper, the notation $p^a_{i,j}$ will be used as short form for $\delta(s_i, a, s_j)$. The function $A(s_i)$ is used to denote the actions in A available from the state s_i.

Note that a parameter-free pMDP coincides with standard MDP, as defined in [11]. A MDP can be obtained from a pMDP by simply assigning values to parameters. Formally, we need to create an *instantiation* $val : \theta \to \mathbb{R}$ s.t. the δ function is well-defined, i.e., $\sum_{s_j \in S} p^a_{i,j} = 1$ for all $s_i \in S$ and $a \in A(s_i)$. In the following we use $\mathcal{M}[val]$ to denote the MDP obtained from the pMDP $\mathcal{M}$ with instantiation val.

Both MDP and pMDP models can be augmented with *reward*s to quantify a benefit (or loss) due to the occurrence of a certain transitions. A reward usually represents non-functional aspects such as average execution time, power consumption or usability. Rewards are formally specified by using the notion of *reward structure*, i.e., a function $r : S \times A \times S \to \mathbb{R}$. Given a standard MDP and a reward structure r, a deterministic policy π specifies for each state s_i the action $\pi(s_i) \in A(s_i)$ chosen by a decision maker to solve nondeterminism. The notion of *best policy* π^* refers to the policy able to maximize the expected cumulated reward over a potentially infinite horizon. The best policy can be computed solving the following Bellman's equation 1 using dynamic programming approaches as reported in [11].

$$\pi^*(s_i) = \underset{a \in A(s_i)}{\arg\max} \sum_j p^a_{i,j} \cdot (r^a_{i,j} + \gamma V^*(s_j)) \tag{1}$$

where $V^*(s_j)$ represents the expected cumulated reward when starting from s_j and acting optimally along a infinite horizon; $\gamma \in [0, 1]$ is a discount factor that alleviates the contribution of future rewards in favor of present rewards.

3.2 Probabilistic Computation Tree Logic

To specify requirements of interest we consider here the logic PCTL. The syntax supports the definition of state formulas ϕ and path formulas ψ, which are evaluated over states and paths, respectively. Formally, a formula is defined as follows:

$$\phi ::= \mathit{true} \mid a \mid \phi \wedge \phi \mid \neg\phi \mid \mathcal{P}{\bowtie}p[\psi], \quad \psi ::= \mathtt{X}\phi \mid \phi \;\mathtt{U}\; \phi, \tag{2}$$

where $a \in AP$ and a path formula ψ is used as the parameter of the *probabilistic path operator* $\mathcal{P}{\bowtie}p[\psi]$, such that $\bowtie \in \{\leq, <, \geq, >\}$ and $p \in [0, 1]$ is a probability bound. The symbol X represents the *next* operator, U is the *until* operator. The operators G (i.e., *globally*) and F (i.e., *eventually*) can be derived from the previous ones as for CTL. A state $s \in S$ satisfies $\mathcal{P}{\bowtie}p[\psi]$ if, under any nondeterministic choice, the probability of taking a path from s satisfying ψ is in the interval specified by ${\bowtie}p$.

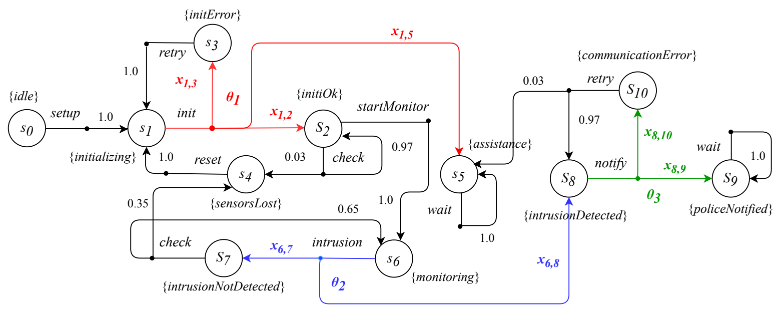
Parametric model checking [14] is a verification technique able to analyze the parametric variability of a pMDP model $\mathcal{M}$ and determine how such a variability affects the satisfaction of a set of target PCTL properties. Formally, the outcome of the model checker is a mapping between *hyper-rectangle*s and truth values, where an hyper-rectangle is a multidimensional rectangle $h = \bigtimes_{x \in \theta} [l_x, u_x]$ with $l_x, u_x \in \mathbb{R}$ lower- and upper-bound for parameter x, respectively. Intuitively, for each *true* hyper-rectangle h, the model $\mathcal{M}[val]$ satisfies the properties iff $val(x) \in [l_x, u_x]$ for all $x \in \theta$.

3.3 Bayesian Inference

Bayesian inference [15] really comes into its own in domains where uncertainty must be taken into account. The main goal is to learn about one or more uncertain/unknown parameters θ affecting the behavior of a stochastic phenomenon of interest. The Prior knowledge (i.e., initial hypothesis or belief) of θ is incrementally updated based on a collected data sample $y = (y_1, y_2, \ldots, y_n)$ describing the actual behavior of the target phenomenon. By using Bayes' theorem we obtain the Posterior distribution $f(\theta|y)$, describing the best knowledge of θ, given the evidence y.

$$f(\theta|y) \propto f(\theta) \cdot f(y|\theta) \tag{3}$$

The density $f(y|\theta)$ is usually referred to as the likelihood function and represents the compatibility of the data with the hypothesis. The hypothesis is often



region	state-action	affected level	target states	beliefs (i.e., uncertain probability values)
θ_1	s_1-*init*	application	s_2, s_3, s_5	$x_{1,2} = 0.95, x_{1,3} = 0.03, x_{1,5} = 1-(x_{1,2}+x_{1,3}) = 0.02$
θ_2	s_6-*intrusion*	integration	s_7, s_8	$x_{6,7} = 0.05, x_{6,8} = 1 - x_{6,7} = 0.95$
θ_3	s_8-*notify*	infrastructure	s_{10}, s_{11}	$x_{8,9} = 0.98, x_{8,10} = 1-x_{8,9} = 0.02$

Fig. 2: MDP model of the SafeHome system.

available from external sources such as expert information based on past experience or previous studies. This information is encoded by the Prior distribution $f(\theta)$. The posterior distribution can be used in turn to perform point and interval estimation. Point estimation is typically addressed in the multivariate case, by summarizing the distribution through the Posterior mean $E[f(\theta|y)]$ and the (95%) Highest Density Region HDR$[f(\theta|y)]$, defined as follows.

$$E[f(\theta|y)] = \int \theta \cdot f(\theta|y) d\theta, \quad \text{HDR}[f(\theta|y)] = \{\theta : f(\theta|y) \geq 0.95\} \tag{4}$$

The magnitude of the HDR region yields the highest possible accuracy in the estimation of the true value of θ and is usually adopted as a measure of the confidence gained after the inference process. In Bayesian statistics, it represents the credible region within which parameter values fall with probability 0.95.

4 A Running Example: the SafeHome System

The SafeHome security system represents an open-source benchmarking example in charge of controlling and configuring alarms and related sensors that implement a number of security and safety features such as intrusion detection. For the sake of readability, here we use an extract of the whole SafeHome by emphasizing the relevant characteristics for our problem domain. We let the reader refer to [5] for a comprehensive description.

Figure 2 shows the high-level behavior of the system through a pMDP model. The system behavior exhibits three main phases: *initialization*, *monitoring* and *alarm*, in charge of sensor initialization, detection, and alarm handling, respectively. From state s_2 the SafeHome system tries to initialize all the available sensors by executing the action *init*. If the task succeeds, the sensors are correctly registered and the action *startMonitoring* can be executed to proceed towards

Table 1: PCTL requirement examples for the SafeHome system.

id	type	description	PCTL definition
R_1	global reliability	The probability of reaching a state where assistance is required is less than 0.05	$P_{<0.05}$ [F *assistance*]
R_2	sensors availability	The probability of observing operable sensors without failures is greater than 0.9	$P_{>0.9}$ [!*sensorsLost* U *sensorsOk*]
R_3	network reliability	If sensors are operable, the probability of eventually notifying the emergency unit is greater than 0.98	*sensorsOk* $\rightarrow P_{>0.98}$ [F *policeNotified*]

the *monitoring* and then *alarm* phase. According to [4], sources of uncertainty in CPSs affect the behavior of the SUT at different levels: *application* level, due to events/data originating from software components running upon physical units of the CPS; *infrastructure* level, due to data transmission through networking and/or cloud infrastructure; *integration* level, due to interactions among physical units at either application level or infrastructure levels. For instance, consider the following common scenario in our target system. When the system is in state s_6 (i.e., *monitoring* holds), sensors can send the *intrusion* trigger to the security system that eventually causes a notification to be sent to an external emergency service (i.e., *policeNotified* state). However, the intrusion detection is affected by uncertainty at integration level. In fact, this capability is conditioned by the way sensors interact and their individual ability of correctly sensing the physical environment. Thus, the action *intrusion* leads to either state s_8 (i.e., the intrusion has been sensed) or state s_7 (i.e., the intrusion has not been sensed) with a substantial degree of uncertainty. This uncertain outcome is explicitly represented by uncertain parameters (i.e., $x_{6,8}$ and $x_{6,7}$, respectively). The uncertain parameters associated with a state-action pair is called *uncertain region* and we denote it as θ_i. The disjoint union of all θ_i is θ (i.e., the set of uncertain parameters). Figure 2 lists all the uncertain regions in SafeHome and affected levels.

Table 1 lists some requirements for our example, formally specified using PCTL. It is worth noting that the ability of satisfying these requirements depends on the actual value of model parameters. Figure 2 contains initial (uncertain) beliefs on these parameters.

5 The Testing Framework

This section illustrates the whole testing framework. The presentation is partitioned into two main fragments, reflecting the two main phases of our proposal.

5.1 Offline Analysis

The tester specifies the SUT behavior through a pMDP model (e.g., the SafeHome model in Figure 2). The uncertain values of each region θ_i are formally defined by a k-dimensional *categorical* distribution [9], with k the number of target states from the state-action (s, a) identified by θ_i. For instance, the region θ_1 is defined by the density function $Cat(x_{1,2}, x_{1,3}, x_{1,5})$, describing the distribution of transition probabilities from s_1 to s_2, s_3, and s_5, respectively, when the action *init* is chosen. As described in [9], the natural conjugate Prior of a categorical distribution is the *Dirichlet* distribution (i.e., a multivariate generalization of the *Beta*). For instance, the mdoeler specifies the Prior knowledge of θ_1 by using either a non-informative Prior $Dir(1, 1, 1)$, or a informative one, such as:

$$f(\theta_1) = Dir(47, 2, 1) \tag{5}$$

when past experience is available. In this latter case, the Prior has been built based on 50 past observations as follows: $47 \cdot s_2$, $2 \cdot s_3$, and $1 \cdot s_5$.

The Prior knowledge specification provides the baseline for further offline and online analysis. The initial guess for the uncertain parameters (e.g., values assigned to parameters in Figure 2) is automatically extracted by summarizing the Priors through the mean values. The 95% HDR is used instead of computing the range of possible values for each parameter in θ. Thus, we leverage this information to limit the search space only to those values that are credible with respect to the given beliefs. For instance, given the informative Prior defined in Equation 5, the HDR sets the following bounds.

$$HDR[f(\theta_1)] = x_{1,2} \in [0.87, 0.99], x_{1,3} \in [0.00, 0.09], x_{1,5} \in [0.00, 0.05] \tag{6}$$

Since the Dirichlet is multivariate, the HDR is composed of a number of intervals, one for each parameter.

It is worth noting that the HDR of each marginal distribution of a Dirichlet (i.e., univariate Beta distribution) is instead a single interval defining the bounds of each individual parameter. In the rest of the paper we will use HDR$[f_x(\cdot)]$ to denote the HDR of the marginalized $f(\cdot)$ by retaining the variable x. For instance, considering the Prior of θ_1 introduced in Equation 5, the following holds.

$$\text{HDR}[Dir_{x_{1,2}}(\cdot)] = [0.87, 0.99] \tag{7}$$

After computing the HDR of each Prior, we execute the parametric model checking functionality of PRISM to obtain the hyper-rectangles that meet the desired PCTL requirements (e.g., SafeHome properties in Table 1). In our running example, the outcome of this activity is a set of *true* hyper-rectangles $\{h_1, \ldots, h_n\}$ encoding verification condition for the SafeHome. Each element h_i is composed of a number of closed intervals, one for each uncertain parameter.

5.2 Online Model-based Testing

As anticipated in Section 5, the online phase takes as input the model and the hyper-rectangles to carry out the testing activity. The online MBT aims at exploring the SUT in a controlled way by directing the effort towards the uncertain model regions.

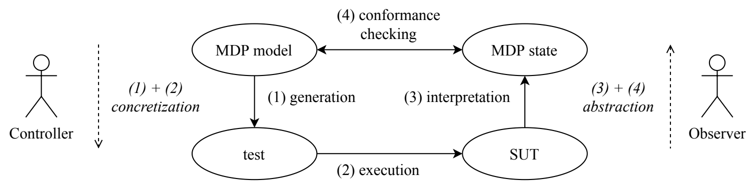


Fig. 3: Conformance game iteration.

Conformance Game. Figure 3 shows the main steps of the whole iterative approach. The idea originally introduced in [7] is to view the SUT as a black box and distinguish between *controllable* behavior from the tester (i.e., inputs or more in general external stimuli from the environment) and *observable* behavior from the running software system (i.e., outputs or more in general an observable stimulus–response). The Controller and the Observer components execute a *conformance game* [7,2] until the *termination condition* is met. The game starts from the initial state of the MDP model and, for each step, the controller chooses an available action in $A(s)$ from the current state s, depending on the adopted test case generation strategy. The generation step (1) translates the action to a controllable behavior to reach the same level of abstraction of the SUT. Intuitively, the chosen action maps to a valid input provided by using a service exposed by the SUT APIs. At this point, the external stimulus is provided to the SUT (2) that reacts in turn by exposing an observable outcome. Thus, the observer evaluates (3) the outcome and interprets it to determine the target model state s' in order to reach again the level of abstraction of the MDP. Here the evaluation is conducted by means of a post-condition function mapping to model states. This way, we can determine the target state s' s.t. the post-condition, evaluated on the observed outcome, holds. The last step is the conformance checking (4) that verifies whether the obtained outcome is feasible in the sense of the formal specification. Formally, the above steps are used to verify the existence of a *conformance relation* between the model and the SUT. The conformance relation is formalized in turn by leveraging the notions of *probabilistic alternating simulation* and *refinement*, as described in [16]. A comprehensive theoretical discussion of these aspects is not part of the contribution of this paper, so we let the reader refer to [16] for further details.

Bayesian inference module. Besides the conformance game, the Observer component feeds a Bayesian inference process (see Section 3) to calibrate the uncertain model parameters based on the gathered evidence by testing. An overview on the statistical machinery used to perform this activity follows. Formally, we let the parameters of the categorical distribution describing θ_i be defined by a Dirichlet Prior as follows:

$$(x_{i,j}, \dots, x_{i,k}) \sim Dir(\alpha_i), \text{ with } \alpha_i = (\alpha_{i,j}, \dots, \alpha_{i,k}) \tag{8}$$

Values in α_i are the hyper-parameters of the Prior. Based on this formulation, we can learn the uncertain parameters by belief monitoring during the testing process. For each executed test, the Prior probability is updated based on the experience using Bayes' theorem instroduced in Section 3. In our context, belief monitoring can be efficiently performed since the Prior and the Posterior belong to the same family of distributions (i.e., Equation 8 representes a conjugate Prior). Namely, the conformance game keeps track of the number of occurrences $n_{i,j}$ that represents how many times the transition (s_i, s_j) has been observed for all θ_i and uncertain parameter $x_{i,j}$. Thus, given a sequence of observations y (i.e., the runtime evidence), the Posterior is defined as follows.

$$(x_{i,j}, \ldots, x_{i,k})|y \sim Dir(\alpha'_i), \text{ with } \alpha'_i = (\alpha_{i,j} + n_{i,j}, \ldots, \alpha_{i,k} + n_{i,k}) \tag{9}$$

For instance, considering the SafeHome system, we can compute the Posterior of θ_1 by updating the hyper-parameters $\alpha'_1 = (47 + 93, 2 + 2, 1 + 5)$ if we observe: $93 \cdot s_2$, $2 \cdot s_3$, and $5 \cdot s_5$, as outcome of 100 invocations of the action *init* from state s_1. By summarizing the Posterior with the HDR we obtain the following updated bounds.

$$\begin{aligned} HDR[f(\theta_1)] &= x_{1,2} \in [0.87, 0.99], x_{1,3} \in [0.00, 0.09], x_{1,5} \in [0.00, 0.05] \\ HDR[f(\theta_1|y)] &= x_{1,2} \in [0.89, 0.97], x_{1,3} \in [0.00, 0.05], x_{1,5} \in [0.01, 0.07] \end{aligned} \tag{10}$$

Uncertainty-aware test case generation. We introduce here the test case generation strategy used by our online MBT algorithm. This strategy leverages the notion of *uncertainty-aware* reward structure, motivated by the practical need of identifying those actions that increase the likelihood of testing uncertain regions (i.e., transitions annotated with uncertain parameters). In fact, our goal is to equip MBT with the ability of stressing the uncertain components of the SUT. The uncertainty-aware reward structure is formally defined as follows.

Definition 1 (uncertainty-aware reward structure). *Given a pMDP model $\mathcal{M}$, an uncertain region θ_i, and two numeric values $r_h, r_l \in \mathbb{N}_{>0}$ s.t. $r_h \gg r_l$, the* uncertainty-*aware reward structure u is defined as follows:*

$$u^a_{i,j} = \begin{cases} r_h & p^a_{i,j} > 0 \textit{ and } \exists \theta_i \textit{ for } (s_i, a) \\ r_l & \textit{otherwise} \end{cases}$$

The rationale is to assign a high reward r_h to transitions associated to uncertain parameters and a low reward r_l elsewhere. Then we use the *uncertainty*-aware reward structure to automatically compute the best exploration policy (Equation 1) that maximizes the expected cumulated uncertainty-aware rewards. Intuitively, given θ_i, the best policy π^*_i drives a decision maker optimally towards the uncertain model region θ_i.

Indeed, multiple uncertain regions determine alternative testing scenarios targeting different portions of the model. The way we sample from available choices (i.e., actions provided by alternative best policies) determines the whole test case generation strategy. Our strategy provides control over test scenarios during MBT based on a probabilistic function as defined below.

Definition 2 (HDR-aware strategy). *Given a pMDP model $\mathcal{M}$ a set of uncertain regions $\theta_1, \dots, \theta_k$ and related best policies $\pi_1^*, \dots, \pi_k^*$, the* HDR-aware *test case generation strategy is defined by the following partial probabilistic function:*

$$\mathcal{P}(a|s) = \begin{cases} 0 & \omega(s,a') = 0 \\ \omega(s,a)/\sum_{a' \in A(s)} \omega(s,a') & \textit{otherwise} \end{cases} \tag{11}$$

where ω represents a per-state weight function that maps a state and an action to a value in $\mathbb{R}_{\geq 0}$ such that:

$$\omega(s,a) = \max_{i:\pi_i^*(s)=a} \|HDR[f(\theta_i|y)]\| \tag{12}$$

Intuitively, the weight function selectively increases or decreases the probability of certain actions based on the magnitude of the Posteriors associated with uncertain regions. As anticipated in Section 3, Bayesian statistics uses the magnitude of the HDR as a measure of the degree of confidence in the inference process: the smaller the magnitude, the higher the confidence. Thus, here we leverage this measure to selectively increase the probability of actions that drive testing towards regions associated with a lower degree of confidence.

Termination condition. Our approach uses the ability of detecting violation of requirements to build the termination condition for the online MBT algorithm. In particular, since the testing activity incrementally builds the Posterior knowledge of each θ_i, we can iteratively compare the summarization of the Posteriors and the (pre-computed) hyper-rectangles encoding verification conditions. Thus, the termination of the MBT can be formalized by means of the following two alternative cases.

Definition 3 (Successful run). *Given a pMDP $\mathcal{M}$, a set of hyper-rectangles H, and the Posterior $f(\theta_i|y)$ for all i, we say that the MBT is* succesful *iff there exists $h \in H$, $[l_x, u_x] \in h$ s.t. $[l_x, u_x] \supseteq HDR[f_x(\theta_i|y)]$ for all i and $x \in \theta_i$.*

Definition 4 (Failing run). *Given a pMDP $\mathcal{M}$, a set of hyper-rectangles H, and the Posterior $f(\theta_i|y)$ for all i, we say that the MBT is* failing *iff for all $h \in H$ there exists i, $x \in \theta_i$ s.t. $[l_x, h_x] \cap HDR[f_x(\theta_i|y)] = \emptyset$, with $[l_x, h_x] \in h$.*

The intuitive meaning of these two cases follows. The testing activity can terminate by either confirming that requirements are met (i.e., successful run) or identifying violated requirements (i.e., failing run) based on the observed evidence. In the first case, the Posterior knowledge tells that the model $\mathcal{M}[val]$ satisfies requirements for all instantiation *val* constructed by using the Posteriors' HDR. In fact, all the values that can be drawn from the HDRs meet the pre-computed verification conditions. In the latter case, the verification conditions cannot be satisfied because the HDR identifies disjoint intervals with respect to pre-computed hyper-rectangles.

It is worth noting that Definition 3 and Definition 4 identify two disjoint conditions. Furthermore, termination by satisfying one of the two conditions is always guaranteed because of the asymptotic behavior of the Posterior in the limit of infinite observations. Loosely, if consistent estimates are available, then Bayesian inference is consistent [17]. Moreover, the Posterior converges to a distribution independent of the initial Prior if the random variable in consideration has a finite probability space [18].

6 Empirical Evaluation

We introduce our research questions and design of the evaluation in Section 6.1; we present the results in Section 6.2; we finally discuss threats to validity in Section 6.3.

6.1 Research Questions and Design

The main goal of the evaluation is to investigate the cost-effectiveness of our testing method. The cost (or effort) refers to the number of tests required to achieve termination. The effectiveness here represents the ability of identifying requirements violations. Thus, we aim at answering three research questions:

RQ1: Is the approach able to detect requirements violations?
RQ2: What is the cost required by our testing method to achieve termination?
RQ3: How does our approach compare with existing MBT methods in terms of cost-effectiveness?

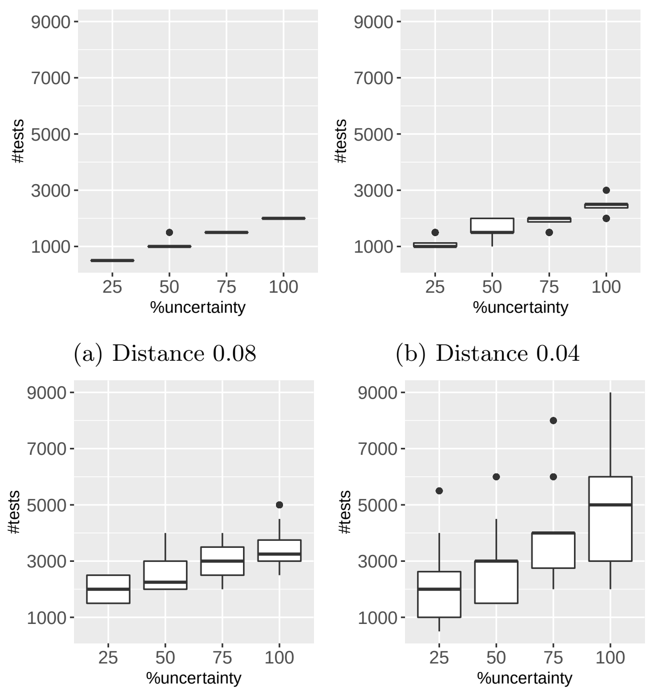
We addressed these questions by conducting an extensive testing campaign using different versions of the SafeHome as SUT. In particular, we considered variations taking control over two main factors of interest: *degree of uncertainty* in terms of percentage of uncertain model parameters (varying from 25% to 100%); and *distance* between actual values of model parameters and verification conditions given by hyper-rectangles (varying from 0.01 to 0.16). We have tested each version of the SUT by using the approach presented in this paper and we compared its cost-effectiveness with a traditional *Random* MBT approach [19], and also an existing *uncertainty*-aware MBT method called *Flat* [16].

For all experiments, we measured the number of tests spent for each uncertain region (i.e., the cost) and the failure detection capability (i.e., the effectiveness). Hereafter we discuss the most relevant results and we refer the reader to our implementation and dataset for the replicability of the experiments[1].

6.2 Results

Results for RQ1. We addressed this question by assessing the ability of exhibiting failing runs when the actual values of uncertain parameters fall outside the boundaries of the hyper-rectangles. Thus, we took control over actual values

[1] The MBT module is open source software publicly available at `https://github.com/SELab-unimi/mbt-module`. A replication package of the experiments is available at `https://github.com/SELab-unimi/sefm2020-replication-package`.



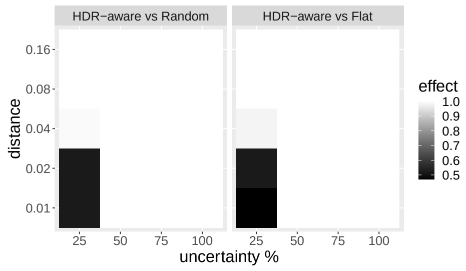
(a) Distance 0.08 (b) Distance 0.04

(c) Distance 0.02 (d) Distance 0.01

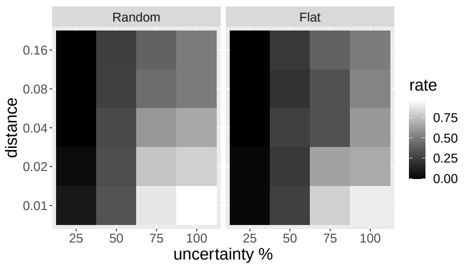
Fig. 4: Effort by varying the distance and the degree of uncertainty.

and hyper-rectangles and then we executed the HDR-aware MBT 100 times for each combination of degree of uncertainty, distance, and truth value (i.e., either true or false w.r.t. requirements violation). From the results, we observe that, in all cases, the process effectively returns the expected outcome with bounded effort. Namely, the MBT process terminated with no more than 9000 tests, representing the effort value measured in the worst case (i.e., 100% uncertainty and 0.01 distance). Further details on the evaluation of the effort follow.

Results for RQ2. Figure 4 shows the effort required by the HDR-aware MBT. Each box-plot has been built considering 100 runs for each degree of uncertainty and a specific distance value. Since the case “0.16 distance” is comparable to the 0.08 one, it has been omitted. From the data reported by each individual plot, we observe that the effort increases linearly with the degree of uncertainty. Namely, for each distance value we assessed the existence of a linear dependency between degree of uncertainty and number of tests with correlation value greater than 0.5. The slope of the estimated lines varies from 16.4 to 37.5. The shorter the distance the steeper the growth. This means that parameters close to hyper-rectangle borders are likely to increase the effort when the number of uncertain parameters increases. Furthermore, we observe that the variability of the effort values increases when reducing the distance value. The average



(a) $\hat{A}_{12}$ effect size



(b) Error rate

Fig. 5: Effectiveness comparison among HDR-aware, Random, and Flat.

variance of the effort is 2998.7, 62815.7, 355587.1, and 2433554.3, for distance values 0.08, 0.04, 0.02, and 0.01, respectively. In general, we can observe that by halving the distance value, we increase the variance of an order of magnitude. The interpretation of this result tells that the closer the actual values to the hyper-rectangles, the lower the predictability of the effort.

Results for RQ3. To address this research question, we first computed the effort required by selected (existing) MBT strategies and then we compared it with the effort required by the HDR-aware MBT by using the standardized non-parametric Vargha and Delaney's $\hat{A}_{12}$ effect size measure [20]. In this context, the $\hat{A}_{12}$ encodes the probability that running the HDR-aware MBT yields less effort than the Random and the Flat MBT strategies. Figure 5a shows a heat map of the effect size by varying uncertainty and distance factors. The two plots refer to the pairwise comparisons: HDR-aware versus Random (left); and HDR-aware versus Flat (right). Data shows that Random and Flat are similar, i.e., the $\hat{A}_{12}$ effect size values are comparable. In general, the probability of observing less effort when adopting the HDR-aware MBT is high. It represents the certain event for almost all cases. We observe outliers when the degree of uncertainty is low (25%) and the distance is short (less than 0.04). In this settings, all the strategies (HDR-aware, Random, and Flat) are likely to exhibit the same effort.

The effectiveness has been assessed by executing all the selected strategies by assuming the same total amount of effort. Namely, we fixed the limit according to the maximum effort required by the HDR-aware method. The rationale is to execute both Random and Flat by using an effort value that ensures termination of the HDR-aware method with the correct outcome. At the end of each execution we summarized the Posteriors to determine whether the run succeeded or failed with respect to pre-computed hyper-rectangles. Thus, we counted the number of wrong outcomes to measure the verification error rate when using either Random or Flat instead of HDR-aware. Figure 5b shows a heat map of the error rate by varying uncertainty and distance factors. The two plots refer to the errors obtained by using: Random (left); and Flat (right). Data shows that both approaches are likely to terminate by exhibiting a wrong outcome. Thus, the effectiveness of both Random and Flat is lower w.r.t. the HDR-aware method.

The two plots show a comparable pattern: the error rate increases when the distance decreases and the uncertain parameters increase. In both cases we observe the worst effectiveness values with distance equal to 0.01 and uncertainty equal to 100%. In this setting, the error rate values are 0.96 and 0.90 for Random and Flat methods, respectively.

6.3 Threats to Validity

To limit threats to *external validity* we conducted a testing campaign on several versions of the SafeHome by taking control and detailing the important factors of interest (i.e., uncertain regions, uncertainty degree, and distance) for all executed experiments. Further generalization of our findings to different application domains and model sizes requires additional experiments. We dealt with *internal validity* threats in the empirical study by directly manipulating independent variables. Namely, we have direct assess to both actual values of uncertain parameters and design-time beliefs expressed by Priors. It is worth noting that direct manipulation of these factors has been crucial to assess cause-effect between them and the cost-effectiveness of our approach. Such a fine grained access to independent variables allows the internal validity to be increased with respect to conclusions based on an association observed without manipulation. Direct manipulation permits the creation of the same experimental conditions within repeated runs. We addressed threats to *conclusion validity* by reducing the possibility of obtaining results by chance. We repeated experiments 100 for each SUT variant and for each testing method. Then, we followed the guidelines introduced in [20] to detect statistical difference. Namely, we conducted a pairwise comparison among testing methods using the Mann-Whitney U test to calculate the p-value with significance level $\alpha = 0.05$. We also detected a *practical value* using the standardized Vargha and Delaney's $\hat{A}_{12}$ non-parametric effect size measure. We handled major threats to *construct validity* by assessing the validity of the metrics adopted in our experiments. In particular, the cost has been assessed by considering the number of tests required to achieve termination. This represents a traditional metric in assessing randomized testing algorithms [20]. The effectiveness has been measured by verifying the ability of identifying failures (i.e., requirements violation in our context). In search-based testing, this represents a traditional measure to assess the effectiveness.

7 Related Work

A taxonomy of potential sources of uncertainty affecting the development of software systems is presented in [21]. Uncertainty is categorized at different stages such as requirements, design, and production. Testing in this work is almost neglected. Further effort in categorizing and guiding software engineers in recognizing different types of uncertainty has been presented in [22]. Probabilistic models have been adopted extensively to model and analyze uncertainties in the context of self-adaptive systems. The approach introduced in [23] continuously updates transition probabilities of discrete time Markov models using efficient runtime monitoring. Another approach, introduced in [12], describes runtime quantitative verification and sensitivity analysis to support adaptation in order to achieve perpetual meeting of nonfunctional requirements. Queueing networks have been adopted and extended in [24] with adaptation knobs to dynamically fulfill performance goals. All these lines of research aim at modifying probabilistic models to react to uncertain changes during operation. In [25] MDPs are extended by attaching confidence intervals to transition probabilities in order to compute Pareto optimal policies. Our approach does not apply multi-objective optimization but uses uncertainty to drive MBT.

Recent research activities show increasing effort in delivering approaches and techniques that jointly consider testing and uncertainty quantification methods. Uncertainty sampling has been introduced in [26] to generate suitable test data. Namely, a “Query Strategy Framework” is adopted to infer a behavioral model and then select those tests on which the behavior of the system is uncertain. This approach outperforms conventional and adaptive random testing at exposing faults. A so called uncertainty-wise UML-based modeling framework has been introduced in [4] with the aim of creating models that can be executed to test CPSs. A offline MBT approach that leverages on the uncertainty-wise modeling framework has been presented in [5]. The approach generates test cases in a cost-effective way by minimizing the number of tests but maintain coverage of models. The approach presented in [2,16] incorporates uncertainty mitigation into an online MBT framework. Nevertheless, requirements are neglected during the testing process which is guided by coarse information of uncertain components.

To summarize, the methodology introduced in this paper differs from the state-of-the-art since it combines offline analysis of parametric variability of uncertain model beliefs and online MBT to detect requirements violations.

8 Conclusion

This paper introduces a novel approach to online model-based testing under uncertainty encoded as parameters of a Markov Decision process. We use design-time parametric model checking and then online testing algorithms to gather runtime evidence and detect requirements violations. The design-time phase analyzes the parameter space to pre-compute the constraints for requirements satisfaction. The online testing activity feeds a Bayesian inference process able to detect violations of pre-computed constraints. We provided a theoretical discussion of the approach and the description of an empirical evaluation aiming at assessing its cost-effectiveness. During the evaluation we conducted a large testing campaign using a number of variations of the SafeHome case study. The experience collected during our experiments suggests that the approach is able to spot requirements violations with bounded effort. The HDR-aware method outperforms both the Random and the Flat strategies considered in our quantitative comparison. Our testing framework has been released publicly to encourage adoption and repetition of experiments.

References

1. N. Esfahani and S. Malek, "Uncertainty in self-adaptive software systems," in *Software Engineering for Self-Adaptive Systems II: Int. Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 214–238.
2. M. Camilli, C. Bellettini, A. Gargantini, and P. Scandurra, "Online model-based testing under uncertainty," in *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*, Oct 2018, pp. 36–46.
3. M. Camilli, A. Gargantini, R. Madaudo, and P. Scandurra, "Hyppotest: Hypothesis testing toolkit for uncertain service-based web applications," in *Integrated Formal Methods*, W. Ahrendt and S. L. Tapia Tarifa, Eds. Cham: Springer International Publishing, 2019, pp. 495–503.
4. M. Zhang, S. Ali, T. Yue, R. Norgren, and O. Okariz, "Uncertainty-wise cyber-physical system test modeling," *Software & Systems Modeling*, Jul 2017. [Online]. Available: https://doi.org/10.1007/s10270-017-0609-6
5. M. Zhang, S. Ali, and T. Yue, "Uncertainty-wise test case generation and minimization for cyber-physical systems," *Journal of Systems and Software*, vol. 153, pp. 1–21, 2019. [Online]. Available: https://doi.org/10.1016/j.jss.2019.03.011
6. E. M. Hahn, H. Hermanns, B. Wachter, and L. Zhang, "Param: A model checker for parametric markov models," in *Computer Aided Verification*, T. Touili, B. Cook, and P. Jackson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 660–664.
7. M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, and A. Pretschner, *Model-Based Testing of Reactive Systems: Advanced Lectures (Lecture Notes in Computer Science).* Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.
8. M. Camilli, A. Gargantini, P. Scandurra, and C. Bellettini, "Towards inverse uncertainty quantification in software development (short paper)," in *Software Engineering and Formal Methods - 15th International Conference, SEFM 2017, Trento, Italy, September 4-8, 2017, Proceedings*, ser. Lecture Notes in Computer Science, A. Cimatti and M. Sirjani, Eds., vol. 10469. Springer, 2017, pp. 375–381. [Online]. Available: https://doi.org/10.1007/978-3-319-66197-1_24
9. C. P. Robert, *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation*, 2nd ed. Springer, May 2007.
10. V. Forejt, M. Kwiatkowska, G. Norman, and D. Parker, "Automated verification techniques for probabilistic systems," in *Formal Methods for Eternal Networked Software Systems: 11th International School on Formal Methods for the Design of Computer, Communication and Software Systems, SFM 2011, Bertinoro, Italy, June 13-18, 2011. Advanced Lectures.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 53–113.
11. M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 1994.
12. A. Filieri, G. Tamburrelli, and C. Ghezzi, "Supporting self-adaptation via quantitative verification and sensitivity analysis at run time," *IEEE Transactions on Software Engineering*, vol. 42, no. 1, pp. 75–99, Jan 2016.
13. C. Courcoubetis and M. Yannakakis, "The complexity of probabilistic verification," *J. ACM*, vol. 42, no. 4, p. 857–907, Jul. 1995. [Online]. Available: https://doi.org/10.1145/210332.210339
14. E. M. Hahn, T. Han, and L. Zhang, "Synthesis for pctl in parametric markov decision processes," in *Proceedings of the Third International Conference on NASA*

Formal Methods, ser. NFM'11. Berlin, Heidelberg: Springer-Verlag, 2011, p. 146–161.

15. J. Berger, *Statistical Decision Theory and Bayesian Analysis*, ser. Springer Series in Statistics. Springer, 1985.
16. M. Camilli, A. Gargantini, and P. Scandurra, "Model-based hypothesis testing of uncertain software systems," *Software Testing, Verification and Reliability*, vol. 30, no. 2, p. e1730, 2020, e1730 stvr.1730. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.1730
17. J. L. Doob, "Application of the theory of martingales," in *Actes du Colloque International Le Calcul des Probabilités et ses applications*, 1949, pp. 23–27.
18. D. A. Freedman, "On the asymptotic behavior of bayes estimates in the discrete case ii," *The Annals of Mathematical Statistics*, vol. 36, no. 2, pp. 454–456, 1965.
19. M. Veanes, C. Campbell, W. Schulte, and N. Tillmann, "Online testing with model programs," in *Proceedings of the 10th European Software Engineering Conf. / 13th ACM Int. Symp. on Foundations of Software Engineering*, 2005, pp. 273–282.
20. A. Arcuri and L. Briand, "A practical guide for using statistical tests to assess randomized algorithms in software engineering," in *Proceedings of the 33rd International Conference on Software Engineering*, ser. ICSE '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 1–10. [Online]. Available: https://doi.org/10.1145/1985793.1985795
21. A. J. Ramirez, A. C. Jensen, and B. H. C. Cheng, "A taxonomy of uncertainty for dynamically adaptive systems," in *Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, ser. SEAMS '12. Piscataway, NJ, USA: IEEE Press, 2012, pp. 99–108. [Online]. Available: http://dl.acm.org/citation.cfm?id=2666795.2666812
22. D. Perez-Palacin and R. Mirandola, "Uncertainties in the modeling of self-adaptive systems: A taxonomy and an example of availability evaluation," in *Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering*, ser. ICPE '14. New York, NY, USA: ACM, 2014, pp. 3–14. [Online]. Available: http://doi.acm.org/10.1145/2568088.2568095
23. A. Filieri, L. Grunske, and A. Leva, "Lightweight adaptive filtering for efficient learning and updating of probabilistic models," in *Proceedings of the 37th International Conference on Software Engineering - Volume 1*, ser. ICSE '15. Piscataway, NJ, USA: IEEE Press, 2015, pp. 200–211. [Online]. Available: http://dl.acm.org/citation.cfm?id=2818754.2818781
24. E. Incerto, M. Tribastone, and C. Trubiani, "Software performance self-adaptation through efficient model predictive control," in *International Conference on Automated Software Engineering*, 2017, pp. 485–496.
25. D. Scheftelowitsch, P. Buchholz, V. Hashemi, and H. Hermanns, "Multi-objective approaches to markov decision processes with uncertain transition parameters," in *International Conference on Performance Evaluation Methodologies and Tools*, 2017, pp. 44–51.
26. N. Walkinshaw and G. Fraser, "Uncertainty-driven black-box test data generation," in *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)*, March 2017, pp. 253–263.