Adaptive Model Learning for Continual Verification of Non-Functional Properties

Radu Calinescu
Department of Computer Science
University of York, UK
radu.calinescu@york.ac.uk

Yasmin Rafiq
Department of Computer Science
University of York, UK
yr534@york.ac.uk

Kenneth Johnson
Department of Computer Science
University of York, UK
kenneth.johnson@york.ac.uk

Mehmet Emin Bakır
Department of Computer Science
University of York, UK
meb524@york.ac.uk

ABSTRACT

A growing number of business and safety-critical services are delivered by computer systems designed to reconfigure in response to changes in workloads, requirements and internal state. In recent work, we showed how a formal technique called continual verification can be used to ensure that such systems continue to satisfy their reliability and performance requirements as they evolve, and we presented the challenges associated with the new technique. In this paper, we address important instances of two of these challenges, namely the maintenance of up-to-date reliability models and the adoption of continual verification in engineering practice. To address the first challenge, we introduce a new method for learning the parameters of the reliability models from observations of the system behaviour. This method is capable of adapting to variations in the frequency of the available system observations, yielding faster and more accurate learning than existing solutions. To tackle the second challenge, we present a new software engineering tool that enables developers to use our adaptive learning and continual verification in the area of service-based systems, without a formal verification background and with minimal effort.

Categories and Subject Descriptors

D.2.4 [**Software Engineering**]: Software/program verification—*model checking; reliability; statistical methods*

Keywords

on-line model learning, runtime quantitative verification, discrete-time Markov models, service-based systems

ICPE'14, March 22–26, 2014, Dublin, Ireland.

http://dx.doi.org/10.1145/2568088.2568094.

1. INTRODUCTION

Rarely a day passes without new announcements of yet more applications being "moved to the cloud" or "running on the Internet-of-Things" in the name of increased flexibility, richer functionality, or cost and energy savings. Nevertheless, few of these announcements mention the dependability and performance implications of such long-reaching decisions. As the applications involved increasingly include services such as UK Government ICT procurement[1], New York Stock Exchange market data analysis[2] and US Department of Defence solutions[3], this raises serious concerns.

In recent work, we advocated the continual formal verification of the non-functional properties (NFPs) of such *evolving critical systems* [2, 3], and devised theoretical and practical tools supporting the approach [4, 6, 7, 21]. These tools employ established or new, lower-overhead model checking techniques to assess whether the quality-of-service (QoS) requirements of a system continue to be satisfied as the system evolves. The approach has been used successfully in applications including QoS management and optimisation of service-based systems [4, 6], and reliability NFP analysis for cloud computing infrastructure [7, 21].

As part of the aforementioned work, we identified the key research challenges that need to be addressed in order to extend the applicability of continual verification, and to support its adoption in QoS engineering practice [3]. In this paper, we propose solutions that tackle important instances of two of these challenges. The first challenge is the maintenance of an up-to-date QoS model of an evolving critical system. The new on-line model learning method presented in the paper addresses this challenge for discrete-time Markov chains (DTMCs), such as those used for the continual verification of the reliability NFPs of evolving critical systems (e.g., [4, 8, 12]). The second challenge that we tackle is the adoption of continual verification in QoS engineering practice. We introduce a new software engineering tool that uses our model learning method, and contributes to enabling practitioners to exploit continual NFP verification with lim-

[1] `http://gcloud.civilservice.gov.uk`
[2] `http://www.nyse.com/press/1306838249812.html`
[3] `http://aws.amazon.com/federal/`

ited effort and without formal verification expertise. The main contributions of the paper are:

1. A parameterised on-line learning method that infers the state transition probabilities of a DTMC model of a system from observations of the system behaviour, and adjusts its parameters dynamically depending on the frequency of these observations. This *adaptive learning* leads to a faster and more accurate inference of the transition probabilities than that provided by existing methods.
2. Rigorous theoretical results linking the parameters chosen dynamically by our learning method to the expected error in the accuracy of the learnt state transition probabilities. This allows the configuration of the adaptive learning method so that it yields results within an acceptable expected error range.
3. A software-as-a-service (SaaS) development tool that automatically generates web service proxies which use our adaptive learning method to support continual reliability NFP verification in service-based systems. The new tool is freely available as an Amazon Machine Image (AMI) that service-based system developers can use with no installation or configuration effort.
4. The integration of the SaaS tool with our existing COntinual VErification (COVE) framework from [6]. The integrated toolset supports the end-to-end development of reconfigurable service-based systems that take advantage of the results introduced in this paper with minimal practitioner effort and formal verification expertise.

The paper is organised as follows. Section 2 introduces concepts and notation used throughout the rest of the paper. Section 3 presents our adaptive model learning method, and several experiments used to evaluate and compare its effectiveness with that of related approaches. Next, Section 4 describes our software-as-a-service engineering tool that allows developers of service-based systems to take advantage of the new learning technique. A case study from the telehealth application domain is used to demonstrate the effectiveness of this tool in Section 5, and related work is discussed in Section 6. Section 7 concludes the paper with a brief summary and an overview of future work directions.

2. BACKGROUND

2.1 Quantitative Verification of Discrete-Time Markov Chains

DEFINITION 1. *A cost-annotated discrete-time Markov chain (DTMC) is a tuple*

$$M = (S, s_0, P, L, c), \tag{1}$$

where:

- S *is a finite set of states;*
- $s_0 \in S$ *is the initial state;*
- P *is an* $|S| \times |S|$ *transition probability matrix;*
- $L : S \to 2^{AP}$ *is a labelling function which assigns a set of atomic propositions from* AP *to each state in* S*;*
- $c : S \to \mathbb{R}_+$ *is a costing function that associates a cost* $c(s) \geq 0$ *with each state* $s \in S$*.*

For any states $s_i, s_j \in S$*, the element* p_{ij} *from* P *represents the probability of transitioning to state* s_j *from state* s_i*, and* $\sum_{s_j \in S} p_{ij} = 1$*.*

Quantitative or *probabilistic* model checkers (e.g., PRISM [24], MRMC [22] and Ymer [31]) operate on Markovian models expressed in a high-level, state-based language. Given a DTMC description in this language, the low-level representation (1) is derived automatically. Our work uses the probabilistic model checker PRISM [24], which supports the analysis of DTMC properties specified in a cost/reward-augmented version of probabilistic computational tree logic (PCTL) [20], whose syntax is defined below.

DEFINITION 2. *Let* AP *be a set of atomic propositions and* $a \in AP$*,* $p \in [0, 1]$*,* $k \in \mathbb{N}$*,* $r \in \mathbb{R}$ *and* $\bowtie \in \{\geq, >, <, \leq\}$*. Then a state-formula* Φ *and a path formula* Ψ *in PCTL are defined by the following grammar:*

$$\begin{aligned} \Phi &::= true | a | \Phi \wedge \Phi | \neg\Phi | \mathrm{P}_{\bowtie p}(\Psi) \\ \Psi &::= X\Phi | \Phi\, U \Phi | \Phi\, U^{\leq k} \Phi \end{aligned} \tag{2}$$

and the cost/reward augmented PCTL state formulae are defined by the grammar:

$$\Phi ::= \mathrm{R}_{\bowtie r}[\mathrm{I}^{=k}] | \mathrm{R}_{\bowtie r}[\mathrm{C}^{\leq k}] | \mathrm{R}_{\bowtie r}[\mathrm{F}\Phi]. \tag{3}$$

PCTL distinguishes between state and path formulae. The state formulae include the standard logical operators $\wedge$ and $\neg$, which also allow a formulation of other usual logical operators (disjunction ($\vee$), implication ($\Rightarrow$), etc.) and *false*. The main extension of the state formulae, compared to non-probabilistic logics, is the replacement of the traditional path quantifiers $\exists$ and $\forall$ with a probabilistic operator $\mathcal{P}$. This operator defines upper or lower bounds on the probability of the system evolution. As an example, the formula $\mathcal{P}_{\geq p}(\Psi)$ is true at a given state if the probability of the future evolution of the system satisfying Ψ is at least p. The path formulae that can be used with the probabilistic path operator are:

- the "next" formula $X\Phi$, which holds if Φ is true in the next state of a path;
- the time bounded "until" formula $\Phi_1\, U^{\leq k} \Phi_2$, which requires that Φ_1 holds continuously up to some time step $x < k$ and Φ_2 becomes true at time step $x + 1$;
- unbounded "until" formula $\Phi_1\, U \Phi_2$, whose semantics is identical with that of the bounded "until", but the time-step bound is set to infinity $t = \infty$.

Finally, the cost/reward operator R can be used to analyse the expected cost at timestep k ($\mathrm{R}_{\bowtie r}[\mathrm{I}^{=k}]$), the expected cumulative cost up to time step k ($\mathrm{R}_{\bowtie r}[\mathrm{C}^{\leq k}]$), and the expected cumulative cost to reach a future state that satisfies a property Φ ($\mathrm{R}_{\bowtie r}[\mathrm{F}\Phi]$).

The semantics of the PCTL is defined with a satisfaction relation $\models$ over the states S and possible paths $Path^M(s)$ that are possible in a state $s \in S$ of a model M with the structure from (1). Further details about the formal semantics of PCTL are available from [11, 20].

2.2 On-line Learning of DTMC Transition Probabilities

The DTMC modelling and analysis formalism from the previous section is traditionally used for the offline verification of non-functional system properties. To extend its

applicability to continual verification, the DTMC model it relies upon must be updated permanently, so that it is maintained in sync with the changing behaviour of the continually verified system. Typically, this model updating involves monitoring the evolving system, and using the observations obtained in this way to learn about any changes in the DTMC transition probabilities P from (1).

A basic Bayesian on-line learning method that can be used for this purpose was proposed in [12], and extended by our work-in-progress results from [5]. This section summarises the extended on-line learning method from [5], which is used as a basis for the adaptive learning method proposed in this paper and described in detail in Section 3.

The algorithm we introduced in [5] learns the transition probabilities p_{ij} of a DTMC model M with the form in (1), starting from *a priori* estimates p_{ij}^0 and the observations of the last $k \geq 1$ system transitions from state s_i to states $s_j \in S$. Assuming that the l-th observation of a transition from state s_i, $1 \leq l \leq k$, is a transition to state $s_{j_l} \in S$, we define

$$x_{ij}^l = \begin{cases} 1 & \text{if } j_l = j, \\ 0 & \text{otherwise} \end{cases}, \tag{4}$$

and we calculate the estimate probability of a state transition from s_i to s_j after the k-th observation as

$$p_{ij}^k = \frac{c_i^0}{c_i^0 + k} p_{ij}^0 + \frac{k}{c_i^0 + k} \frac{\sum_{l=1}^k w_i^l x_{ij}^l}{\sum_{l=1}^k w_i^l}, \tag{5}$$

where $c_i^0 > 0$ is a *smoothing parameter* that quantifies the confidence in the accuracy of p_{ij}^0, and $w_i^l \in (0, 1]$ is a weight that reflects the *age* of the l-th observation. We showed in [5] that an effective choice of weights is

$$w_i^l = \alpha_i^{-(t_k - t_l)}, \tag{6}$$

where t_l, $1 \leq l \leq k$, represents the timestamps of the l-th observation, and $\alpha_i \geq 1$ is an *ageing parameter*. As we showed in [5], the learning algorithm (5)–(6) has two key advantages over other learning techniques. First, the weights w_i^l decrease the impact of old observations on the estimates p_{ij}^k, significantly speeding up the detection of sudden changes in actual transition probabilities (e.g., due to failures of system components), in particular when such changes occur after long periods of relatively constant behaviour. Second, reorganising the terms in (5) allows p_{ij}^k to be calculated from p_{ij}^{k-1} in O(1) time and using O(1) memory, a key advantage for an on-line learning algorithm.

3. ADAPTIVE DTMC MODEL LEARNING

Our experiments from [5] show that the effectiveness of the transition-probability learning algorithm (5)–(6) depends on the choice of the parameters c_i^0 and α_i, and that no combination of values for these parameters is suitable for all scenarios. To address this limitation, we extend the learning algorithm with the ability to select suitable parameters c_i^0 and α_i at runtime. The dynamic selection of these parameters adapts the learning algorithm to the frequency of the observations, and is based on the following theoretical results.

PROPOSITION 1. *Let x_1, x_2, …, x_k be an independent trials process with expected value $E(x_l) = \mu$ and variance $V(x_l) = \sigma^2$, for $l = 1, 2, \ldots, k$. Let $w_1, w_2, \ldots, w_k > 0$ be a set of weights, and $A_k = \frac{\sum_{l=1}^k w_l x_l}{\sum_{l=1}^k w_l}$ be the weighted average of x_1, x_2 …, x_k. Then*

$$E(A_k) = \mu \quad \textit{and} \quad V(A_k) = \frac{\sum_{l=1}^k (w_l)^2}{\left(\sum_{l=1}^k w_l\right)^2} \sigma^2. \tag{7}$$

Proof: The expected value $E(A_k)$ can be calculated as

$$\begin{aligned} & E\left(\frac{\sum_{l=1}^k w_l x_l}{\sum_{l=1}^k w_l}\right) = \frac{E\left(\sum_{l=1}^k w_l x_l\right)}{\sum_{l=1}^k w_l} = \\ & \quad \text{(since } \textstyle\sum_{l=1}^k w_l \text{ is a constant [19, Theorem 6.2])} \\ & = \frac{\sum_{l=1}^k E(w_l x_l)}{\sum_{l=1}^k w_l} = \\ & \quad \text{(since } w_1 x_1, w_2 x_2, \ldots, w_k x_k \text{ are random variables} \\ & \qquad \text{with finite expected values [19, Theorem 6.2])} \\ & = \frac{\sum_{l=1}^k w_l E(x_l)}{\sum_{l=1}^k w_l} = \frac{\sum_{l=1}^k w_l \mu}{\sum_{l=1}^k w_l} = \mu \\ & \quad \text{(since } w_l \text{ is a constant [19, Theorem 6.2]).} \end{aligned}$$

In a similar way, the variance $V(A_k)$ is given by

$$\begin{aligned} & V\left(\frac{\sum_{l=1}^k w_l x_l}{\sum_{l=1}^k w_l}\right) = \frac{V\left(\sum_{l=1}^k w_l x_l\right)}{\left(\sum_{l=1}^k w_l\right)^2} = \\ & \quad \text{(since } \textstyle\sum_{l=1}^k w_l \text{ is a constant [19, Theorem 6.7])} \\ & = \frac{\sum_{l=1}^k V(w_l x_l)}{\left(\sum_{l=1}^k w_l\right)^2} = \\ & \quad \text{(since } w_1 x_1, w_2 x_2, \ldots, w_k x_k \text{ are independent} \\ & \qquad \text{random variables [19, Theorem 6.8])} \\ & = \frac{\sum_{l=1}^k (w_l)^2 V(x_l)}{\left(\sum_{l=1}^k w_l\right)^2} = \frac{\sum_{l=1}^k (w_l)^2 \sigma^2}{\left(\sum_{l=1}^k w_l\right)^2} = \frac{\sum_{l=1}^k (w_l)^2}{\left(\sum_{l=1}^k w_l\right)^2} \sigma^2 \\ & \quad \text{(since } w_l \text{ is a constant [19, Theorem 6.7]).} \end{aligned}$$

□

COROLLARY 1. *Consider again the independent trials process x_1, x_2, …, x_k from Proposition 1, and let $\epsilon > 0$. Then*

$$P\left(\left|\frac{\sum_{l=1}^k w_l x_l}{\sum_{l=1}^k w_l} - \mu\right| \geq \epsilon\right) \leq \frac{\sum_{l=1}^k (w_l)^2}{\left(\sum_{l=1}^k w_l\right)^2 \epsilon^2} \sigma^2. \tag{8}$$

Proof: The result is a direct application of Chebyshev's Inequality (e.g., [19, Theorem 8.1]) to the discrete random variable A_k with the expected value and variance from (7). □

PROPOSITION 2. *Consider the transition-probability learning algorithm (5)–(6), and let $\epsilon > 0$. Then*

$$P\left(\left|\frac{\sum_{l=1}^k w_i^l x_{ij}^l}{\sum_{l=1}^k w_i^l} - p_{ij}\right| \geq \epsilon\right) \leq \frac{\sum_{l=1}^k (w_l)^2}{4\left(\sum_{l=1}^k w_l\right)^2 \epsilon^2}, \tag{9}$$

where p_{ij} represents the actual transition probability between states s_i and s_j of the model M from (1).

Proof: Since the actual transition probability between states s_i and s_j is p_{ij}, $x_{ij}^l \in \{0, 1\}$, $1 \leq l \leq k$, are discrete random variables with (a) distribution function $P(1) = p_{ij}$ and $P(0) = 1 - p_{ij}$; (b) expected value $\mu = E(x_{ij}^l) = 1 \times p_{ij} + 0 \times (1 - p_{ij}) = p_{ij}$; and (c) variance $\sigma^2 = V(x_{ij}^l) = E\left(\left(x_{ij}^l\right)^2\right) - \left(E(x_{ij}^l)\right)^2 = \left(1^2 \times p_{ij} + 0^2 \times (1 - p_{ij})\right) - (p_{ij})^2 = p_{ij} - (p_{ij})^2$. The inequality (9) is now easy to obtain by replacing these μ and σ^2 values in (8), and noting that $\sigma^2 = p_{ij} - (p_{ij})^2 \leq \frac{1}{4}$ for all possible values of p_{ij}.

□

Dynamic selection of learning algorithm parameters.

To take advantage of the result from Proposition 2, we consider a time interval during which the mean distance between successive observations is $\overline{t} > 0$. Accordingly, $w_i^l = \alpha_i^{-(t_k - t_l)} \approx \alpha_i^{-(k-l)\overline{t}}$ and, after straightforward algebraic manipulations,

$$\frac{\sum_{l=1}^{k}(w_l)^2}{\left(\sum_{l=1}^{k} w_l\right)^2} \approx \frac{\sum_{l=1}^{k} \alpha_i^{-2(k-l)\overline{t}}}{\left(\sum_{l=1}^{k} \alpha_i^{-(k-l)\overline{t}}\right)^2} = \frac{(\alpha_i^{k\overline{t}}+1)(\alpha_i^{\overline{t}}-1)}{(\alpha_i^{k\overline{t}}-1)(\alpha_i^{\overline{t}}+1)} \approx \frac{\alpha_i^{\overline{t}}-1}{\alpha_i^{\overline{t}}+1},$$

if $\alpha^{k\overline{t}} \gg 1$. Replacing this result in (9) we obtain:

$$P\left(\left|\frac{\sum_{l=1}^{k} w_i^l x_{ij}^l}{\sum_{l=1}^{k} w_i^l} - p_{ij}\right| \geq \epsilon\right) \leq \frac{1}{4\epsilon^2}\frac{\alpha_i^{\overline{t}}-1}{\alpha_i^{\overline{t}}+1}, \quad \text{if } \alpha_i^{k\overline{t}} \gg 1. \tag{10}$$

Our adaptive transition-probability learning algorithm uses the result in (10) to adjust the smoothing parameter c_i^0 and the ageing parameter α_i from (5)–(6) dynamically, based on the mean distance between recent observations $\overline{t}$ as follows:

1. Given a small ϵ , we select α_i such that the probability from (10) is below a small value p_{max}, i.e.,

$$\frac{1}{4\epsilon^2}\frac{\alpha_i^{\overline{t}}-1}{\alpha_i^{\overline{t}}+1} \leq p_{max} \Rightarrow \alpha_i \leq \left(\frac{1+4\epsilon^2 p_{max}}{1-4\epsilon^2 p_{max}}\right)^{\frac{1}{\overline{t}}}. \tag{11}$$

2. Having selected the α_i, c_i^0 is chosen such that $\alpha_i^{c_i^0 \overline{t}} \gg 1$. Since the first term of (5) dominates the calculation of p_{ij}^k until the number of observations k is larger than c_i^0, this ensures that the k observations play a major role in the p_{ij}^k estimate only once $\alpha_i^{k\overline{t}} \gg 1$ as well. In practice, we use $\alpha_i^{c_i^0 \overline{t}} = 10$, or

$$c_i^0 = \frac{1}{\overline{t}\log_{10}\alpha}. \tag{12}$$

Complexity Analysis.

Our adaptive learning method requires the calculation of the ageing parameter α_i from (11), smoothing parameter c_i^0 from (12), weights w_i^l from (6) and probability estimates p_{ij}^k from (5) after each observation. As we showed in [5], algebraic manipulation can be used to rearrange 5)–(6) so that the last two calculations can be performed in O(1) time and using constant, O(1) space. Calculating the mean distance $\overline{t}$ between recent observations—used to compute α_i in (11)—requires the algorithm to store the timestamps of all observations within a sliding time window of fixed duration. The number of such timestamps is proportional to the frequency f of observations, so the space complexity of this calculation is O(f). The actual calculation of $\overline{t}$, however, can be carried out in O(1) time using a running sum, and computing α_i and c_i^0 also takes constant time. Accordingly, the overall space complexity of the adaptive learning algorithm is O(f), and its time complexity is O(1).

Evaluation.

To evaluate the effectiveness of the adaptive learning method, we carried out a broad range of experiments in which we compared its results with those produced by existing learning methods. The existing methods selected for this comparison were the basic Bayesian learning method from [12], and the fixed-parameter, ageing-enabled learning method from our previous work in [5]. The concrete methods compared in these experiments and their parameters are summarised in Table 1.

Table 1: Learning methods compared in the evaluation experiments

Method	Description
Method 1	basic Bayesian learning from [12], obtained by setting $w_i^l = 1$ in (5) for all $1 \leq l \leq k$, and using the smoothing parameter $c_i^0 = 500$.
Method 2	fixed-parameter, ageing-enabled learning algorithm from [5], obtained by setting $c_i^0 = 500$ and $\alpha = 1.001$ in (5)–(6).
Method 3	fixed-parameter, ageing-enabled learning algorithm from [5], obtained by setting $c_i^0 = 500$ and $\alpha = 1.01$ in (5)–(6).
Method 4	our new adaptive learning algorithm with smoothing parameter c_0 and ageing parameter α given by (11)–(12) for $p_{max} = \epsilon = 0.05$.

Figures 1–2 depict the experimental results of two typical scenarios in which we assessed the effectivenes of the adaptive learning method. The two scenarios involved learning the probability p of tossing heads with a biased coin from observations of coin tosses, when p changes over time between a "normal" value of $p = 0.96$ and a lower value. The aim of these scenarios was to simulate a degradation in the reliability with which a system component completed a given task within a predefined amount of time, and to test the ability of the four learning methods to identify this degradation. The two scenarios considered different types of reliability degradation—a longer (i.e., 8000-second) and more significant (i.e., down to $p = 0.87$) one in Scenario 1, and a shorter (1200-second) and less significant (down to $p = 0.9$) one in Scenario 2. Finally, learning each type of reliability degradation was attempted for two different observation frequencies. Thus, observation "inter-arrival" time was exponentially distributed, with a mean of 100ms (or a mean frequency of 10s^{-1}) during the first half of the experiments, and a mean of 500ms (i.e., a mean frequency of 2s^{-1}) during the second half of the experiments. A qualitative analysis of the experimental results in Figures 1–2 shows that the adaptive learning algorithm (Method 4) outperforms the existing learning algorithms (Methods 1–3) as follows:

- At the beginning of the experiment, the p^k estimate probability for the adaptive algorithm approaches p faster than for the basic algorithm in Method 1 and the two combinations of fixed-parameter ageing-enabled algorithms in Methods 2–3.

- During the "high frequency" half of the experiments, the adaptive algorithm is as good at detecting the decrease in the value of p as the "high α" algorithm in Method 3 (but with a p^k estimate that oscillates less around the actual p), and far better than the "low α" algorithm in Method 2 and the basic algorithm in Method 1;

- During the "low frequency" half of the experiments, the adaptive algorithm produces estimates that are as accurate and as smooth as the "low α" algorithm (Method 2), and much smoother than the "high α" algorithm (Method 3).

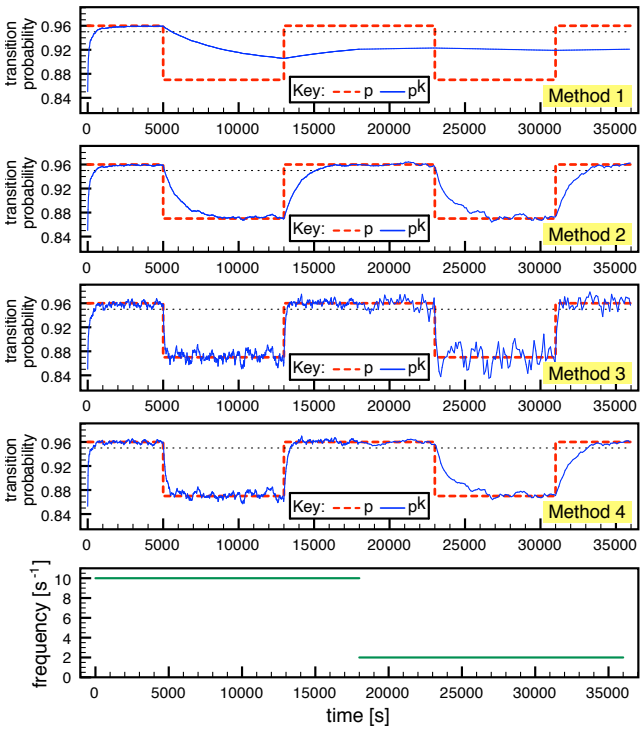

Figure 1: Experimental results—scenario 1

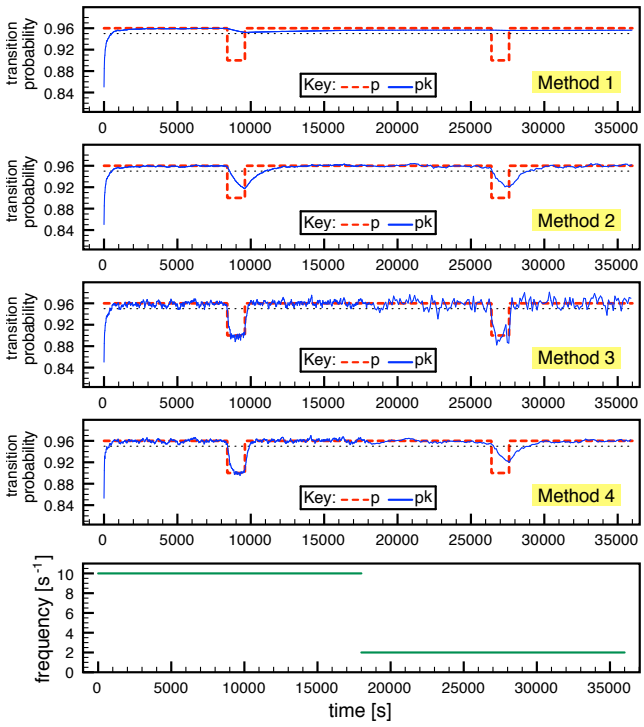

Figure 2: Experimental results—scenario 2

Although some of the estimates produced by the "high alpha" algorithm in Method 3 during the decrease in the value of p in the second half of the experiment are closer to p than the estimates produced by the adaptive algorithm, this is achieved at the expense of significant oscillation. Such oscillation is likely to trigger false alarms in a real-world scenario. If this is not a problem, then the adaptive algorithm can be configured to provide similar estimates by adjusting its confidence interval through increasing ϵ and/or p_{max}.

For a quantitative evaluation of the effectiveness of our adaptive learning method, consider a situation in which an alarm is triggered if the estimate probability p^k (representing the reliability of a system component, as explained above) drops below a threshold value $p^{required} = 0.95$. This threshold value is shown as a dotted line in all graphs in Figures 1–2. Assuming that the learning methods are used to detect such violations of a reliability threshold, we measured the following three non-functional properties of the learnt p^k values from Scenarios 1 and 2:

- The time t_{down} elapsed between the drop in the value of p and the moment when the estimate p^k becomes smaller than $p^{required}$.
- The time t_{up} elapsed between the moment when p regains its "normal" value (i.e., $p = 0.96$) after a period of degraded reliability, and the moment when the estimate p^k becomes at least $p^{required}$.
- The number of false positives n_+, i.e., instances when p^k drops below $p^{required}$ although p has its normal value.

Table 2 shows the value of these properties, separately for the periods of high-frequency and low-frequency observations from the experiments. These results indicate that Method 1 is suited for identifying only the first change in the probability p, whereas the other methods yield p^k probability estimates that follow the changes in p with more or less accuracy. The adaptive learning algorithm (Method 4) detects the changes in the value of p faster than Method 2 in the high-frequency observation area, and, like this method, produces no false positives. In the low-frequency observation area, the two methods are comparable, while Method 3 achieves slightly lower t_{down} and t_{up} but has the significant disadvantage of generating tens of false positives. As mentioned before, if these false positives are deemed acceptable, then Method 4 can achieve similar results by choosing larger p_{max} and ϵ values than those in Table 1.

Table 2: Quantitative analysis of the experiments in Scenarios 1–2

Scenario & Method (Sx_My)	high-frequency observations			low-frequency observations		
	t_{down} [s]	t_{up} [s]	n_+	t_{down} [s]	t_{up} [s]	n_+
S1_M1	570	—	0	—	—	—
S1_M2	95	2110	0	76	2149	0
S1_M3	6	217	54	33	239	38
S1_M4	26	405	0	128	2100	0
S2_M1	150	—	0	—	—	—
S2_M2	162	1350	0	46	1156	0
S2_M3	2	195	65	0	113	134
S2_M4	13	303	0	131	1120	0

The last set of experiments described in the paper was carried out for the scenario illustrated in Figure 3. In this scenario, we assume that p represents the probability that a system will perform an operation or task over another (or over remaining idle), and we suppose that p varies over a 10-hour time period (e.g., between 8am and 6pm during a working day) as shown by the thick dashed line. Our experiments assessed to what extend the estimate probability p^k provided by each of the four learning remained with the interval $[p - \epsilon, p + \epsilon]$ while the observation frequency was

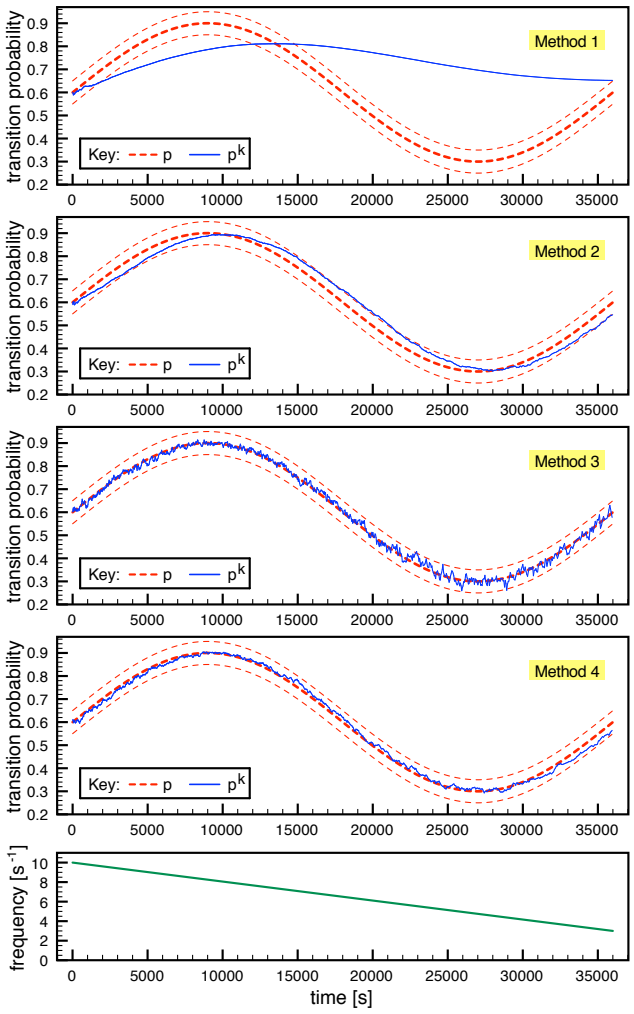

Figure 3: Experimental results—scenario 3

decreased linearly from 10s^{-1} to 2s^{-1}. The value $\epsilon = 0.05$ was chosen, in order to match the value of ϵ used by the adaptive learning algorithm (cf. Table 1). The typical experimental results in Figure 3 show that Method 1 cannot handle this degreee of variability, while Method 2 yields p^k estimates within the desired interval around p most of the time. In contrast, Methods 3–4 produce estimates that remain within this interval throughout the 10-hour simulated time period. The main difference between these two methods is that Method 4 (the adaptive learning algorithm) achieves this objective with much less oscillation around the actual value p.

In order to measure the accuracy of the estimates quantitatively, we performed 100 experiments simular to those from Figure 3, for each of the four learning methods. For each experiment, we measured the cumulated time $t_{outside}$ during which the estimate probability p^k resided outside the interval $[p - \epsilon, p + \epsilon]$. Table 3 reports these times, averaged over the 100 experiments carried out for each learning method, confirming that the adaptive learning methods outperforms the other three methods according to this criterion.

4. IMPLEMENTATION

To ease the adoption of the theoretical results from Section 3 in quality-of-service engineering practice, we implemented a software engineering tool and reusable middleware that allow practitioners to exploit our adaptive learning method in the development of self-adaptive service-based systems (SBSs) with the architecture from Fig. 4. This architecture comprises $n \geqslant 1$ operations performed by remote third-party services, and our new software engineering tool generates automatically the n *intelligent proxies* used to interface the SBS workflow with sets of remote service such that the i-th SBS operation can be carried out by $m_i \geqslant 1$ functionally equivalent services.

Table 3: Cumulated times when the estimate probability p^k is outside the interval $[p-\epsilon, p+\epsilon]$, averaged over 100 36,000-second experiments

METHOD	$t_{outside}$ [s]
Method 1	31390.64
Method 2	4791.11
Method 3	60.91
Method 4	15.17

The role of the intelligent proxies is to ensure that each execution of an SBS operation is carried out through the invocation of a concrete service selected such that the non-functional requirements of the system are satisfied. Whenever an instance of the i-th proxy is created, it is initialised with a sequence of "promised" service-level agreements $sla_{ij} = (p^0_{ij}, c_{ij}), 1 \leqslant j \leqslant m_i$, where $p^0_{ij} \in [0, 1]$ and $c_{i,j} > 0$ represent the provider-supplied probability of success and the cost for an invocation of service s^j_i, respectively. The n proxies are also responsible for announcing each service invocation and its outcome to a *model updater*, which we implemented as reusable middleware, and we integrated with our COntinual VErification (COVE) framework from [6]. The model updater uses the adaptive learning algorithm from Section 3 to adjust the transition probabilities of an initial DTMC model of the SBS workflow in line with these proxy notifications.

Finally, the up-to-date DTMC model maintained by the model updater is used by an existing COVE *autonomic manager*, which performs continual non-functional property verification to select the service combination used by the n proxies so that it satisfies the SBS requirements with minimal cost at all times. Accordingly, the proxies, model updater and autonomic manager with its quantitative verification engine implement a monitor-analyse-plan-execute (MAPE) autonomic computing loop [23].

The new software engineering tool is implemented as a Java web application, generates each intelligent proxy as a Java ARchive (JAR) component, and is freely available:

- Pre-installed as a web application on the public Amazon Machine Image with AMI ID `ami-db7020b2` and AMI Name `WB-IPGenTool-2013` from the `us-east-1` Amazon EC2 region (Figure 5). Starting an Amazon EC2 (`http://aws.amazon.com/ec2/`) virtual machine that uses this AMI has the significant advantage that multiple developers can then instantly access and use the tool from a web browser running on their local machines, with no installation or configuration effort.
- As an open-source application for deployment on a local development machine, at `http://www-users.cs.york.ac.uk/~raduc/COVE`.

The model updater and the components of the COVE framework it was integrated with are implemented as an open-source Java library, which is also freely available from `http://www-users.cs.york.ac.uk/~raduc/COVE`.

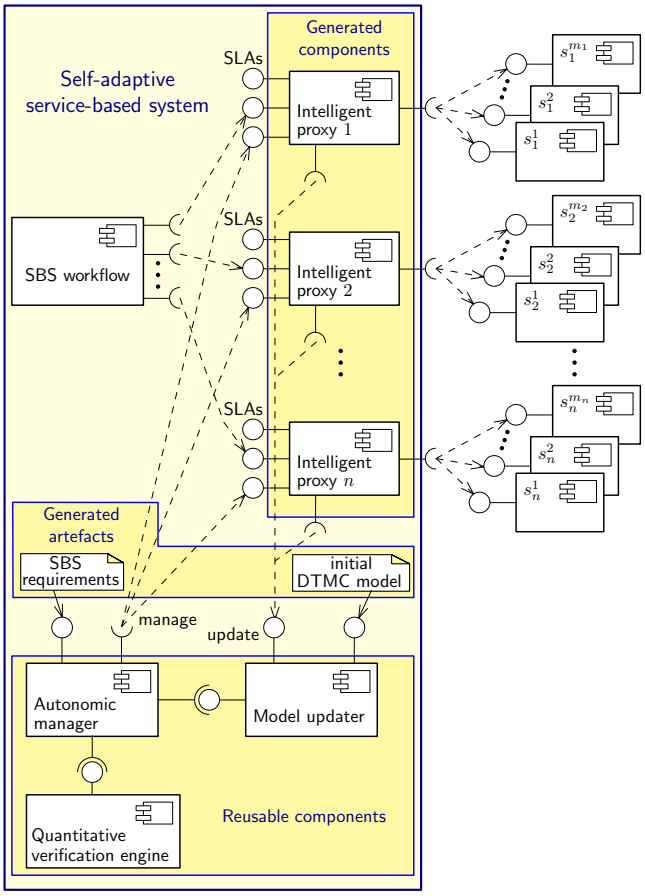

Figure 4: Self-adaptive service-based system that uses continual non-functional property verification, originally proposed in [4] and extended in [3].

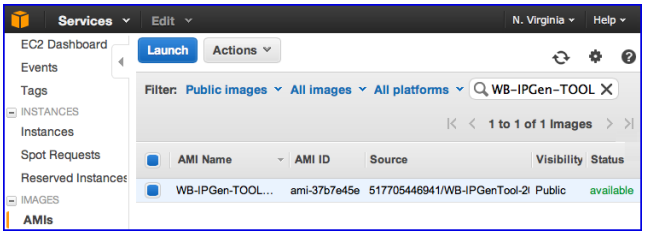

Figure 5: Public Amazon Machine Image pre-installed with the intelligent proxy generator tool

SBS Development Process.

The development of a self-adaptive SBS using the new proxy generator and middleware comprises three stages:

1. The developer selects $m_i \geq 1$ functionally equivalent services that implement the i-th SBS operation, $1 \leq i \leq n$, and uses the new proxy generator to synthesise the i-th intelligent proxy as a Java package, starting from the m_i web service WSDL definitions. The m_i services may be associated with different levels of reliability and different costs. In addition, our proxy generator can accommodate differences in the parameter and return types of the m_i web methods that implement the SBS operation, by allowing the developer to specify conversions between these parameters and return types and those of the SBS operation. This is a key advantage of our proxy generator, since in practice it is difficult to find equivalent services whose methods also have identical signatures (Figure 6).

2. The developer uses existing COVE tools [6] to generate the initial DTMC model used to set up the model updater and to formalise the SBS requirements in PCTL, starting from an annotated SBS activity diagram in the XMI format generated by the Eclipse-based Papyrus graphical editing tool for UML 2 (`http://www.eclipse.org/papyrus/`). The process is presented in detail in [6].

3. The developer integrates the n intelligent proxies with the code that implements the SBS workflow, in a similar manner to using standard web service proxies. Additionally, an instance of the model updater and an instance of the COVE autonomic manager from [6] are created and initialised with the initial DTMC model and the array of PCTL requirements from the previous stage, respectively.

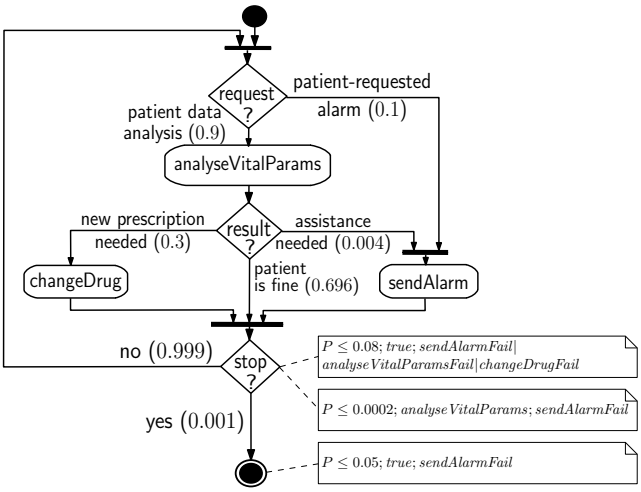

Figure 7: UML activity diagram of a telehealth SBS. Estimate *a priori* probabilities are associated with the edges that originate in a decision node, and comments specifying the SBS requirements are associated with relevant nodes.

5. CASE STUDY

We used the adaptive model learning method from Section 3 and the tools and development process described in Section 4 to implement a self-adaptive version of a telehealth service-based system taken from [3, 4, 12]. In this SBS, the vital parameters of a patient are periodically measured by a wearable device and analysed by third-party medical services. The result of the analysis may trigger the invocation of an alarm service (that determines, for instance, the dispatch of an ambulance), may lead to the invocation of a pharmacy service to deliver new medication to the patient, or may confirm that the patient is fine. In addition, the patient can initiate an alarm by using a panic button on the wearable device. The workflow of the telehealth SBS is

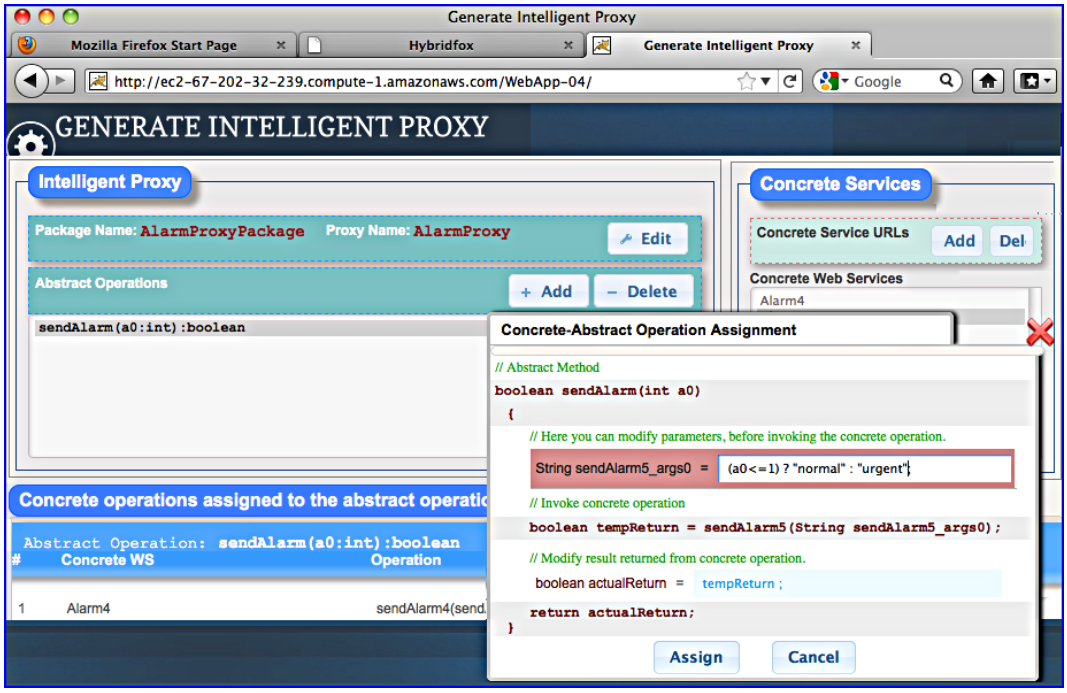

Figure 6: Instance of the intelligent proxy generator, running as a web application on an Amazon virtual machine, and used from a web browser. Parameter/return type conversions are supported between "abstract" SBS operations and "concrete" services.

Table 4: Service prior success probabilities and costs

service	prior success probability ($p^0_{i,j}$)	cost ($c_{i,j}$)
sendAlarm_1	0.968	0.02
sendAlarm_2	0.968	0.01
changeDrug_1	0.96	0.3
changeDrug_2	0.95	0.1
$\text{analyseVitalParams}_1$	0.965	5.0
$\text{analyseVitalParams}_2$	0.95	4.0
$\text{analyseVitalParams}_3$	0.96	3.0

shown in Fig. 7, and the three non-functional requirements used in the case study are:

R_1: The probability that one execution of the workflow ends in a service failure is at most $p_{R_1} = 0.08$.

R_2: The probability that an alarm failure occurs within $N = 10$ executions of the workflow is at most $p_{R_2} = 0.05$.

R_3: The probability that an invocation of the analysis service is followed by an alarm failure is at most $p_{R_3} = 0.0002$.

The self-adaptive version of the telehealth SBS used $m_1 = 2$ sendAlarm services, $m_2 = 2$ changeDrug services, and $m_3 = 3$ analyseVitalParams services. These seven services were simulated using real Java web services deployed on Amazon EC2 "small instance" virtual machines. Individual configuration files were used to specify the variation of the actual probability of successful invocation for each web service, $p_{i,j}$, $1 \leq i \leq 3$, $1 \leq j \leq m_i$, over the duration of each experiment. The *a priori* success probabilities $p^0_{i,j}$ and the costs $c_{i,j}$ for an invocation of each of these services are shown in Table 4. A Java implementation of the telehealth SBS workflow from Fig. 7 was integrated with intelligent proxies for its three operations, and was run on a standard 2.66 GHz Intel Core 2 Duo Macbook Pro computer.

Fig. 8 shows a typical experiment in which the self-adaptive SBS selects the service combinations for its telehealth workflow dynamically, over a 1.5-hour wall-clock time period. Low-cost combinations of services are preferred when their combined probabilities of successful completion satisfy all SBS relibility requirements, and are discarded in favour of higher-cost service combinations when their joint reliability violates one or more of these SBS requirements. These decisions are taken based on the estimate probabilities of success p^k_{ij} calculated by our adaptive learning algorithm (initialised with $\epsilon = p_{max} = 0.05$), and on the continual verification of the updated SBS model:

- At the beginning of the experiment, the lowest-cost service combination is selected, as the high *a priori* success probabilities $p^0_{i,j}$ of all services make all service combinations seem suitable. This is the expected behaviour, since a service whose provider-specified SLA does not satisfy the SBS requirements should not be included in the system.

- When the SBS learns that $\text{analysisVitalParams}_3$ is underperforming in the area labelled 'A' in the diagram, it starts using the higher-cost $\text{analysisVitalParams}_2$ service.

- While a higher-cost service is used for an SBS operation, the adaptive learning algorithm "rebuilds trust" in the temporarily discarded lower cost service (area labelled 'B' in the diagram). This is due to the fact that the observations of frequent failures from area 'A' are associated with weights that decrease over time, so the estimate $p^k_{3,3}$ slowly approaches the prior value

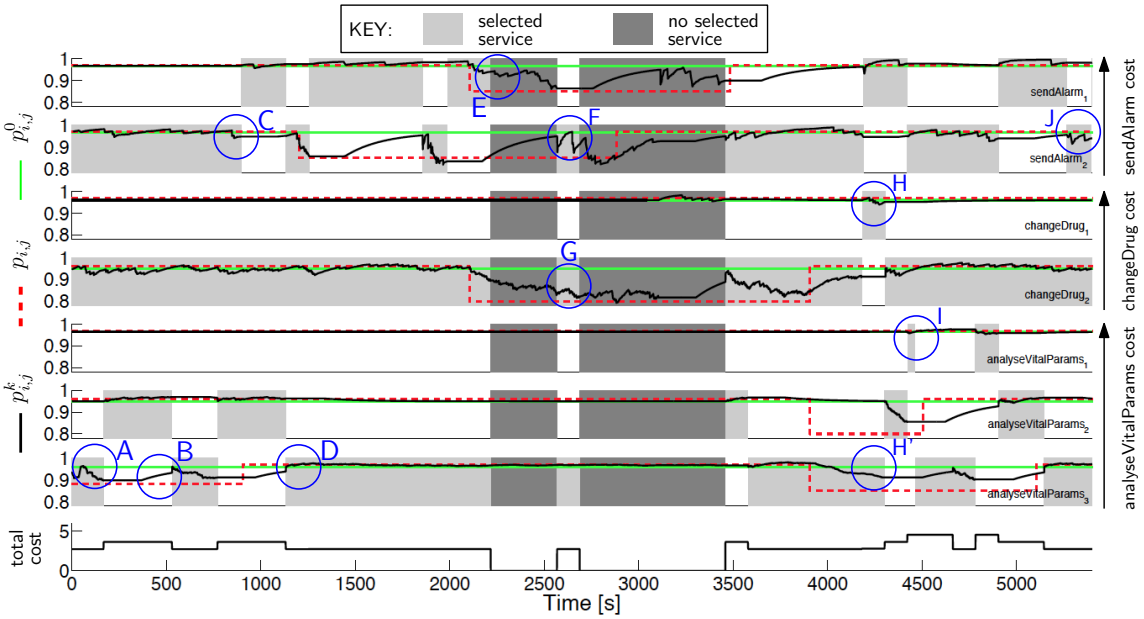

Figure 8: Automated service selection for the telehealth service-based system; the circular areas labelled 'A', 'B', etc. are analysed in Section 5

$p^0_{3,3}$. The learning algorithm was configured to assume that a service returned to its prior success probability when the autonomic manager resumes using it, which explains why $p^k_{3,3}$ grows suddenly to $p^0_{3,3}$ when the analysisVitalParams$_3$ is selected again in area B.

- In area C, a slight variation in the estimate success probability of the sendAlarm$_2$ service triggers a potentially unnecessary transition to the more expensive service sendAlarm$_1$. Choosing strict intervals of confidence (i.e., smaller ϵ and/or p_{max} parameters) for the adaptive learning could reduce such "false positives", although eliminating them altogether is not possible (cf. Proposition 2).

- In area D, the SBS resumes using analysisVitalParams$_3$, which has now recovered.

- In area E, the system learns that even the high-cost alarm service sendAlarm$_2$ is unreliable, to the extent that the SBS requirements are no longer satisfied. Under the configuration used in the experiment, no service was selected in this scenario, and an error message was generated instead to alert the system operator.

- In area F, the system retries to use the alarm service that experienced a low success rate first, and learns that this services has not yet recovered.

- Area G shows that some services have little impact on the overall SBS compliance with its requirements: given that only requirement R_1 depends on a successful completion of the changeDrug SBS operation (and only marginally), a decrease in the reliability of changeDrug$_2$ does not determine the SBS to abandon this service.

- Nevertheless, the SBS does switch to the more expensive changeDrug$_1$ service in area H–H', at a moment when changeDrug$_2$ is actually more reliable than it was in area G. The decision is motivated by the decrease in the reliability of dataAnalysis$_3$, which the system compensates for by choosing a slightly more expensive drug service (the cost difference between changeDrug$_2$ and changeDrug$_1$ is only 0.2) instead of switching to a significantly more expensive analysis service (ceasing to use analyseVitalParams$_3$ would have amounted to a cost increase of at least 1.0 for this operation).

- The strategy adopted in area H–H' is unsuccessful, so the most expensive analysis service is eventually selected in area I.

- Finally, in area J all services have recovered and operated close to their advertised SLAs, so the self-adaptive SBS returns to using the lowest-cost service combination for the telehealth service-based system.

A key capability of our adaptive learning method is its ability to learn not only changes in the reliability of individual services, but also changes in the rates with which the SBS operations are performed. To evaluate this functionality, we considered the effect of changes in the probability $p_{request\_sendAlarm}$ that a request handled by the telehealth SBS is a patient-initiated alarm. A temporary increase in this probability may be caused, for instance, by a flu outbreak. Fig. 9 depicts the analysis of requirement R_1 from our case study, for a range of service combinations and for $p_{request\_sendAlarm}$ values between 0.05 and 0.15. This analysis shows that even a small change in the probability of alarm requests is sufficient to render unacceptable a service combination that was previously compliant with requirement R_1. This confirms the importance of updating the SBS model in line with any fluctuations in the probabilities with which the SBS operations are executed.

Scalability.

To evaluate the scalability and generality of our approach, we carried out a number of experiments that assessed the ap-

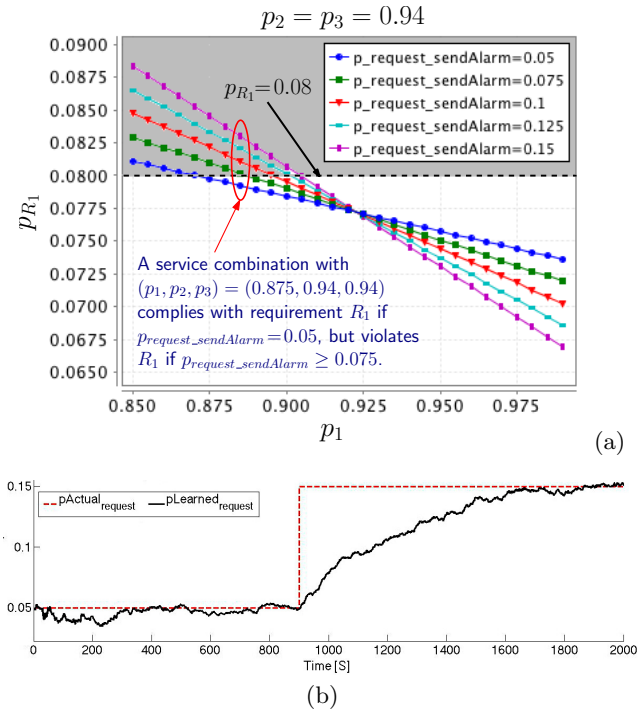

Figure 9: (a) The effect of changes in the probability of alarm requests; and (b) learning this probability

plicability and overheads of executing the runtime adaptive learning and model analysis in multiple scenarios. We selected the following workflows used by a number of projects in this area:

1. the healthcare case study described in this paper, and previously used in [4, 3, 12];

2. the e-commerce workflow obtained from [15];

3. the travel assistant workflow derived from the state-chart representation presented in [32].

These workflows comprise invocations to three, four and five abstract operations, respectively.

For each of the workflows we devised a parameterised DTMC and defined four PCTL requirements, including one PCTL property to determine the expected cost of a single invocation of the workflow. The size of the models ranged between 11 and 17 states. As we envisage that practical self-adaptive service-based systems will rarely use more than two or three concrete services for each abstract operation, we then ran experiments that considered between two and six concrete services for each of the abstract operations. Due to space constraints, we could not include the DTMC models and properties for the e-commerce and travel-assistant workflows in this paper. However, these DTMC models and properties, and detailed descriptions of each of these experiments are available at `http://www-users.cs.york.ac.uk/~raduc/COVE`.

Each experiment measured the time taken to initialise the system and select the optimal concrete service configuration in the worst-case scenario whereby all combinations of concrete services satisfied the SBS requirements. Note that this is the worst-case scenarios because the autonomic

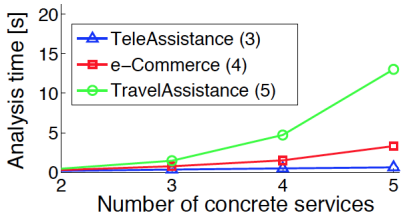

Figure 10: Scalability results for 2–5 "concrete" services per SBS operation

manager stops verifying the suitability of a service combination as soon as it learns that the combination violates one of the requirements. Fig. 10 summarises the results of our experiments, averaged over multiple runs. According to these results, up to three services per SBS operation can be analysed within two seconds for each of the considered workflows, which confirms the feasibility of the approach for typical SBSs of practical importance from the domains explored in our experiments. Increase the number of services to four services leads to verification times of up to $5s$, which is likely to be acceptable for many practical applications. This is particularly true when large numbers of false positives and false negatives in the associated learning process need to be avoided, so longer time is already needed to identify the changes on which the autonomic manager must act.

The growth in analysis time shown in Fig. 10 makes the current implementation of the approach suitable for systems comprising small to medium numbers of operations, and using between two and four services per SBS operation. While the second constraint is, in our opinion, not significant, the former implies that SBSs comprising large numbers of operations cannot yet benefit from this approach. However, recent work by several research groups and ourselves has led to significant advances in the use of incremental and compositional techniques to reduce quantitative verification times, often by multiple orders of magnitude [7, 14, 21, 25]. We envisage that integrating these techniques into the approach will significantly enhance its ability to support the development and operation of much larger service-based systems.

6. RELATED WORK

Significant research has focused on monitoring the performance and reliability properties of technical systems, and on modelling and analysing these properties formally. The spread of evolving critical systems [2] led to a growing need for combining techniques from the two research areas in a runtime context, in order to achieve a continual verification of the non-functional properties of these systems [3].

The projects addressing the challenges of continual verification have so far focused primarily on reducing the overheads of runtime analysis of formal models [7, 13, 14, 16, 21], with relatively little effort dedicated to ensuring that the analysed formal models are updated in line with the changes in the analysed system. The work presented in [12] proposes the on-line learning algorithm referred to as "Method 1" in the evaluation part from Section 3, where we show that our adaptive learning is better suited for all scenarios in which the learnt DTMC transition probabilities undergo multiple changes over time. The approach introduced in [34, 35] uses Kalman filter estimators to update the parameters of queueing-network performance models. Our results complement this approach, as they target DTMC reliability

models. Finally, our new adaptive learning approach is a significant improvement over the on-line learning approach from our previous work in [5]. This was shown in the evaluation part from Section 3, where representative instances of the learning approach from [5] (labelled "Method 2" and "Method 3") were compared to the new adaptive learning.

The management and optimisation of SBS non-functional properties through dynamic service selection has been the focus of significant research over the past decade. The solutions proposed by this research include approaches that use intelligent control loops (e.g., [1, 9, 28]) and approaches that emulate the cooperative behaviour of biological systems (e.g., [17, 29]). The approach supported by our new intelligent proxy generator and middleware belongs to the first category, so the rest of this section focuses on comparing our work with results from this area, and in particular with solutions that employ formal models that can represent SBSs accurately and in a realistic way. The approaches proposed in [18, 28, 27, 30] use UML activity diagrams or directed acyclic graphs to synthesise simple performance models based on queuing networks [28, 27] or, like our approach, Markovian reliability models [18, 30]. These models are then used to establish the quality-of-service (QoS) properties of the analysed SBS systems. However, unlike these approaches, our solution also uses an adaptive learning technique to update the initial model based on observations of the system behaviour. The QoS-driven selection of services in self-adaptive service-based systems is addressed in [1, 9, 10, 33]. All these approaches lack adaptive learning capabilities, and propose theoretical solutions that are hard to replicate in practical SBSs. In addition, approaches such as [1, 9, 26, 33] involve the optimisation of the service selection on a per request basis. These approaches require perfect knowledge of the QoS capabilities of the available services, which renders them ineffective in the scenarios targeted by our work, where the characteristics of services need to be learnt from observations of their behaviour.

The work presented in this paper also differs from our previous results in [4], as it introduces an adaptive learning method that is underpinned by new theoretical results and used to estimate not only changes in the reliability of individual services, but also variations in the probabilities with which the operations of an evolving system are invoked. Furthermore, we describe a new proxy generation tool and model updater that are missing from our previous work.

7. CONCLUSION

We introduced a new on-line learning method for maintaining discrete-time Markov reliability models of evolving critical systems in sync with the systems they represent. Unlike existing approaches to updating such models, our new method adapts its parameters dynamically, to suit the frequency of the observations it relies upon and the developer-specified confidence intervals for its estimates. This adaptation is based on a rigorous theoretical foundation, also introduced in the paper, and we showed that our model learning method outperforms existing approaches in a range of scenarios of practical relevance.

Our adaptive model learning method is a key component for the continual verification of the non-functional properties of evolving systems. To make the new method available to practitioners interested in continual verification, we implemented a software engineering tool and middleware that enable its adoption by developers of self-adaptive service-based systems. This development tool is pre-installed on a publicly available Amazon EC2 machine image, so developers can use the tool without having to first install and configure it and the third-party libraries it uses. The effectiveness of the tool and its integration with our existing continual verification framework from [6] was demonstrated in a case study from the telehealth application domain, and the scalability of the approach was evaluated for three service-based systems used by projects in this area. The results of this evaluation indicate that the approach is applicable to SBS workflows of practical significance. Extending the applicability of the approach to large service-based systems requires its integration with recently emerged incremental and compositional verification techniques [7, 14, 21, 25]. Achieving this integration represents an area of ongoing work for our project. The main target of this work is the incremental verification technique we proposed in [21], which we deem particularly suitable for our purpose due to its ability to produce system-level verification results by re-analysing only the parts of the system that were affected by a change.

Another area of ongoing work for our project is the extension of the adaptive model learning method with the ability to handle models supporting the analysis of different categories of non-functional requirements (e.g., performance and energy related), along the lines of our previous work in [4].

Acknowledgment

This work was partly supported by the UK Engineering and Physical Sciences Research Council grant EP/H042644/1.

8. REFERENCES

[1] D. Ardagna and B. Pernici. Adaptive service composition in flexible processes. *IEEE Trans. Softw. Eng.*, 33(6):369–384, 2007.

[2] R. Calinescu. Emerging techniques for the engineering of self-adaptive high-integrity software. In J. Camara et al., editors, *Assurances for Self-Adaptive Systems*, volume 7740 of *LNCS*, pages 297–310. Springer, 2013.

[3] R. Calinescu, C. Ghezzi, M. Kwiatkowska, and R. Mirandola. Self-adaptive software needs quantitative verification at runtime. *Communications of the ACM*, 55(9):69–77, September 2012.

[4] R. Calinescu, L. Grunske, M. Kwiatkowska, R. Mirandola, and G. Tamburrelli. Dynamic QoS management and optimization in service-based systems. *IEEE Trans. Softw. Eng.*, 37:387–409, 2011.

[5] R. Calinescu, K. Johnson, and Y. Rafiq. Using observation ageing to improve Markovian model learning in QoS engineering. In *2nd ACM/SPEC Intl. Conf. on Performance Engineering*, pages 505–510, 2011.

[6] R. Calinescu, K. Johnson, and Y. Rafiq. Developing self-verifying service-based systems. In *28th Intl. IEEE/ACM Conference on Automated software Engineering*, 2013. To appear.

[7] R. Calinescu, S. Kikuchi, and K. Johnson. Compositional reverification of probabilistic safety properties for large-scale complex IT systems. In *Large-Scale Complex IT Systems*, volume 7539 of *LNCS*, pages 303–329. Springer, 2012.

[8] R. Calinescu and M. Z. Kwiatkowska. Using quantitative analysis to implement autonomic IT systems. In *Proceedings of the 31st International Conference on Software Engineering, ICSE 2009*, pages 100–110. IEEE Computer Society, 2009.

[9] G. Canfora, M. D. Penta, R. Esposito, and M. L. Villani. A framework for QoS-aware binding and re-binding of composite web services. *Journal of Systems and Software*, 81(10):1754–1769, 2008.

[10] V. Cardellini, E. Casalicchio, V. Grassi, and F. L. Presti. Scalable service selection for web service composition supporting differentiated qos classes. Technical Report Technical Report RR-07.59, Dip. di Informatica, Sistemi e Produzione, Universita di Roma Tor Vergata, 2007.

[11] F. Ciesinski and M. Größer. On probabilistic computation tree logic. In C. Baier et al., editors, *Validation of Stochastic Systems - A Guide to Current Research*, volume 2925 of *LNCS*, pages 147–188. Springer, 2004.

[12] I. Epifani, C. Ghezzi, R. Mirandola, and G. Tamburrelli. Model evolution by run-time adaptation. In *Proc. 31st Intl. Conf. Software Engineering (ICSE'09)*, pages 111–121, 2009.

[13] A. Filieri and C. Ghezzi. Further steps towards efficient runtime verification: Handling probabilistic cost models. In *Software Engineering: Rigorous and Agile Approaches (FormSERA), 2012 Formal Methods in*, pages 2–8, 2012.

[14] A. Filieri, C. Ghezzi, and G. Tamburrelli. Run-time efficient probabilistic model checking. In *Proc. 33rd International Conference on Software Engineering*, pages 341–350. IEEE Computer Society, 2011.

[15] A. Filieri, C. Ghezzi, and G. Tamburrelli. A formal approach to adaptive software: continuous assurance of non-functional requirements. *Formal Aspects of Computing*, 24(2):163–186, 2012.

[16] V. Forejt, M. Kwiatkowska, D. Parker, H. Qu, and M. Ujma. Incremental runtime verification of probabilistic systems. In S. Qadeer and S. Tasiran, editors, *Runtime Verification*, volume 7687 of *Lecture Notes in Computer Science*, pages 314–319. Springer Berlin Heidelberg, 2013.

[17] R. Frei, G. D. M. Serugendo, and J. Barata. Designing self-organization for evolvable assembly systems. In *Second IEEE Intern. Conf. on Self-Adaptive and Self-Organizing Systems, SASO 2008*, pages 97–106, 2008.

[18] S. Gallotti, C. Ghezzi, R. Mirandola, and G. Tamburrelli. Quality prediction of service compositions through probabilistic model checking. In S. Becker, F. Plasil, and R. Reussner, editors, *Proc. 4th International Conference on the Quality of Software-Architectures, QoSA 2008*, volume 5281 of *LNCS*, pages 119–134. Springer, 2008.

[19] C. M. Grinstead and J. L. Snell. *Introduction to Probability*. American Mathematical Society, 1997.

[20] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.

[21] K. Johnson, R. Calinescu, and S. Kikuchi. An incremental verification framework for component-based software systems. In *Proc. 16th Intl. ACM Sigsoft Symposium on Component-Based Software Engineering*, pages 33–42, 2013.

[22] J.-P. Katoen, M. Khattri, and I. S. Zapreev. A Markov reward model checker. In *Quantitative Evaluation of Systems*, pages 243–244, Los Alamitos, 2005. IEEE Computer Society.

[23] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *IEEE Computer Journal*, 36(1):41–50, January 2003.

[24] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV'11*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[25] M. Kwiatkowska, G. Norman, D. Parker, and H. Qu. Assume-guarantee verification for probabilistic systems. In *TACAS'10*, pages 23–37. Springer, 2010.

[26] Q. Liang, X. Wu, and H. C. Lau. Optimizing service systems based on application-level QoS. *IEEE Trans. Service Computing*, 2:108–121, 2009.

[27] M. Marzolla and R. Mirandola. Performance prediction of web service workflows. In *International Conference on Quality of Software Architectures, QoSA 2007*, volume 4880 of *LNCS*, pages 127–144. Springer, 2007.

[28] D. Menascé, H. Ruan, and H. Gomaa. QoS management in service-oriented architectures. *Perform. Eval.*, 64(7):646–663, 2007.

[29] F. Saffre, R. Tateson, J. Halloy, M. Shackleton, and J.-L. Deneubourg. Aggregation dynamics in overlay networks and their implications for self-organized distributed applications. *The Computer Journal*, 2008.

[30] N. Sato and K. S. Trivedi. Stochastic modeling of composite web services for closed-form analysis of their performance and reliability bottlenecks. In *ICSOC*, volume 4749 of *LNCS*, pages 107–118. Springer, 2007.

[31] H. L. S. Younes. Ymer: A statistical model checker. In K. Etessami et al., editors, *Computer Aided Verification*, volume 3576 of *LNCS*, pages 429–433. Springer, 2005.

[32] L. Zeng, B. Benatallah, A. H.H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. Qos-aware middleware for web services composition. *IEEE Trans. Softw. Eng.*, 30(5):311–327, May 2004.

[33] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. QoS-aware middleware for web services composition. *IEEE Trans. Software Eng*, 30(5):311–327, 2004.

[34] T. Zheng, M. Woodside, and M. Litoiu. Performance model estimation and tracking using optimal filters. *IEEE Transactions on Software Engineering*, 34(3):391–406, 2008.

[35] T. Zheng, J. Yang, M. Woodside, M. Litoiu, and G. Iszlai. Tracking time-varying parameters in software systems with extended Kalman filters. In *Proceedings of the 2005 conference of the Centre for Advanced Studies on Collaborative research*, CASCON '05, pages 334–345. IBM Press, 2005.