Stochastic model checking for predicting component failures and service availability

Muffy Calder and Michele Sevegnani

Abstract—When a component fails in a critical communications service, how urgent is a repair? If we repair within 1 hour, 2 hours, or n hours, how does this affect the likelihood of service failure? Can a formal model support assessing the impact, prioritisation, and scheduling of repairs in the event of component failures, and forecasting of maintenance costs? These are some of the questions posed to us by a large organisation and here we report on our experience of developing a stochastic framework based on a discrete space model and temporal logic to answer them. We define and explore both standard steady-state and transient temporal logic properties concerning the likelihood of service failure within certain time bounds, forecasting maintenance costs, and we introduce a new concept of *envelopes of behaviour* that quantify the effect of the status of lower level components on service availability. The resulting model is highly parameterised and user interaction for experimentation is supported by a lightweight, web-based interface.

Index Terms—Reliability, decision support systems, predictive models, stochastic systems, discrete-event systems.

1 Introduction

We report on our experience of developing a stochastic model and temporal logic analysis to support the management of a critical communications service deployed within a large organisation. The service is key to their main safety-critical system and so at all times, it must operate with acceptable risk of failure.

Components of the service are organised hierarchically, with several levels of redundancy. The service operates continuously and individual component failures are monitored and logged; the service is almost always in a *degraded* configuration, *i.e.* one in which there are failed components, but because of redundancy, the service is still operating. The time/cost for a component repair depends on a number of factors, including the nature of the failure and physical distance or access to the component (many components are physically remote). The key management questions posed to us were how can a formal model support:

- assessing the impact, prioritisation, and scheduling of repairs in the event of component failures,
- forecasting of maintenance costs.

The modelling and analysis challenge for us was to develop an effective, tractable framework that addresses the questions *the organisation cares about*, in the context of a degraded service – so standard approaches that reason from a fixed initial state are not applicable. There had been no previous attempts to model the service, and no system requirements or specification documents were available to us, however, we were given access to all operational documentation, historical failure data, and freedom to interview the operating engineers.

In this paper we focus specifically on questions and decisions of the form:

- *M. Calder and M. Sevegnani are with the School of Computing Science, University of Glasgow, Glasgow, G12 8RZ, UK. E-mail: {muffy.calder,michele.sevegnani}@glasgow.ac.uk*

xxx.

From a given degraded configuration, do we need to fix a particular fault right now, in the next n hours, or can we wait until tomorrow?

For example, if we can determine the likelihood of service failure over the next n hours remains well below an established safety threshold, but thereafter rises well above the threshold, then we can decide that a repair need not be immediate, but must be completed within the next n hours. However, the decision may also depend on other parameters (*e.g.* resources and costs) to minimise or maximise.

The aim of this paper is to describe how we used formal modelling and analysis based on continuous-time Markov chains (CTMCs) and the temporal logic *Continuous Stochastic Logic* (CSL) [1]. Our focus is *not* textbook performance analysis such as occupancy or first passage time, but the questions *posed* to us by the organisation, as mentioned above. While the questions were posed for a particular deployed communications system, as the research progressed, we could also see an appetite within the organisation to evaluate design aspects such as adding degrees of autonomy to the system, changing levels of redundancy and monitoring practices, so we also factored these into requirements.

Our framework is based on treating each component as a discrete-state process, with events representing failures, repairs *etc.* The overall system is the concurrent composition of all components, synchronising on common events. The passage of time is modelled continuously and so the underlying models are continuous-time Markov chains, *i.e.* the state space is discrete but time is continuous. While modelling component-based systems with CTMCs is standard, we model at a higher level, using the PRISM language (for reactive systems), which allows for elegant treatment of components as modules, events as guarded commands, and levels of the hierarchy as module compositions. The properties we consider are probabilistic, such as likelihood of service failure and a new concept called *envelopes of behaviour* that quantify the effect of different combinations of status of lower level components on service availability. We also

consider properties that quantify recoverability and survivability after a loss of service and properties that employ rewards to forecast maintenance costs, both cumulatively and in steady-state.

Property analysis is by model checking, which provides exhaustive analysis and probabilistic guarantees of temporal logic properties over all possible executions. The properties are easily expressed in CSL and the PRISM model checker provides mechanisms that reason from states other than the initial one. This approach is complementary to simulation, which explores single executions. Further, by varying the underlying time interval, we plot trajectories of temporal property likelihood. This allows us to answer the questions posed for this system, which are not based on the standard analyses for CTMCs.

We do not expect engineers to be familiar with the details of the model nor the PRISM language, so we developed a web-based, lightweight interface that supports simple instantiation, using sliders, of model parameters, and with meaningful pre-determined default values.

1.1 Summary of modelling and analysis framework

The overall framework is depicted in Fig. 1 and summarised as follows. Model definition and analysis is indicated by solid lines, feedback from the analysis is indicated by dashed lines. The example property outputs (bottom of Fig. 1) are screenshots (that are displayed within the web app). The model is validated by comparing the results of steady-state temporal logic properties with the expected (or required) results from the safety and business cases and observed results inferred from the field data (left-hand side of Fig. 1). The model is used for quantified prediction, *e.g.* of service failure at future times and cumulative costs, by examination of transient temporal logic properties that are displayed through the web app (right-hand side of Fig. 1). Model parameters (*e.g.* rates and component configurations) are modified through the web app. The framework can be used:

1) in real-time, on-line, to inform operational decision making,
2) after the system has been deployed, and the model is parameterised by operational data, to investigate whether or not a particular architecture actually meets service requirements,
3) at design time, to investigate whether or not a particular architecture meets service requirements,
4) as a combination of the first and second in which a "catalogue" of predictions (generated off-line) for a variety of degraded configurations is provided and then consulted as the system evolves in real-time.

Modelling for reliability analysis with CTMCs is well known, but this was not an obvious solution to the problem as originally presented; we had to work with the engineers to uncover the stochastic nature of failures, what was required from a model, and from a modelling approach. We identified that compositional modelling was intuitive for the service, representing each component in the hierarchy by a discrete-state process and the overall service as their concurrent composition. The advantages of analysis by logic

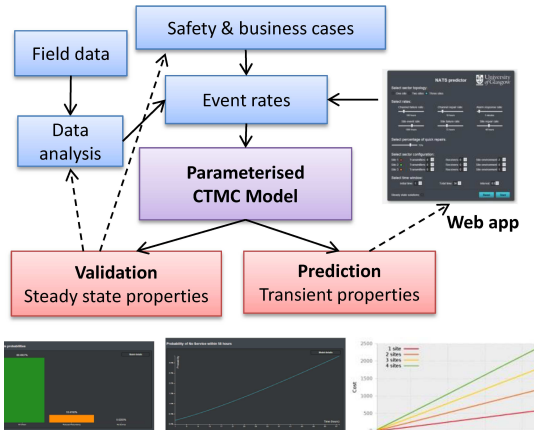


Fig. 1. Modelling and analysis framework.

property and probabilistic model checking, compared with simulation, are establishing causal relations and quantitative properties (expressed explicitly in the logic) and exhaustive exploration of the entire state space. Simulation is focussed on individual trajectories, analysis is therefore approximate and generally must include large numbers of runs.

The paper is organised as follows. The next section contains an overview of the communications service and in the following section we review basic definitions of CTMCs and CSL. In Section 4 we give an overview of the model and discuss event rates and their inference from observed behaviour over an example (historical) time period. Section 5 defines the propositions, steady-state and transient CSL properties we use for analysis, along with example results. Section 6 shows how, by way of examples, analysis of transient properties can inform decision making concerning which component to repair, and to what timescale. In Section 7 we define envelopes of behaviour and give example results. In Section 8, we consider recoverability and survivability, two properties that quantify behaviours *after* reaching no-service and in Section 9 we present an analysis of maintenance costs using PRISM rewards, again illustrating with examples. An overview of the web app is in Section 10, and in Section 11 we reflect upon our methodology and this study. Related work is discussed in Section 12, followed by conclusions.

2 Overview of the Communications Service

The components of the service are *sectors*, *sites*, and *channels*, which operate over various *frequencies*. There are 35 sectors (physical, disjoint entities) each of which is allocated a fixed set of frequencies, plus an emergency frequency. There are 17 sites, each with antennas, or *channels*, that transmit (Tx) and receive (Rx) on different frequencies. There is redundancy by design: every sector is allocated several frequencies, a frequency is covered by more than one site, and in every site there are (usually) idle backup channels. An example with 3 sectors and 6 sites is given in Fig. 2.

A **channel** is characterised by three parameters: whether it is receiver (Rx) or transmitter (Tx), the frequency, and the site reference. A **site** typically consists of a pair of receiver channels and a pair of transmitter channels. The convention is the main channel is known as channel A and the backup

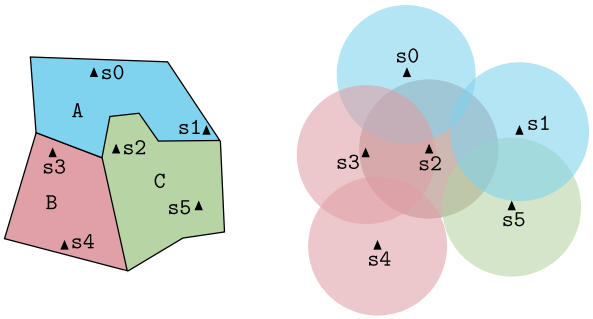


Fig. 2. Example with 3 sectors (A, B, C) and 6 sites (s0, ··· , s5). Each site is located in exactly one (physical) sector (left), but frequencies overlay sectors (right). Areas covered by the frequencies in each site: A = (s0, s1, s2), B = (s2, s3, s4) and C = (s2, s5). Note site s2 operates frequencies for all sectors. Also, s2 provides coverage for sector B even though it is physically located within the boundaries of sector C.

channel is channel B. Sites include sensors that monitor for power line status, communication link status, and physical intrusion and flooding. Together these sensors indicate the site *environment* and environment events are characterised as major events that cause a failure of the site (*e.g.* intrusion, power-line and backup generator failure, flooding) or minor events that mean the site is more likely to fail, but is still functioning (*e.g.* power-line failure but backup generators functioning). While in general the channels A and B operate independently of each other, the site environment affects them both simultaneously, *e.g.* a flood at the site causes both channels to fail.

An *n-ary* **sector** has n constituent sites[1]. Without loss of generality, for the remainder of this paper, unless otherwise stated, we assume sectors with three sites. Within a sector, the sites behave independently of each other and they may be at different distances from each other. This means that rates of events (*e.g.* that change status of a component) may differ from site to site, within a sector. For example, for a given sector, the rate of a given event at the first site may be different from the rate for the same event but at the second site (in that sector).

Components are monitored in real-time and their status is reported using the colour coding: green – functioning or serviceable; red – faulty, raise an alarm; blue – under-maintenance; amber – reduced-redundancy and possibly not fully functioning (for example, when one antenna goes down for a given frequency).

In more detail, the status for a receiver/transmitter is either: serviceable (green), faulty (red), site failure (red), or under-maintenance (blue). There is no reduced-redundancy for a single channel (*i.e.* no amber). Minor site environment events typically precede major site failures events and a site environment can have status: serviceable (green), minor site failure (amber), or major site failure (red). A site has status: serviceable (green), no-service (red), or reduced-redundancy (amber). Finally, the status of a sector is: serviceable (green), reduced-redundancy (amber), and no-service (red). The last is our ultimate concern as no-service for a sector is a catastrophic failure for the organisation.

1. Sites may be shared among several sectors.

3 Technical Background

Following [2], given a finite set of atomic propositions AP, a *(labelled) continuous-time Markov chain* (CTMC) is a triple $\mathcal{C} = (S, R, L)$ where S is a finite set of states with a designated initial state, $R : S \times S \to \mathbb{R}_{\geq 0}$ a rate matrix, and $L : S \to 2^{AP}$ a labelling of states. The exit rate $E(s) = \sum_{s' \in S} R(s, s')$ denotes the probability of taking a transition from s within t time units and is equal to $1 - e^{-E(s)\cdot t}$. If $R(s, s') > 0$ for more than one state s', a *race* between outgoing transitions from s exists. That is, the probability of moving from s to s' in a single transition is the probability that the delay of going from s to s' finishes before the delays of any other outgoing transition (from s). We use an informal, graphical notation for indicating the states and transitions of a CTMC, for example, in Fig. 3.

We use *Continuous Stochastic Logic* (CSL) [1], a stochastic extension of the Computational Tree Logic (CTL) that allows the expression of a probability measure of the satisfaction of a temporal property in either transient or steady-state behaviours. The formulae of CSL are state formulae Φ with path formulae Ψ:

$$\begin{array}{lcl} \Phi & ::= & \mathit{true} \mid a \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathcal{P}_{\bowtie p}[\,\Psi\,] \mid \mathcal{S}_{\bowtie p}[\,\Psi\,] \\ \Psi & ::= & \mathbf{X}\,\Phi \mid \Phi\,\mathbf{U}^{\mathrm{I}}\,\Phi \end{array}$$

where a ranges over a set of atomic propositions AP, $\bowtie \in \{\leq, <, \geq, >\}$, $p \in [0, 1]$, and I is an interval of $\mathbb{R}_{\geq 0}$.

Informally, path formula $\mathbf{X}\,\Phi$ is true on a path starting in s if Φ is satisfied in the next state following s in the path, whereas $\Phi_1\,\mathbf{U}^{\mathrm{I}}\,\Phi_2$ is true on a path ω if Φ_2 holds at some time instant in the interval I in a state s' in ω and at all preceding times Φ_1 holds. We additionally use the *eventually* path operator $\mathbf{F}$ (future) defined as $\mathbf{F}^{\mathrm{I}}\,\Phi \equiv \mathit{true}\,\mathbf{U}^{\mathrm{I}}\,\Phi$.

A *transient* formula $\mathcal{P}_{\bowtie p}[\,\Psi\,]$ is true in state s, denoted by $s \models \mathcal{P}_{\bowtie p}[\,\Psi\,]$, if the probability that Ψ is satisfied by the paths starting from state s meets the bound $\bowtie\ p$. A *steady-state* formula $\mathcal{S}_{\bowtie p}[\,\Psi\,]$ is true in a state s if the steady-state (long-run) probability of being in a state that satisfies Ψ meets the bound $\bowtie p$.

We use the PRISM probabilistic model checker [3], which allows us to leave the bound $\bowtie\ p$ unspecified. The probability is calculated in PRISM thus: $\mathcal{P}_{=?}[\,\Psi\,]$ and $\mathcal{S}_{=?}[\,\Psi\,]$. Additionally, PRISM allows for *experimentation*: the verification of an open formula, when the range, and step size of the variable(s) are specified. This allows us to plot *trajectories of property likelihood* over the free variable(s). For example, a typical property is $\mathcal{P}_{=?}[\,\mathbf{F}^{\leq t}\phi\,]$, which delivers the probability that we can reach a state in which ϕ is true, within t units of time (*e.g.* hours or minutes). We typically consider hours as the unit of time and vary t from 1 to 48 (*i.e.* behaviour over the next 48 hours).

PRISM allows for the augmentation of models with *rewards* (or, equivalently, costs) that are associated with states or transitions. The model checker can analyse properties that relate to the expected values of these rewards by using the $\mathcal{R}$ operator, which works in a similar fashion to the $\mathcal{P}$ and $\mathcal{S}$ operators, except that it depends on the name of a reward structure. We employ rewards on transitions and *cumulative* and *steady-state* reward properties. A cumulative (reward) property has the form $\mathcal{R}\{\texttt{reward}\}_{=?}[\,\mathbf{C} \leq t\,]$, which corresponds to the reward (named `reward`) accumulated along *all* paths until t time

units have elapsed. A steady-state (reward) property has the form $\mathcal{R}\{\texttt{reward}\}_{=?}\,[\,\mathbf{S}\,]$, which corresponds to the (named) reward per unit time, in the long-run.

The PRISM language supports high level specification of processes and compositionality. Processes are represented by modules consisting of non-deterministic choice over action-labelled guarded commands (which denote transitions); modules are composed over all common actions. A guarded command has the form:
$[action]\ guard \rightarrow rate : update$
meaning the process makes a transition to a state described by the *update* at the given *rate* when the *guard* is true. In the update, if x is a variable, then x' denotes the value of x in the next state.

Transitions are synchronised (*i.e.* occur simultaneously) between modules when they have the same action labels, in which case the rate of the synchronised transition is the *product* of all the individual rates. For example, given
$[action_1]\ guard_1 \rightarrow rate_1 : x' = 2$ and
$[action_1]\ guard_2 \rightarrow rate_2 : y' = 3$
when $guard_1$ and $guard_2$ are *true*, then in a next state, $x = 2$ and $y = 3$, with transition rate $rate_1 \cdot rate_2$.

4 Overview of Model

Key considerations when developing the model are the level of abstraction for components and events, dependencies between events, modelling the passage of time, and tractability.

Each component is represented by a PRISM module that includes a local variable for status. Dependencies between events are modelled by synchronisation. While strictly speaking, the labels of states in a CTMC are the propositions that are true in that state, here, we use a convenient state *labelling* to represent status and conflate labels with propositions. Tab. 1 contains a summary of the (labels of the) states that are represented in a model with a ternary sector, using regular expression notation, *e.g.* '|' for disjunction and '$*$' for wildcard. Since our primary concern is the availability of the higher level services, we do not reason about individual channels directly and thus employ a counter abstraction for the lowest level of the hierarchy. (A counter abstraction records the *counts* of processes in a particular state, rather than details of which process is in which state.)

The organisation expressed no views concerning how to model the passage of time, certainly there was no requirement for real-time. However, during our interviews with the operating engineers, it became clear that – at the level of abstraction in which they understood the system – the system exhibits many Markovian properties, *i.e.* time-homogeneous sojourn time distributions and behaviour that is determined by current state, not the process history. We therefore chose to adopt continuous CTMCs as the underlying semantics, which gives more detail than discrete time and also allows us to employ mean time between failure (MTBF) values: if the MTBF is r, then the associated rate for the (failure) event is $1/r$ and the probability the event occurs/has completed by time t is exponential: $1 - e^{r \cdot t}$. Note that any distribution can be arbitrarily well approximated by a phase-type distribution, *i.e.* a mixture of exponentials [4]. We will discuss this approach for the approximation of rates from historical data using hyper-Erlang distributions in Section 11.

After experimentation with a number of different abstractions, we found the following as an ideal compromise between detail, tractability and efficiency of analysis, and ease of expression of key properties. The components *channels*, *sites*, and *sectors* are modelled by PRISM modules, and events that must be synchronised have the same action labels. In graphical representations (*e.g.* Fig. 3) black transitions are internal (no synchronisation), red and green transitions denote synchronised events. Frequencies are not represented explicitly, as they are not relevant to sector service availability.

4.1 Channels

We use the following representation for individual channel states: S for serviceable, F for faulty, M for under-maintenance, and E for environment failure events. Recall there is no reduced-redundancy in a single channel (*i.e.* there is no amber for an individual channel).

We employ a *counter abstraction* whereby a **pair of A and B channels** is represented by a single module and state labels indicate the *counts* of the constituent channels. For example, state (label) SS means that both A and B channels are serviceable, state SF means that one channel is serviceable and the other is faulty (note, the label notation is not positional *i.e.* SF is not distinguished from FS). The CTMC for a pair of channels is given on the left-hand side of Fig. 3. States are colour coded to indicate status so whereas individual channels may be green/blue/red, channel pairs are green/amber/red[2]. For example, state SF is amber (reduced-redundancy) because one channel is serviceable and the other is not.

4.2 Sites

We represent a site by a triple (Tx,Rx,Env) consisting of the two channel pairs and a site environment. This means the notation for a site is positional: for example, site $(\mathsf{SF}, \mathsf{SS}, \mathsf{E0})$ is distinguished from $(\mathsf{SS}, \mathsf{SF}, \mathsf{E0})$. The former denotes a state where the transmitter is reduced-redundancy and the receiver is serviceable, whereas the latter denotes a state where the transmitter is serviceable and the receiver is reduced-redundancy.

States of sites are labelled and classified by three colours: W (working) for serviceable (green), R for reduced-redundancy (amber), and N for no-service (red). In PRISM, a **site** is represented by the concurrent composition of three modules: the transmitters, the receivers, and the site environment. Fig. 3 illustrates the resulting CTMC for a channel pair and site environment with symbolic rates a, b, c, *etc.* A key aspect of the model is the interaction between the site environment and the channels: the transition between E1 and E2 in the site environment synchronises with any channel transition to state E (red arrows in Fig 3); that is, an environment failure causes the channel to move to state E. Similarly, the (site environment) transition between E2 and E0 synchronises with the channels (reset) transition to SS

2. A component that is under-maintenance is not serviceable, therefore we have abstracted away from the "under-maintenance" (blue) class.

(green arrows in Fig. 3). Note, not all states are reachable. For example, the state $(\mathsf{SS}, \mathsf{E}, \mathsf{E2})$ is not possible because of synchronisation on site failure: when a site failure occurs, both the transmitter and receiver synchronise on this event and move to (channel) state E. We assume the rates for events for transmitters and receivers are identical (unless clearly specified otherwise) and if either the transmitter *or* receiver is no-service, then the entire site is no-service.

4.3 Sectors

Sectors (ternary) are represented as follows: WWW denotes a serviceable sector (green), NNN is a no-service sector (red), and amber is for a reduced-redundancy sector, which consists of all remaining states, *i.e.* the language defined by $L \setminus \{\mathsf{WWW}, \mathsf{NNN}\}$, where $L = (\mathsf{W}|\mathsf{N}|\mathsf{R})(\mathsf{W}|\mathsf{N}|\mathsf{R})(\mathsf{W}|\mathsf{N}|\mathsf{R})$. Note, this notation is positional.

As example, the code snippet in Fig. 4 gives the PRISM modules for the transmitters (Tx) and the environment for site X, in the context of rate declarations. Currently, PRISM does not allow text variables and so state labels are represented by (local) integer variables `s0_X` and `env_X`, *e.g.* 2 for FF, *etc.* Note the last two transition choices in the transmitters module, labelled by `alarm_major_X` and `fix_X`, cause the synchronisation with the site environment.

4.4 Rates

The model is governed by seven rates, which we refer to as $a, \ldots, g$; these are indicated in Fig. 3. Note there are two transitions from states with a faulty channel (F): a *quick*, local repair that returns to the serviceable state, and a *slower* transition to the under-maintenance state. The former reflects a failure that can usually be fixed remotely. The latter reflects the fact it may take some time for an engineer to physically reach a site and/or repair the fault. Interviews with engineers indicated the ratio between these rates is typically about $3:1$.

Rate a indicates the failure rate of a *single* channel. Intuitively, it describes the transition of a channel from state S to state F (downwards arrows in the Fig. 3). Since state SS contains two channels that can individually and independently fail, the rate for transition $\mathsf{SS} \to \mathsf{SF}$ must be $2a$.

Rate b is the rate of a quick repair. It describes the transition of a channel from state F to state S (without passing through an M state). Interviews with engineers revealed that the time to repair a single channel and a pair of channels is the same, we use b (*not* $2b$) as the rate for transition $\mathsf{FF} \to \mathsf{SF}$.

Rate c is the rate for slow repairs and describes the transition of a channel from state F to state M. Events of this kind are always in a race condition with b-rated events. In order to reflect the $3:1$ ratio between quick and slow repairs, c is defined as b/r, where r is an additional parameter for the expected ratio between quick and slow repairs.

Similarly, rate d is the duration of a repair of an under-maintenance channel (M), *i.e.* a transition of a channel from state M to state S.

Rates e and g are the rates for minor environment events and environment failures, respectively, and g is the rate of site repair.

The company gave us access to their SAP incident ticketing system, which they employ for long term storage of logged failures. The data logs record failure occurrences and repair durations, as well as a textual description, which allowed us to categorise events. Inference of rates was by manual inspection, sector by sector, for nominated time periods. Longer term, we aim to influence the design of readouts and tickets, and subsequently to automate the inference process.

As an example, we give results for one sector, which we call *FIR*, over a one year period: February 2012 to February 2013. The data included 61 alarms, of which 24 were environment events. From this data we calculated mean inter-failure times, which we then used to define failure rates (namely rates a, e and g), and we calculated and used repair duration times and mean repair duration times to define repair rates. The results are reported in Tab. 2.

Examination of the field data confirmed the inferred rates are of the expected orders of magnitude and also our assumption (as told to us during interviews with engineering staff) that the duration of repairs is independent of the number of channels (requiring repair). However, analysis raised some issues that require further consideration. First, some events were impossible to classify by analysing the chosen data set. For example, the textual descriptions for repair events did not specify whether an event was a quick or a slow repair. Therefore, in order to infer the ratio between rates b and c, we assumed that repair events with a duration *greater* than 2 hours were slow repairs. The inferred ratio was $6.6:1$, somewhat different from the expectation of the engineers. Second, rare events such as site failures did not occur in the time span covered by the data set; we had to inspect data from previous years to find an occurrence. Third, we identified two classes of events that may require a different representation model: dependent events such as failure of both A and B channels, and deterministic events such as scheduled maintenance. We will return to these issues in Section 11.

5 Temporal Logic Properties

We now turn our attention to CSL properties for analysis, considering atomic propositions, steady-state and transient properties, and example results.

Atomic propositions indicate the status (*i.e.* level of service) of channel pairs, sites, *etc.* and are defined in Tab. 3.

Steady-state properties express *long-run* behaviour[3]. Typically we examine steady-state behaviour for a given sector, computing the likelihood to be in a *serviceable* state, a *reduced-redundancy* state, or a *no-service* state, in the long-run. Namely, we consider three steady-state properties:

$$\mathcal{S}_{=?}\,[\,\texttt{serviceable\_sector}(A)\,]$$
$$\mathcal{S}_{=?}\,[\,\texttt{rr\_sector}(A)\,]$$
$$\mathcal{S}_{=?}\,[\,\texttt{noservice\_sector}(A)\,]$$

Transient properties express the probability of reaching a state that satisfies a proposition *within a period of time*. For our analysis, the crucial question is: what is the likelihood

3. Note, in a CTMC, long-run behaviour is a distribution over states.

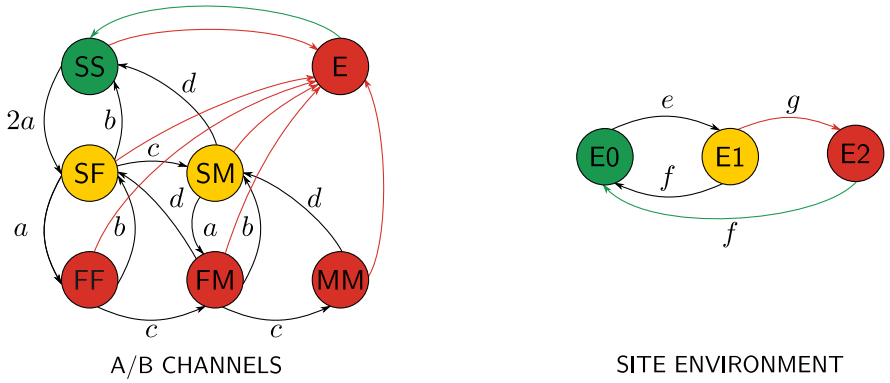


Fig. 3. CTMC for channel pair and site environment. Synchronisation on red and green transitions. Rates: a = channel failure, b = channel quick repair, c = channel slow repair, d = channel under-maintenance repair, e = minor environment event, f = site repair, g = environment failure.

TABLE 1
State labelling and colour coding.

Component	Colour	States	Description
channel	green	S	serviceable channel
	blue	M	under-maintenance channel
	red	F	faulty channel
	red	E	site failure
channel pair	green	SS	serviceable AB
(A,B)	amber	SF|SM	reduced-redundancy AB
	red	FF|FM|MM|E	no-service AB
site	green	SS, SS, E0	W serviceable site
(Tx,Rx,Env)	amber	SS, SS, E1	R reduced-redundancy site
	amber	SF,(SM|SF|SS),(E0|E1)	R reduced-redundancy site
	amber	SM,(SM|SF|SS)(E0|E1)	R reduced-redundancy site
	amber	(SM|SF|SS), SF(E0|E1)	R reduced-redundancy site
	amber	(SM|SF|SS), SM(E0|E1)	R reduced-redundancy site
	red	E, E, E2	N no-service site
	red	(FF|FM|MM) ∗ ∗	N no-service site
	red	∗(FF|FM|MM)∗	N no-service site
ternary sector	green	WWW	serviceable sector
(site,site,site)	amber	all other combinations	reduced-redundancy sector
	red	NNN	no-service sector

TABLE 2
Inferred rates from historical event data for sector FIR.

Rate	Inferred value
Mean inter-failure time	452 h
Mean repair time	18 h
Environment event	1107 h
Percentage of quick repairs	15
Environment failure	1 every 11.33 years

of reaching *no-service* in a given *sector* within time t. This is expressed by the transient property, for sector A,

$$\mathcal{P}_{=?}\left[\mathbf{F}^{\leq t}(\texttt{noservice\_sector}(A))\right] \quad (1)$$

By *experimentation* in PRISM with property (1) we can consider different instantiations of t, to plot how the likelihood changes over time. But there is another parameter to consider: the *state* from which we compute the likelihood.

TABLE 3
Atomic propositions for status of channel pairs, sites and sectors.

$$\begin{aligned}
\texttt{serviceable_chan}(c) &= (c = \mathsf{SS})\\
\texttt{serviceable_env}(e) &= (e = \mathsf{E0})\\
\texttt{serviceable_site}(s) &= \texttt{serviceable_chan}(\mathrm{Tx}_s)\\
&\quad \wedge \texttt{serviceable_chan}(\mathrm{Rx}_s)\\
&\quad \wedge \texttt{serviceable_env}(\mathrm{Env}_s)\\
\texttt{serviceable_sector}(A) &= \textstyle\bigwedge_{s \text{ site in } A} \texttt{serviceable_site}(s)\\
\texttt{rr_chan}(c) &= (c = \mathsf{SF}) \vee (c = \mathsf{SM})\\
\texttt{rr_env}(e) &= (e = \mathsf{E1})\\
\texttt{rr_site}(s) &= \neg(\texttt{serviceable_site}(s) \vee \texttt{noservice_site}(s))\\
\texttt{rr_sector}(A) &= \textstyle\bigvee_{s \text{ site in } A} \texttt{rr_site}(s)\\
\texttt{noservice_chan}(c) &= (c = \mathsf{FF}) \vee (c = \mathsf{FM}) \vee (c = \mathsf{MM}) \vee (c = \mathsf{E})\\
\texttt{noservice_env}(e) &= (e = \mathsf{E2})\\
\texttt{noservice_site}(s) &= \texttt{noservice_chan}(\mathrm{Tx}_s)\\
&\quad \vee \texttt{noservice_chan}(\mathrm{Rx}_s)\\
&\quad \vee \texttt{noservice_env}(\mathrm{Env}_s)\\
\texttt{noservice_sector}(A) &= \textstyle\bigwedge_{s \text{ site in } A} \texttt{noservice_site}(s)
\end{aligned}$$

```
// Parameterised rates
const double a=1/failure; const double b=1/qrepair;
const double c=b/r;       const double d=1/repair;
const double e=1/event;   const double f=1/fix_e_event;
const double g=1/site_failure;

module Site_Tx_X  // A/B channels
  // 0=SS, 1=SF, 2=FF, 3=SM, 4=FM, 5=SM, 6=E
  s0_X : [0..6];

  [] s0_X=0 -> 2*a:(s0_X'=1);
  [] s0_X=1 -> a:(s0_X'=2);
  [quick_0_X] s0_X=1 -> b:(s0_X'=0);
  [] s0_X=1 -> c:(s0_X'=3);
  [quick_0_X] s0_X=2 -> b:(s0_X'=1);
  [] s0_X=2 -> c:(s0_X'=4);
  [alarm_0_X] s0_X=3 -> a:(s0_X'=4);
  [repair_0_X] s0_X=3 -> d:(s0_X'=0);
  [quick_0_X] s0_X=4 -> b:(s0_X'=3);
  [] s0_X=4 -> c:(s0_X'=5);
  [repair_0_X] s0_X=4 -> d:(s0_X'=1);
  [repair_0_X] s0_X=5 -> d:(s0_X'=3);
  [alarm_major_X] true -> 1:(s0_X'=6);
  [fix_X] s0_X=6 -> 1:(s0_X'=0);
endmodule

module Site_env_X  // Site environment
  env_X : [0..2];  // 0=E0, 1=E1, 2=E2

  [] env_X=0 -> e:(env_X'=1);
  [alarm_major_X] env_X=1 -> g:(env_X'=2);
  [fix_X] env_X=1 -> f:(env_X'=0);
  [fix_X] env_X=2 -> f:(env_X'=0);
endmodule
```

Fig. 4. PRISM specification of transmitters and site environment modules, for site X. Rates are parameterised by constants `failure`, `qrepair`, *etc.*

(Note, hereafter we use configuration and state interchangeably.) In standard model checking, the given state is, by default, the initial state of the system. In our case, this would be the all-green configuration (serviceable channels, sites, sectors, *etc.*). However, we are considering a deployed system in which failures have occurred and the interesting cases are the *degraded*, amber configurations. Specifically, once we have reduced-redundancy, we require to quantify the criticality of the situation and take informed decisions – for example, do I need to fix a fault now, or can I wait? And if I can wait, for how long should I wait?

5.1 Example results

For example sector FIR, steady-state analysis results are given in the left-hand column in Tab. 4, indicating that in the long-run, the sector is serviceable for the majority of time (over 88%). We also analysed the historical data for that sector (over one year), to calculate the percentage time spent in a *serviceable* state, *etc.*, indicated in the right-hand column in Tab. 4. As can be seen, the two results compare well. This is not surprising given the model rates are derived from the same data set. For completeness, we give detailed results for the 52 configurations with probability $> 10^{-3}$ in Fig. 5a, noting the log scale for probabilities and use of shades of green to indicate degree of degradation/reduced-redundancy.

For transient property analysis, recall we require to choose a state (from which to perform the analysis). For sector FIR there are $389,017$ states, of which one is fully serviceable (WWW), $166,375$ are no-service (NNN), and $222,641$ are degraded, reduced-redundancy configurations. The degraded configurations we examined for the FIR sector, with three sites that we call A, B, and C, are given in Tab. 5. We refer to Tab. 1 for the definitions of W, R and N. Observe that both N and R can be the result of many different site configurations[4], so we selected SM, SM, E1 and MM, MM, E1 for each occurrence of R and N, respectively. We will explain in Sec. 7 how these eight configurations can be related to all the other possible degraded configurations.

Figures 5b, 5c, and 5d give the results for the probability of reaching a no-service configuration, from eight different degraded configurations, over a time interval of 48 hours.

We are considering a service in a safety-critical domain and so we expect the probabilities of no-service to be very low. However, observe the orders of magnitude difference on the y-axis. In Fig. 5b, the scale is 10^{-4}, whereas in Fig. 5c, the scale is 10^{-2}, and in Fig. 5d, the scale is 10^{-1}. Also, observe that in Fig. 5b the steepest trajectory is WWN, which contains one no-service site, and in Fig. 5d, the trajectory with highest probability, RNN, has two no-service sites. However, in the same figures, WNN also contains two no-service sites, but one serviceable site and the overall probability of service failure is constantly low.

Following similar analysis of different sectors with different topologies, we observed service availability increases as the number of sites grows. However, the contribution of each additional site to that increase decreases as more sites are added. The example in Fig. 6a illustrates this: the difference between 3- and 4-ary sectors is negligible. Overall, these results show that site redundancy (*i.e.* sector topology) is the most crucial factor affecting the behaviour of the system and we also conclude the system is not sensitive to the number of sites n, when $n > 3$. This implies the plots for the ternary site given in Fig. 5b to 5d are good approximations for sectors with more sites.

Ideally, for validation, we would compare our probabilistic model results against actual results, for different sets of degraded configurations. However, experimentation was not possible given this is a critical, deployed system. In future, we may be able to compare steady-state results against another trial period, if that data is available to us. For now, we report the engineers we interviewed found the results plausible and this type of analysis was useful to them.

Finally, we remark that channel redundancy *within* a site is a contributory factor to overall behaviour. When both channels A and B are serviceable, *i.e.* the site is W, then this redundancy guarantees safe service levels in the time frame $0 - 48$ hours, even in the extreme configuration in which only one site is in configuration W. For examples of this, see Fig. 5c, in which the plot for WRR is effectively flat, and similarly in Fig. 5d, in which the plots for WRN and WNN are also effectively flat.

6 Transient Properties for Decision Making

We now show, with reference to an example, how predictions of no-service can inform operational decision making. Consider the following scenario:

4. For example, R could be SF, SM, E0 or SF, SS, E1; N could be E, E, E2 or FF, SS, E0 or E, E, E2.

TABLE 4
Comparison of model long-run behaviour and manual analysis of historical data for sector FIR.

Status	Proposition	Model result	Result from historical data
serviceable	`serviceable_sector`(FIR)	88.46%	86.54%
reduced-redundancy	`rr_sector`(FIR)	11.53%	13.56%
no-service	`noservice_sector`(FIR)	10^{-8}%	0.00%

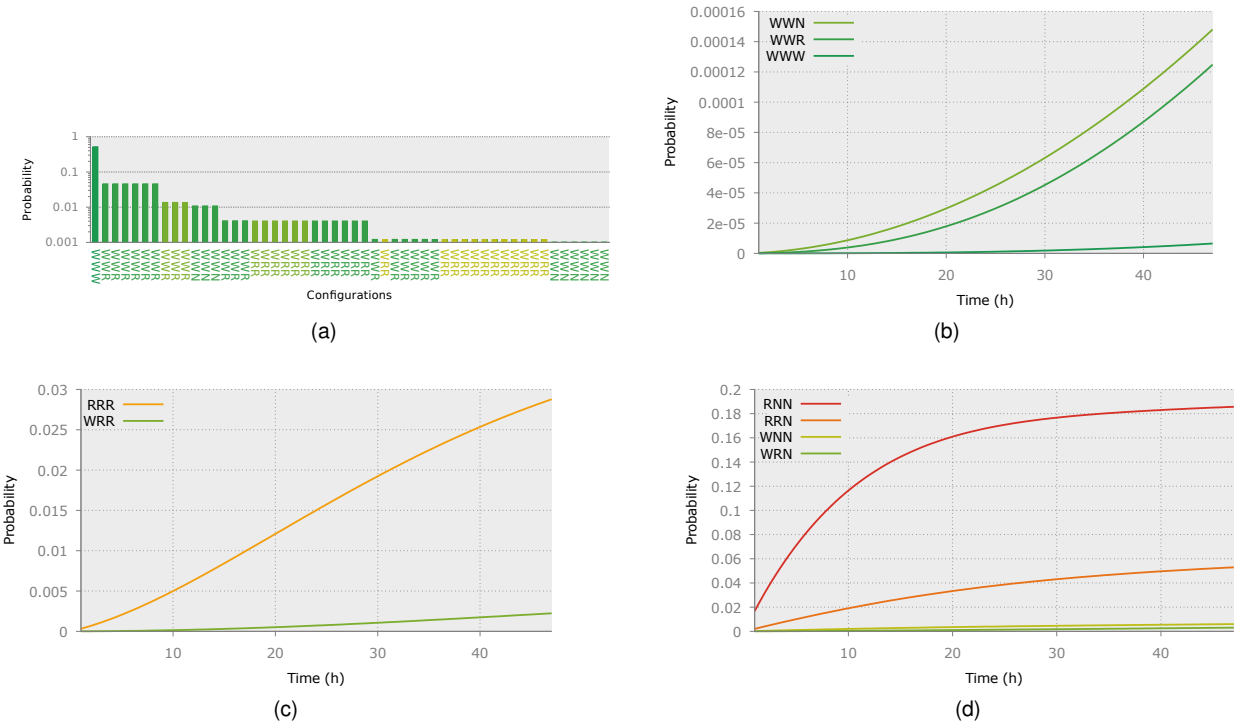


Fig. 5. Analysis results for sector FIR: (a) steady-state distribution of the 52 most probable configurations; probability of reaching a no-service configuration within 48 hours from selected initial configurations: (b) from WWW, WWR and WWN; (c) from WRR and RRR; (d) from RNN, WRN, RRN and WNN.

TABLE 5
Selected degraded configurations for sector FIR.

Site A	Site B	Site C
W	W	N
W	N	N
W	W	R
W	R	R
W	R	N
R	R	R
R	R	N
R	N	N

1) the current configuration of the system is RRR,
2) the system safety threshold (*i.e.* probability of no-service) is 4×10^{-3}, and
3) the mean repair time is 20 hours.

We predict the behaviour of the system by checking the transient property of reaching no-service as explained in the previous section: the plot, from the current configuration, is indicated with the red line in Fig. 8a. We remark that the red line denotes the expected property *if no assumptions are changed in the system*, *i.e.* if we assume the current failure and repair rates. Now consider the red shaded area in Fig. 8a, which indicates the probabilities above the safety threshold. The prediction shows that the system is likely to become unsafe after 20 hours. We reach the conclusion that within 20 hours, we want to be on *another trajectory* for the property, which is below the system safety threshold. We can do this by altering one or more rates so as to, in effect, transition to a more favourable configuration in an alternative CMTC, *i.e.* in one that is structurally the same but has different transition rates. For example, we could ensure that maintenance on one of the no-service sites is prioritised, effectively pushing down the mean repair time to 15 hours. In this case, the expected property of the system over the next 48 hours improves because the system becomes unsafe only after 34 hours instead of 20 hours. This is shown in Fig. 8a with the amber line. Now consider a configuration with one serviceable site, WRR; this is the configuration of the current system (RRR) after the site repair is successfully completed. The expected property is indicated by the green line in Fig. 8a. As can be seen, configuration WRR is much safer because within the time frame, the safety threshold is never reached.

Further, assume we choose to prioritise site maintenance and the one site is repaired after 20 hours (a random value

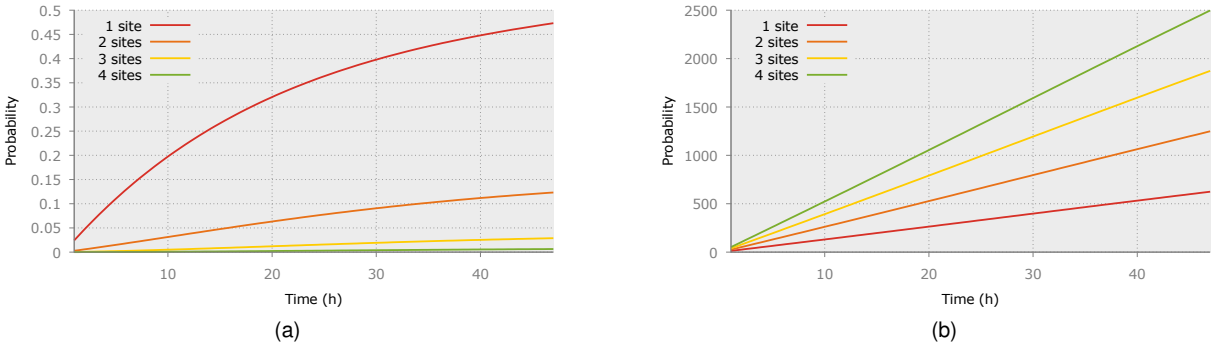


Fig. 6. Comparison of 1- to 4-ary sector topologies over 48 hours: (a) probability of no-service; (b) maintenance cost. Each site configuration is SM, SM, E1 ∈ R.

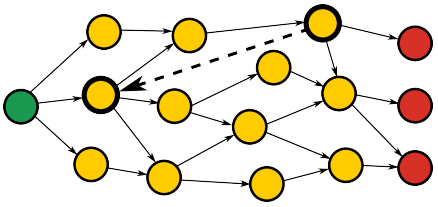


Fig. 7. Changing configuration after analysis of service availability.

taken by the exponential variable when the mean repair time is 15 hours). The transient property never reaches the system safety threshold, as shown by the green line in Fig. 8b. The red line shows the original trajectory: the probability of no-service if the repair is never performed. The discontinuity indicates exactly when the current state of the system is updated to WRR (at time 20h) because the site has became serviceable.

Fig. 7 is a pictorial representation of decision making; transitions indicate component failures. On the left we have the initial (green) state and on the right the (red) no-service states. The (amber) degraded configurations are the majority of states in between these two extremes: the dashed edge indicates the decision to make a discrete transition from one degraded configuration to another (more favourable) one.

Note, we can employ a similar approach to predict a no-service property of the system *after* specific events occur, such as scheduled maintenance or rare site failures (since they have such a small influence over transient probabilities, within a short time frame). In such cases, we are moving the trajectory *up*, instead of *down* at the discontinuity, *i.e.* we are increasing likelihood of no-service.

It may be tempting to consider as more favourable states those that are far (in terms of the number of transitions in the *shortest* path) from a no-service configuration, in the belief that configurations closer to a no-service configuration are more degraded than more distant configurations. But this is misguided, because the *length* of the path is possibly irrelevant. For example, from one amber configuration it may require only two events to reach a no-service configuration, yet both those events are very rare. On the other hand, from another amber configuration, it may take more discrete events (*i.e.* failures) before we reach no-service, yet all of them may be quite likely. So, in the former case, the probability of reaching no-service within a fixed time may well be lower than in the latter case, depending on choice of time interval.

We illustrate with an example. The distance to no-service is at most 6, because every configuration is at most 3 steps away from a site failure. For example, even configuration WWW becomes NNN in 6 steps when, in each site, transitions E0 → E1 → E2 take place. The first one is only two transitions away from no-service, while the second one is four transitions away. This is because two channel failures FS → FF and two site failures E0 → E1 → E2 are needed to reach NNN in the first and in the second cases, respectively. But the first is not *more* degraded because the probability of reaching a no-service configuration within 48 hours is 2.251×10^{-5} and 3.304×10^{-4} for the first and the second configurations, respectively. The second configuration is more degraded – despite being more distant from no-service than the first one: an example of how intuition can be misleading in a probabilistic setting.

7 Envelopes of Behaviour

When we require to reason about behaviour from a *given* degraded configuration, we know exactly the configuration of all the component sites. If we do not know the exact configuration of all the components, we can simply select some representatives, as above. An interesting question is:

can we quantify the effect of the choice of status of the lowest level components on the analysis?

To answer this we consider how to identify, for a given transient property, upper and lower probability bounds induced by the possible combinations of the status of the lower level components. These bounds allow us to define *envelopes of behaviour*, for a property. We illustrate through examples of the likelihood of reaching no-service, within 48 hours, when applied to the FIR sector[5].

FIR is a ternary sector that has 25 possible degraded, or reduced-redundancy configurations.[6] This can be reduced to 8 cases, by symmetry. For each configuration, we define the lower bound to be the lower bound of reaching no-service for the most degraded site, and conversely we define the upper bound to be the upper bound of reaching no-service for the least degraded site. The lower/upper bounds

5. This is the property defined by (1).

6. Each site can take one of 3 forms and there are $3 \times 3 \times 3$ configurations, from which we remove WWW and NNN.

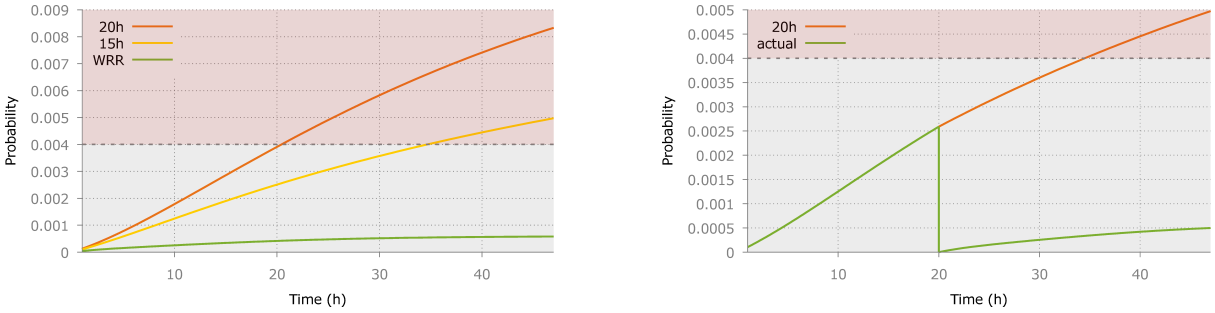


Fig. 8. Transient property for service availability: (a) comparison of different configurations and repair rates and system safety threshold; (b) before and after discrete transition to a new state, in context of 4×10^{-3} system safety threshold.

for the most/least degraded sites are found by analysis of all the possible component sites. We illustrate by example. Assume the first two sites are W, then perform transient property analysis (for the *sector*) for all possible R and N configurations. For the former, there are 17 cases to consider, which reduce to 11 after removing symmetric cases; for the latter, there are 37 to consider, which reduce to 31. Results are shown in figures 9a (WWR) and 9b (WWN); in each, for comparison, we also give the result for SS, SS, E0, which of course is not degraded. To ease interpretation, configurations are colour coded according to the level of degradation: recall, green means no degradation, while red means consistent degradation (*i.e.* across all sites). It is not surprising that two clusters occur, depending on the site environment: all the configurations with E1 are considerably more degraded than the configurations with E0.

For a sector configuration C and property p, we define the *upper (lower) bound of the envelope of behaviour* $C \uparrow_p$ ($C \downarrow_p$) as the uppermost (lowermost) trajectory for property p over all combinations of the status of the lower level components in C, for a given time interval.

For example, the lower bound for configuration WNN, WNN $\downarrow_p$, is the property trajectory obtained by selecting SS, FF, E0 for both N sites. The lower bound for WWR is the property trajectory obtained by selecting SS, FS, E0 for the R site.

Details of the site configurations for each bound are given in Fig. 6, with the corresponding probabilities of no-service over 48 hours shown in Fig. 10. In the latter, for simplicity, we omit the subscript p. Note that the red shaded area between RNN $\uparrow$ and RNN $\downarrow$ indicates the envelope for any configuration in the form RNN.

7.1 Limitations

In this case study we have been able to assume symmetries in configurations. However, if for example, the transmitters and receivers have different rates, then when identifying bounds for R and N, we would have to consider all 17 WWR configurations (instead of 11) and all 37 WWN configurations (instead of 31). Additionally, if rates vary across sites, then analysis has to be repeated independently for each site.

Another possible shortcoming is that, in some cases, the envelope may be too broad. For example, consider configuration RNN, with envelope bounds RNN $\downarrow_p$ and RNN $\uparrow_p$ (as indicated in Fig. 10). That means

TABLE 6
Site configurations for bounds for property p = *probability of reaching a no-service configuration within 48 hours.*

Bound	Site A	Site B	Site C
RNN $\uparrow_p$	SM,SM,E1	E,E,E2	E,E,E2
RRN $\uparrow_p$	SM,SM,E1	SM,SM,E1	E,E,E2
RRR $\uparrow_p$	SM,SM,E1	SM,SM,E1	SM,SM,E1
WNN $\uparrow_p$	SS,SS,E0	E,E,E2	E,E,E2
WRN $\uparrow_p$	SS,SS,E0	SM,SM,E1	E,E,E2
WRR $\uparrow_p$	SS,SS,E0	SM,SM,E1	SM,SM,E1
WWN $\uparrow_p$	SS,SS,E0	SS,SS,E0	E,E,E2
RNN $\downarrow_p$	SS,FS,E0	SS,FF,E0	SS,FF,E0
WWR $\uparrow_p$	SS,SS,E0	SS,SS,E0	SM,SM,E1
WNN $\downarrow_p$	SS,SS,E0	SS,FF,E0	SS,FF,E0
RRN $\downarrow_p$	SS,FS,E0	SS,FS,E0	SS,FF,E0
WRN $\downarrow_p$	SS,SS,E0	SS,FS,E0	SS,FF,E0
RRR $\downarrow_p$	SS,FS,E0	SS,FS,E0	SS,FS,E0
WWN $\downarrow_p$	SS,SS,E0	SS,SS,E0	SS,FF,E0
WRR $\downarrow_p$	SS,SS,E0	SS,FS,E0	SS,FS,E0
WWR $\downarrow_p$	SS,SS,E0	SS,SS,E0	SS,FS,E0

the probability of no-service at 48 hours may take any value in the range $[2.54 \times 10^{-4}, 2.96 \times 10^{-1}]$. If this is considered too broad, we may specify intermediate bounds for different sub-classes of RNN configurations (*i.e.* other than the worst and best case scenarios). For example, upper and lower bounds for RNN configurations with all site environments set to E1 are trajectories with initial states (SM,SM,E1)(MM,MM,E1)(MM,MM,E1) and (SS,SS,E1)(SS,FF,E1)(SS,FF,E1) respectively; this can be confirmed by inspecting the plots in figures 9a and 9b.

8 Recoverability and Survivability

So far, we have considered properties that define the likelihood of reaching a no-service state from degraded (*i.e.* reduced-redundancy) configurations. Now, we turn our attention to properties *after reaching* a no-service state. We consider two stochastic properties: *recoverability* and *survivability*, as proposed in [5]. Both properties refer to behaviours *after* a disaster has occurred – in our case the "disaster" is reaching a no-service state. We assume here that resources are readily available after such a disaster, *i.e.* repair rates are unchanged, though we note that we could

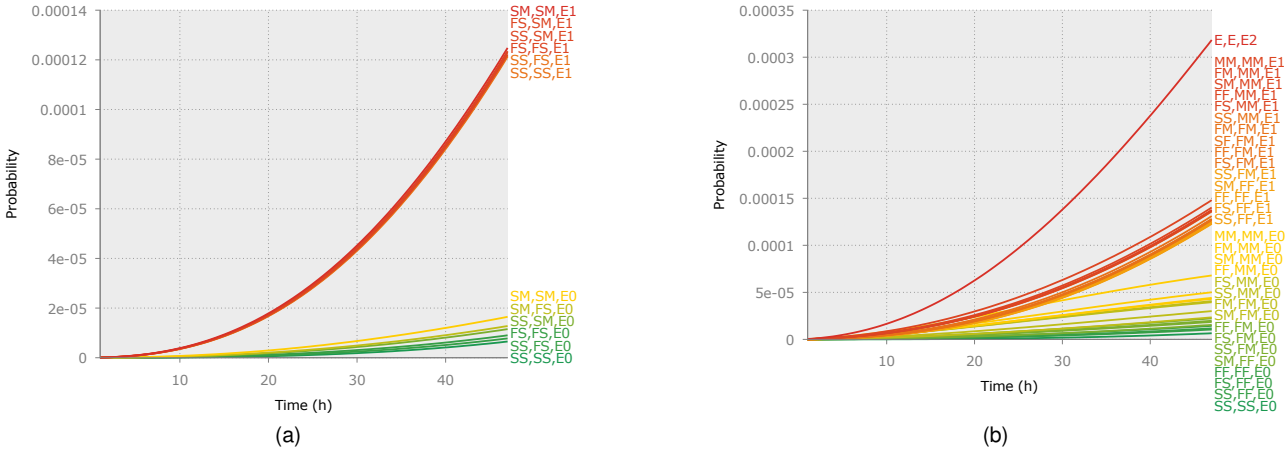


Fig. 9. Probability of no-service within 48 hours for all the WWR (a) and the WWN (b) configurations.

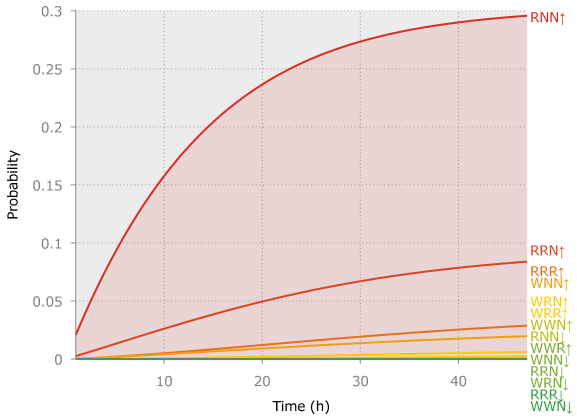


Fig. 10. Envelopes of behaviour for property p for degraded configurations. p = *probability of reaching a no-service configuration within 48 hours.*

alter repair rates to reflect a different availability of resources (see Section 11). As before, we refer to examples taken from the FIR sector. We also write `noservice` as a shorthand for `noservice_sector`(FIR).

Recoverability is the probability of recovering service within time bound t. In our model, this is expressed by:

$$\mathcal{P}_{=?}\left[\mathbf{F}^{\leq t}(\neg\texttt{noservice})\right] \tag{2}$$

We denote this property by r and give results over time interval $t \leq 48$ h in Fig. 11a. The state(s) from which we perform analysis are combinations of the bounds for N identified in Fig. 9b. Observe the envelope of behaviour for any no-service configuration is given by:

$$\mathsf{NNN}\uparrow_r = (\mathsf{SS},\mathsf{FF},\mathsf{E1})(\mathsf{SS},\mathsf{FF},\mathsf{E1})(\mathsf{SS},\mathsf{FF},\mathsf{E1}) \text{ and}$$
$$\mathsf{NNN}\downarrow_r = (\mathsf{E},\mathsf{E},\mathsf{E2})(\mathsf{E},\mathsf{E},\mathsf{E2})(\mathsf{E},\mathsf{E},\mathsf{E2})\ .$$

Generally, the patterns are preserved, *i.e.* SS,FF,E0 is less degraded than MM,MM,E0. However, it interesting to observe that configurations with E0 take longer to recover than configurations with E1, which is the opposite behaviour we observed for N sites in reduced-redundancy configurations. Note that the property r (trivially) evaluates to (probability) 1, for any time bound, when the initial state is a reduced-redundancy configuration, this is because in that state $\neg$`noservice` already holds.

Survivability offers a little more information, indicating the ability of a no-service configuration to recover service, in a timely manner and within a given probability bound. This is a subtle elaboration on property (2) and expressed by:

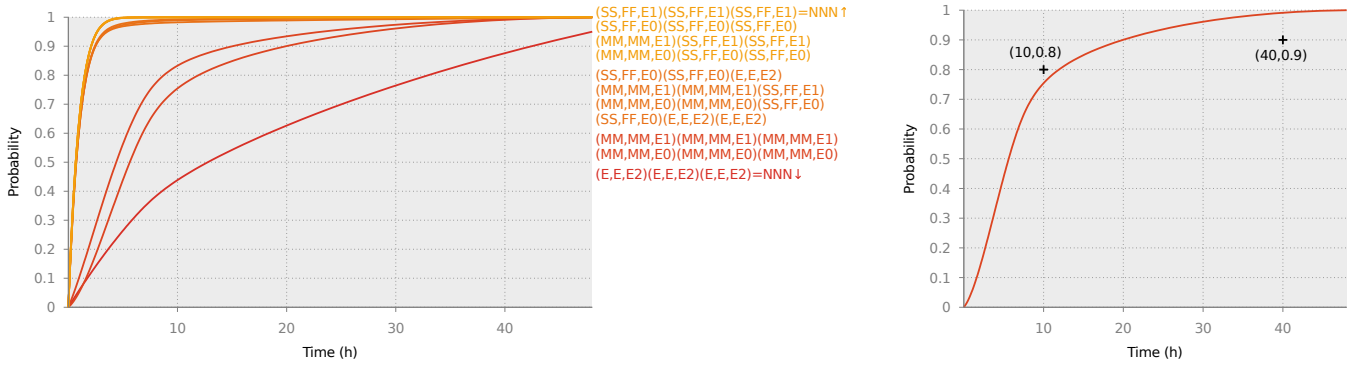
$$\texttt{noservice} \Rightarrow \mathcal{P}_{\leq q}\left[\mathbf{F}^{\leq t}(\neg\texttt{noservice})\right] \tag{3}$$

There are two free variables, a time bound t and a probability q. If we inspect Fig. 11a, for a particular (no-service) configuration, we can conclude the configuration is *survivable* for all points (t, q) on and below the curve, whereas the points above the curve indicate time bound/probability pairs for which the system is *not survivable*. For example, the plot for (MM,MM,E0)(MM,MM,E0)(MM,MM,E0) in Fig. 11b indicates the configuration is not recoverable within 10 hours with probability greater than 0.8, therefore it is not survivable. If we choose pair $(40, 0.9)$ instead, the configuration is recoverable (the point lays below the curve), thus survivable. Further, we can conclude the *system* is survivable for $(48, 0.9)$ because all the configurations are survivable for this pair (this follows from the lower bound NNN $\downarrow_r$).

9 Cost Analysis with Transition Rewards

We may analyse *costs* of particular behaviours, and make decisions based on those costs, using the facility in PRISM to specify rewards and perform analysis of reward-based properties. As an example, we associate a cost with each transition in the model representing a maintenance intervention, and then reason about (*i.e.* forecast) the expected maintenance costs over one month period, for a given configuration, using a *cumulative reward* property, as follows.

We augment our model with the reward structure for site X given in Fig. 12. Each transition is assigned a cost – most have no precondition (*i.e.* the condition is simply `true`), but a condition is used to disambiguate the two transitions labelled by `fix_X`. For confidentiality reasons, we refer here to costs that are fictional but reflect actual proportions: `q` = 10, `r` = 100, `s0` = 100, `s1` = 3000. The cumulative reward property is $\mathcal{R}\{\texttt{cost}\}_{=?}\left[\mathbf{C} \leq 730\right]$ where 730 is the time bound expressed in hours. Results for



(a) (b)

Fig. 11. Recoverability of eleven no-service configurations within 48 hours (a) and survivability of example configuration (MM,MM,E0)(MM,MM,E0)(MM,MM,E0) for pairs $(10, 0.8)$ (not survivable) and $(40, 0.9)$ (survivable) (b).

```
rewards cost
  // Transmitter
  [quick_0_X] true: q;[repair_0_X] true: r;
  // Receiver
  [quick_1_X] true: q;[repair_1_X] true: r;
  // Site environment
  [fix_X] e_X=1: s0;[fix_X] e_X=2: s1;
endrewards
```

Fig. 12. PRISM specification of `cost` reward structure for site X.

19 configurations are given in Tab. 7. Note the maintenance cost of the more degraded configurations are higher than those for less degraded configurations. This is because when considering short time windows, maintenance interventions are more likely to be scheduled in more degraded configurations.

Cost analysis can be used along with safety analysis for decision making as described in Section 6. We note that while total maintenance costs increase linearly with the number of sites (when changing sector topology), as indicated in Fig. 6b, where each site is in the upper bound $\mathsf{SM,SM,E1} \in \mathsf{R}$ configuration, recall the contribution to overall safety, from each site, decreases exponentially with the number of sites, as indicated in Fig. 6a.

Finally, we note that the reward structure defined in Fig. 12 also allows for the analysis of steady-state properties *i.e.* the cost in the long-run. The corresponding formula is $\mathcal{R}\{\mathtt{cost}\}_{=?}[\,\mathbf{S}\,]$ which for the example FIR sector evaluates to 6.93. This represents the expected long-run cost rate per unit of time (one hour in our model), and is independent of the initial configuration.

10 Implementation

We developed a multi-platform web app, as illustrated in Fig. 13 running on an Android tablet. The system is a client-server architecture, implemented in Node.js[7] that relies on remote PRISM instances for heavyweight computations. The web app supports a simple interface for users unfamiliar with the underlying model, enabling simple instantiation, using sliders, of parameters such as rates for events, sector topologies, percentage of quick repairs, and current configuration. Default values are provided. Analysis results displayed on the device are in the formats used in this paper (*e.g.* graphs or bar charts) or PRISM textual output.

7. https://nodejs.org/

TABLE 7
Expected monthly maintenance cost for 19 configurations

Site A	Site B	Site C	Monthly cost
E,E,E2	E,E,E2	E,E,E2	13448.19
SM,SM,E1	E,E,E2	E,E,E2	11851.93
SS,SS,E0	E,E,E2	E,E,E2	10558.47
SM,SM,E1	SM,SM,E1	E,E,E2	10255.67
SS,SS,E0	SM,SM,E1	E,E,E2	8962.21
SM,SM,E1	SM,SM,E1	SM,SM,E1	8659.41
SS,SS,E0	SS,SS,E0	E,E,E2	7668.75
SS,SS,E0	SM,SM,E1	SM,SM,E1	7365.95
SS,SS,E0	SS,SS,E0	SM,SM,E1	6072.49
SS,FF,E0	SS,FF,E0	SS,FF,E0	4964.70
SS,SF,E0	SS,FF,E0	SS,FF,E0	4933.75
SS,SF,E0	SS,SF,E0	SS,FF,E0	4902.81
SS,SS,E0	SS,FF,E0	SS,FF,E0	4902.81
SS,SF,E0	SS,SF,E0	SS,SF,E0	4871.86
SS,SS,E0	SS,SF,E0	SS,FF,E0	4871.86
SS,SS,E0	SS,SF,E0	SS,SF,E0	4840.92
SS,SS,E0	SS,SS,E0	SS,FF,E0	4840.92
SS,SS,E0	SS,SS,E0	SS,SF,E0	4809.97
SS,SS,E0	SS,SS,E0	SS,SS,E0	4779.02

11 Discussion

Methodology. With the exception of envelopes of behaviour, the methods we employ are not novel, our contribution is the way we have employed them: how we modelled the service, what we analysed and how. The new concept of envelopes of behaviour allows us to quantify the effects of lower level components on properties (about higher level components) by identifying best and worst case scenarios, which can be useful when assessing impacts and priorities. We remark that the properties of interest here, *e.g.* (1), (2), do not expose the full expressiveness of the logic. Properties could be more complex formulae such as *when channel c is reduced-redundancy then the probability of sector A being serviceable, between times* 10 *and* 30*, without channel d becoming reduced-redundancy is greater than .75*:

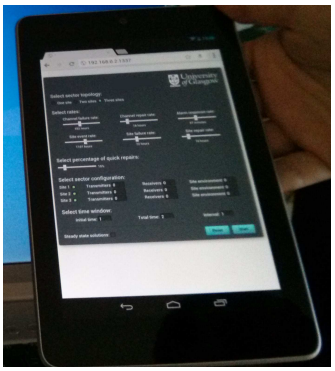


Fig. 13. Web app for setting rates and topologies running on an Android tablet.

$$\texttt{rr\_chan}(c) \Rightarrow \mathcal{P}_{\leq .75}\left[\neg \texttt{rr\_chan}(d)\mathbf{U}^{[10,30]}\texttt{serviceable\_sector}(A)\right]$$

Applicability. The framework is applicable to other component based systems exhibiting Markovian behaviour. Key steps are identifying the components and the hierarchies, the events representing failures, repairs *etc.* and their rates, and, crucially, the dependencies *between events* for a component and dependencies *between components*. Dependencies are represented by common (named) events in PRISM. The overall system is simply the concurrent composition of all the components. In the example presented here, the rates do not depend on the number of components involved in a synchronisation (*i.e.* the products involve identities). Other application rates may, for example, depend on mass action kinetics, which is easily accommodated in our framework. We did not encounter state space explosion problems nor numerical difficulties, mainly because our modelling approach involves counter abstraction and we do not analyse the system from the standard "initial state", but from degraded configurations that can occur as the system is running (regardless of the probability of reaching them). We remark that in the event of state explosion problems, a solution would involve simulation methods, which generally scale much more effectively.

Scheduled events, non-stochastic and spatial aspects. One consequence of modelling within a Markovian framework is that we cannot easily model *scheduled* and non-stochastic events. There are several possible solutions such as a) remaining within the CTMC paradigm but modelling the new events with hyper-Erlang distributions, which means the state space explodes because of all the interleaving/expansions or b) modelling with probabilistic timed automata, which means the (exponentially distributed) failure rates are discretised by a geometric distribution, or c) modelling with hybrid CTMCs that model the scheduled events as discrete switching between CTMCs. Each of these would result in (possibly unnecessarily) more complicated semantics and analysis techniques. On the other hand, it would be relatively simple to encode any spatial aspects of the system (*e.g.* if transmitters/receivers are mobile) using (stochastic) bigraphs with sharing [6]. Given the data we have seen for scheduled maintenance here is sparse, we have not yet incorporated it into the model.

Validation. Ideally, this involves experimentation with the actual system and comparing results with probabilities of the no-service property in the model, for given configurations and time bounds. However, experimentation was not possible because we are dealing with a deployed, critical service: a service failure in the live system is catastrophic. We recognise that traditional statistical validation remains an open question, a solution might be to gain access to the infrastructure for a sector that is not in operation and use it as a testbed for experimentation.

Rates. There are two simple ways to alter rates in the model. The first is simply to alter the rate parameters using the web app, as described above. The second is to encode evolving rates *within* the model, for example, to reflect wear and tear over time or anticipated rate changes after a discrete event. This is easy to do within the PRISM language; we did not do so here because there was no requirement. Rate evolutions could be informed by Bayesian learning over observed data. However, if there are dependencies between events, then the structure of the model (*e.g.* synchronisations between updates in the relevant modules) would have to be altered by hand.

Dependencies. When we inspected the historical data, we found evidence of dependencies between channel A and B faults. The cause of dependencies is as yet unclear, but in part it may be due to the formats for recording faults and the use of free text. However, there may be further contributory factors in that transmitters and receivers are usually commissioned at the same time, and more likely, communications network failures typically affect both channels simultaneously. Determining the causes requires further investigation; modifications to the model would likely include synchronisation of Tx and Rx failures.

12 Related work

The field of probabilistic verification (in particular by model checking) has grown considerably since the 1980s (see [7]). But to our knowledge there has been little work using these methods to predict future service availability and inform *operational* decisions in the presence of component failures.

If we choose to model scheduled maintenance by deterministic, timed events, as mentioned above, then we may also consider how these are handled in [8], where a system with *rejuvenation* – a system that is periodically stopped and then restored in a robust state after maintenance – is modelled as a Markov regenerative process and then Markov renewal theory [9] is applied to carry out quantitative analysis. The work of Trivedi *et al.* [10], [11] would be relevant in that context.

Another approach is considered in [12], which employs a partially observable Markov decision model for a maintenance problem. How these models may provide a suitable semantic underpinning for our framework, especially with regard to reasoning about logical properties, is further work. We note also the possible state space explosion and numerical simulation difficulties in the presence of rare events [13]. We use PRISM in preference to other model checkers, such as the MRMC model checker [14] or SMART [15], because it supports modules, event synchronisations, experiments and CSL rewards: these are integral to our framework. Finally,

parts of this study and preliminary results were presented in [16].

13 Conclusions and Future Work

We have presented a stochastic framework that supports decision making in the event of component failures and our experience of applying it to a critical communications service deployed within a large organisation. A typical question it helps address is: when a component fails how urgent is a repair? In the system we have considered, this involves being able to answer questions such as: from a given degraded configuration, for a given future time period and a safety threshold, what is the likelihood of the system becoming unsafe?

While modelling for reliability analysis with CTMCs is well known, this was not an obvious solution to the problem as originally presented. The novelty of our contribution is the way we have employed CTMCs as models and the stochastic temporal logic CSL to address the questions posed to us, rather than textbook analysis and simulations. We also defined and applied the new concept of envelopes of behaviour, which allow us to quantify the effects (best/-worst case) of the status of lower level components on properties about higher level components. The framework can be used in a variety of ways: from evaluating whether an architecture meets service requirements, to assessing the impact of prioritisation of repairs in real-time.

The temporal logic allows us to define and explore both standard steady-state and transient properties concerning the likelihood of service failure within certain time bounds, and envelopes of behaviour that give the lower and upper (probability) bounds of a property (*e.g.* of service failure within n hours) induced by different combinations of lower level components. We also use the logic to quantify and explore recoverability and survivability after a loss of service, as well as rewards to forecast maintenance costs, both cumulatively and in steady-state. The framework is implemented in the PRISM language and model checker, making extensive use of high level features such as modules, synchronisation and rewards and it is supported by a web-based, lightweight interface that allows users unfamiliar with PRISM to interact with the model.

Much future technical work is possible, for example, on model validation in the context of a critical, deployed system and combining deterministic and stochastic events within the models.

Acknowledgments

We thank our industrial collaborators (Suki Lal, Dave Bellshaw and Terry Wright) for help and guidance throughout the project. This work was partially funded by the EPSRC grant *Verifying Interoperability Requirements in Pervasive Systems* EP/F033206/1, the University of Glasgow EPSRC funded Impact Acceleration Account and Sevegnani's EPSRC Doctoral Prize Research Fellowship.

References

[1] C. Baier, B. R. Haverkort, H. Hermanns, and J.-P. Katoen, "Model-Checking Algorithms for Continuous-Time Markov Chains," *IEEE Trans. Software Eng.*, vol. 29, no. 6, pp. 524–541, 2003.

[2] M. Kwiatkowska, G. Norman, and D. Parker, "Stochastic model checking," in *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM'07)*, 2007.

[3] ——, "PRISM 4.0: Verification of probabilistic real-time systems," in *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, ser. LNCS, vol. 6806. Springer, 2011, pp. 585–591.

[4] W. Bux and U. Herzog, "The phase concept: Approximation of measured data and performance analysis," *Computer Performance*, pp. 23–38, 1977.

[5] L. Cloth and B. R. Haverkort, "Model checking for survivability!" in *Quantitative Evaluation of Systems, 2005. Second International Conference on the*, Sept 2005, pp. 145–154.

[6] M. Sevegnani and M. Calder, "Bigraphs with sharing," *Theoretical Computer Science*, vol. 577, p. 43 74, 2015.

[7] J.-P. Katoen, "The Probabilistic Model Checking Landscape," *LICS16 (Logics in Computer Science)*, 2016.

[8] S. Garg, A. Puliafito, M. Telek, and K. Trivedi, "Analysis of software rejuvenation using markov regenerative stochastic petri net," in *Software Reliability Engineering, 1995. Proceedings., Sixth International Symposium on*, 1995, pp. 180–187.

[9] R. Pyke, "Markov renewal processes: definitions and preliminary properties," *The Annals of Mathematical Statistics*, pp. 1231–1242, 1961.

[10] D. M. Nicol, W. H. Sanders, and K. S. Trivedi, "Model-based evaluation: from dependability to security," *Dependable and Secure Computing, IEEE Transactions on*, vol. 1, no. 1, pp. 48–65, 2004.

[11] P. E. Heegaard and K. S. Trivedi, "Network survivability modeling," *Computer Networks*, vol. 53, no. 8, pp. 1215–1234, 2009.

[12] R. Srinivasan and A. Parlikad, "Semi-markov decision process with partial information for maintenance decisions," *Reliability, IEEE Transactions on*, vol. 63, no. 4, pp. 891–898, Dec 2014.

[13] D. Reijsbergen, P.-T. de Boer, W. R. W. Scheinhardt, and B. R. Haverkort, "Rare event simulation for highly dependable systems with fast repairs," *Perform. Eval.*, vol. 69, no. 7-8, pp. 336–355, 2012.

[14] J.-P. Katoen, I. S. Zapreev, E. M. Hahn, H. Hermanns, and D. N. Jansen, "The ins and outs of the probabilistic model checker MRMC," *Performance Evaluation*, vol. 68, no. 2, pp. 90 – 104, 2011, advances in Quantitative Evaluation of Systems QEST 2009.

[15] G. Ciardo, R. L. Jones, A. S. Miner, and R. Siminiceanu, "Logic and stochastic modeling with SMART," *Performance Evaluation*, vol. 63, no. 6, pp. 578–608, 2006.

[16] M. Calder and M. Sevegnani, "Do I need to fix a failed component now, or can I wait until tomorrow?" *Proceedings Tenth European Dependable Computing Conference (EDCC 2014), IEEE*, 2014.

Muffy Calder is Professor of Formal Methods and Vice-Principal and Head of the College of Science and Engineering at Glasgow University. Previously she was the Chief Scientific Adviser for Scotland. She received her PhD in Computational Science from the University of St. Andrews, Scotland in 1987. She has collaborated with scientists and engineers from a wide range of disciplines, from electrical and aerospace engineering, to cancer and cardiovascular medicine. She is a Royal Society Wolfson Merit Award Holder and a Fellow of the Royal Academy of Engineering, the Royal Society of Edinburgh, and the British Computer Society. She was awarded the OBE for services to Computer Science in 2011.

Michele Sevegnani is an EPSRC Doctoral Prize Research Fellow at the University of Glasgow, based in the School of Computing Science. He received a PhD in Computing Science from the University of Glasgow, Scotland in 2012 and an MSc in Bioinformatics jointly from the universities of Edinburgh (Scotland) and Trento (Italy) in 2008. His research focusses on the theory of bigraphs and how to use it to reason about safety, reliability and predictability of location-aware, event-based, software systems, particularly complex systems that are already deployed.