Do I need to fix a failed component now, or can I wait until tomorrow?

Muffy Calder and Michele Sevegnani
School of Computing Science, University of Glasgow, Glasgow, G12 8RZ, UK.
Email: {muffy.calder,michele.sevegnani}@glasgow.ac.uk

***Abstract*—We investigate how predictive event-based modelling can inform operational decision making in complex systems with component failures. By relating the status of components to service availability, and using stochastic temporal logic reasoning, we quantify the risk of service failure now, and in the future, after a given elapsed time. Decisions can then be taken according to those risks. We demonstrate the approach through application to an industrial case study system in which component failures are sensed and monitored. The system has been deployed for some time. A novel aspect is we calibrate the model(s) according to inferences over historical field data, thus the results of our reasoning can inform decision making in the actual deployed system.**

keywords: ctmc, communications service, safety-critical, temporal logic, predictive modelling.

I. Introduction

Operational decision making about which components to fix and when, in the event of component failures and reduced redundancy in a deployed critical service, is difficult. Ideally, when failures are uncovered (e.g. through monitoring and sensing), an engineer would fix them immediately. But this might not be possible due to limited resources and/or physical distance to a device. So how does an engineer prioritise and make best use of their resources, while still ensuring the service is operating within acceptable levels of risk of failure? We hypothesise that in systems in which failures are monitored and sensed, predictive modelling and reasoning with a stochastic temporal logic can inform decision making by relating the status of assets (i.e. components) to service behaviour so that the risk of service failure now, and in the future, can be quantified, under assumptions about failure rates. The aim of this paper is to demonstrate how event-based, stochastic formal modelling and analysis by model checking temporal logic properties provides such a predictive modelling framework. We demonstrate the approach via application to an industrial case study: a communications service for a safety-critical system and a software system that senses and monitors the status of communication links and the service. The system has been developed and deployed over a number of years in a large global company. While we focus here on the case study, our approach and results are applicable to a wide range of sensor based, monitoring systems and critical services.

We relate asset level behaviour to service level behaviour, in the context of a system architecture that incorporates redundancy (because the service is critical to higher level safety-critical system). More specifically, we quantify how the system architecture is *designed* to meet service requirements, when calibrated with specified rates defined by mean time between failure (MTBF) rates as supplied by manufacturers and system designers, and quantify how the system architecture *actually* meets service requirements, when the models are calibrated with rates derived from historical, field data.

Our objective is to predict future behaviour and risk from a system state in which faults have occurred, and thus inform operational decision making by those who have responsibility for the system, as it is running. A novelty of this work is that we apply formal modelling and analysis techniques in the context of a system that continually senses and monitors failures and has been running for some time. Thus we have access to actual field data about failures.

The models and temporal properties are probabilistic. Property analysis is performed: to *validate* the models, to compare model results with known historical results, and to study *predictive* properties about likely future behaviours from a variety of possible systems states. A typical predictive temporal property is, from a degraded configuration (i.e. one where a given fault or faults have occurred), compute the probability over the next 48 hours of the system reaching a no-service state. Knowledge of how that probability varies over time could help us quantify the risk to the system posed by that fault and the urgency of repair, and contribute to answering questions such as "do I need to fix the fault now, in the next 4 hours, or can I wait until tomorrow"? For example, if the probability of no service is well below the safety threshold during the next 4 hours, but thereafter rises exponentially above the threshold, we would likely conclude that a repair need not be immediate, but must be completed within the next 4 hours. There may be other parameters to minimise/maximise. For example, in some contexts a repair that is guaranteed to be completed within 24 hours may be relatively low cost, whereas a repair guaranteed to be completed within 4 hours may have a much higher cost. If we can demonstrate that safety requirements will be satisfied over the next 24 hours, then the lower cost repair may be an option. Longer term, the model(s) and analysis can be used for experimentation with different monitoring practices, different system architectures and degrees of autonomy, and, ultimately, for on-line run-time monitoring.

Our approach is *event-based*, reflecting the design of the monitoring system and the events that are sensed, monitored and logged. A key decision is how to model the passage of time in the system. We assign *rates* to events, thus our models are continuous time Markov chains (CTMCs): the state space is

discrete but time is continuous. We instantiate the models with rates derived from safety and business cases, with rates derived from actual historical field data, and with hypothetical values that reflect possible changes to business or other technical processes. By adopting CTMCs as our underlying semantics, we can relate our models to 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, i.e. $1 - e^{r \cdot t}$.

We reason about *temporal* properties using *model checking* techniques. We advocate temporal properties and model checking because together they allow us to quantify *all* possible future behaviours, whereas simulation only ever considers one behaviour (or trace) at a time. Typical temporal properties concerning service level behaviour include:

- from a given degraded configuration, what is the probability that the system will reach a state in which no service is offered within the next *n* hours,
- from a given degraded configuration, what is the expected time to repair a given type of fault,
- from a given configuration, what is the expected number of alarms over the next *n* hours,
- in the long run, what is the probability that the system is in a no-service state.

The contributions of the paper include:

- an event based, parameterised, counter abstraction model of a deployed communications link monitoring system,
- steady state properties to validate model against expected and historical behaviour,
- transient properties to quantify criticality of reduced redundancy configurations and to distinguish between different sets of *degraded* configurations,
- example decision making based on transient property results,
- inference of rates from actual (historical) field data,
- example analysis for an instance of the model with real case study data.

The next section contains an overview of the communications link monitoring system and in Section III we review basic concepts and definition of CTMCs and continuous stochastic logic (CSL). Section IV contains an overview of the model, which is parameterised by rates and the topology of the system. Section V gives an overview of inference of rates from historical, field data for the case study. Section VI defines the propositions used in the example analysis. Sections VII and VIII define the properties used in steady state and transient analysis, and example results for the case study, respectively. In section IX we give an example of how transient property analysis results can inform decision making. Section X contains a brief overview of the implementation. In section XI we give an overview of the entire modelling and analysis approach, indicating how and when it can be used, and in Section XII we reflect on aspects of the case study. Related work is discussed in Section XIII, and we conclude in Section XIV.

For reasons of commercial sensitivity, details of the company and the application must remain private.

II. Overview of the communications links monitoring system

The primary components of the system are *sectors*, *sites*, *frequencies* and *channels*. There are 35 sectors, each of which is allocated a fixed set of frequencies, plus an emergency frequency. There are 17 sites, each with several antennas that send (Tx) and receive (Rx) on different frequencies. Here, we refer to antennas as *channels*. 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 idle backup channels. We refer to the main channel as the A channel and the backup channel as the B channel.

Sites are monitored for power line status, communication link status, and there are sensors for intrusion and flooding.

The company runs a software monitoring system in real-time that senses component status and uses a colour coding to indicate the status of components:

- green indicates functioning or serviceable
- red indicates faulty, raise an alarm
- blue indicates under maintenance
- amber indicates reduced redundancy and possibly not fully functioning (for example when one antenna goes down for a frequency).

An alarm goes off every time a component turns red. As we shall reveal later, alarms typically cascade. One motivation of this work is to help engineers develop a better understanding of the service-level impact of these alarms.

III. Technical background

A. Continuous Time Markov Chains

Following [1], 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 outoing transitions from s exits. 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).

In the remainder of this paper we use an informal, graphical notation for indicating the states and transitions of a CTMC, for example, see Figures 1 and 2.

B. Continuous Stochastic Logic

We use Continuous Stochastic Logic (CSL) [2], a stochastic extension of the Computational Tree Logic (CTL) allowing one to express a probability measure of the satisfaction of a temporal property in either transient or in steady-state behaviours. The formulae of CSL are state formulae Φ with path formulae Ψ:

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

where a ranges over a set of atomic propositions AP, $\bowtie \in \{\leq, <, \geq, >\}$, $p \in [0,1]$, and I is an interval of $\mathbb{R}_{\geq 0}$. We additionally use the path operator (syntactic sugar): the *eventually* operator $\mathbf{F}$ (future) where $\mathbf{F}^{\mathrm{I}}\,\Phi \equiv \mathit{true}\,\mathbf{U}^{\mathrm{I}}\,\Phi$.

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 which satisfies Ψ meets the bound $\bowtie p$.

We employ the PRISM probabilistic model checker [3], which allows one to leave the bound $\bowtie p$ unspecified, in which case a 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.

C. Language of reactive modules

There are several languages or formalisms for specifying Markov processes based on rate transition matrix descriptions, state-transition graphs, or guarded command languages such as Reactive Modules [4], which is the basis of the PRISM language. Processes are represented by modules consisting of action-labelled guarded commands (transitions) and are composed using the multiway synchronisation operator over all common actions. Each module has the form: local variable declarations followed by a non-deterministic choice between transitions, each which indicates the value of variables in the next state and may be labelled by an action. A simple transition has the form: $[action]guard \rightarrow rate : update$ meaning the module makes a transition to a state described by the *update* at the given *rate* when the *guard* is true. The $action$ label is optional, [] indicates simple non-deterministic choice. In an update, if x is a variable, then x' denotes the value of x in the next state. An update may be a choice between two or more assignments, for example $[action]guard \rightarrow rate1 : update1 + rate2 : update2$, meaning there is a race condition between the two updates. The rate of the synchronised transition is the product of all the individual rates.

IV. Overview of the model

A **channel** is characterised by three parameters: whether it is receiver (Rx) or transmitter (Tx), the frequency, and the site reference. There are four possible states for a receiver/transmitter: S for serviceable (green), F for faulty (red), M for under maintenance (blue) and E for external site failure (red). Note there is no reduced redundancy in a single channel (i.e. there is no amber).

We do not represent channels individually, but employ a *counter abstraction* in which a **pair of A and B channels** is represented by a single module: the state labels indicate the *counts* of the constituent channels. For example, state SS means that both A and B channels are serviceable, state SF means that one channel is serviceable and the other is faulty. States are colour coded according to three classes: green for serviceable, amber for reduced redundancy, and red for no service. The state transition diagram for a pair of channels is given Figure 2. For the purposes of reasoning about service availability, a component that is under maintenance is considered not serviceable, as indicated in Figure 2.

The **external environment** of a site is characterised by major physical events that cause a failure of the site (e.g. instrusion, powerline and backup generator failure, flooding) or minor events (e.g. powerline failure but backup generators functioning) that mean the site is more likely to fail, but is still functioning. Thus it has three states: E0 for serviceable (green), E1 site event (amber), and E2 site failure (red).

A **site** is represented by the concurrent composition of three modules: the transmitters, the receivers, and the site environment. Thus a state is a triple consisting of the channel pairs and site environment. States are classified by three colours: W for serviceable site (green), R for reduced redundancy site (amber), and N for no service site (red). 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; that is an external site event causes the channel to move to state E. Similarly, the (site environment) transition between E2 and E0 synchronises with a channel (reset) transition to serviceable.

Figure 1 contains state transition diagrams for a channel pair and site environment, indicating synchronisation between the two and symbolic rates.

An $n\text{-}ary$ **sector** is represented by its n constituent sites. Without loss of generality we assume here a sector with three sites. Again, states are labelled and colour classified: W W W for serviceable sector (green), N N N for no service sector (red), and amber for reduced redundancy sector, which consists of all remaining states, i.e. the language defined by $L \setminus (\{\text{W W W}\} \cup \{\text{N N N}\})$, where L = (W|N|R) (W|N|R) (W|N|R). Note that the notation for a channel pair assumes symmetry; for example, we do not distinguish SF from FS and so the latter is not a state. On the other hand, the notations for a site and a ternary sector *do not* assume symmetry; for example, site SF SS E0 is distinguished from SS SF E0. The former denotes a configuration where the transmitter is reduced redundancy and the receiver is serviceable, whereas the latter denotes a configuration where the transmitter is serviceable and the receiver is reduced redundancy. However, we note that the rates for events for transmitters and receivers are identical and if either the transmitter *or* receiver is no service, then the entire site is no service. Also, note that the rates for events will usually differ from site to site; for example, for a given sector, the rate for an event *e* for the first site may be different from the rate for the same event *e* for the second site (in that sector).

Not all configurations are reachable. For example, the site configuration SS E 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.

For a given frequency, a sector is represented by the concurrent composition of its constituent sites, which typically varies between 2 and 5 sites. Note that sites within a sector are independent (i.e. no synchronisation).

Table I contains a summary of the (labels of the) states

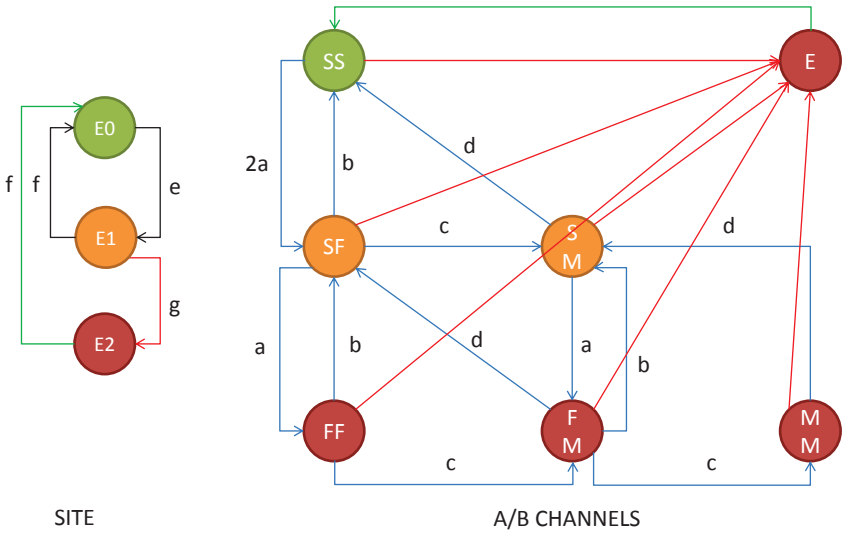

Fig. 1: State transition systems for site environment and channel pair. Synchronisation on red and green transitions. Rates: a = channel failure, b = channel quick repair, c = channel slow repair, d = channel under maintenance repair, e = site event, f = site repair, g = site failure.

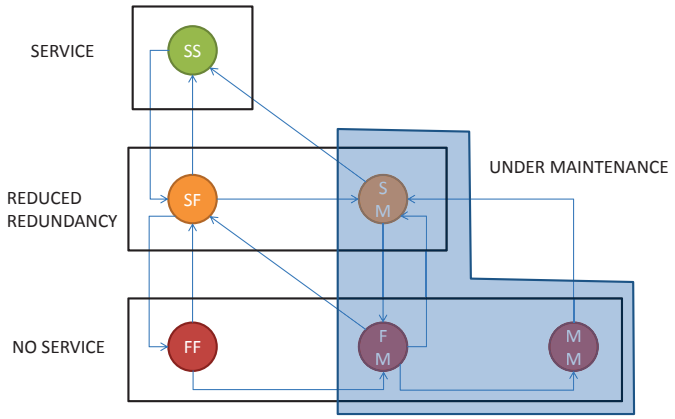

Fig. 2: Levels of service.

that are represented in a model with a ternary sector, using regular expression notation, e.g. '|' for disjunction and '*' for wildcard. Strictly speaking, the labels of states in a CTMC are the propositions that are true in that state. Here, we introduce a convenient labelling for states that indicates the properties of that state.

As example, Figure 3 gives the PRISM module for a generic transmitter (Tx) for channel with site reference X and the Site environment, in the context of rate declarations. Note that the last two transition choices in the transmitter are labelled by actions that cause the transitions to synchronise with the site environment. Further, we include $alarm$ labels for transitions to red states, so that we can use PRISM *rewards* to count the number of alarms, if required.

A. Rates

The model is governed by seven rates $a, \ldots, g$, as indicated in the graphical representation of the modules given in Figure 1 and the PRISM code snippet in Figure 3. Note there are two transitions from the repairing state: a *quick*, local repair that returns to the serviceable state, and a *slower* transition to the under maintenance state. The former reflects an error that can usually be fixed by a remote reboot. The latter reflects the fact it may take some time for an engineer to physically

component	*colour*	*states*	*description*
channel	green	S	serviceable channel
	blue	M	under maintenance channel
	red	F	faulty channel
	red	E	site failure
channel	green	SS	serviceable AB
pair	amber	SF|SM	reduced redundancy AB
(A,B)	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|MF) * *	N no-service site
	red	* (FF|FM|MF) *	N no-service site
ternary sector	green	W W W	serviceable sector
(site,site,site)	amber	all other combinations	reduced redundancy sector
	red	N N N	no-service sector

TABLE I: State labelling and colour coding in counter abstraction model.

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 diagram). Since state SS contains two channels that can individually and independently fail, the rate for transition SS $\rightarrow$ 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 employed to repair a single channel and a pair of channels is the same, we use b instead of $2b$ as the rate for transition FF $\rightarrow$ 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 1:3 ratio between quick and slow repairs, c is defined as $b/3$.

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 external site events and site failures, respectively, and g is the rate of (external) site repair.

To quantify how the system architecture is *designed* to meets service requirements, we instantiated the rates with derived from typical MTBF values, interviews with engineers, and inspection of the business case; to quantify how the system architecture *actually* meets service requirements, we instantiated the rates according to historical, field data. Since the latter is a novel and challenging aspect of our approach, we discuss how the rates are derived in the next section and use those rates in the analysis in sections VII to IX.

V. Inferring rates from field data

After our initial development of the (parameterised) model and investigation of how the architecture is designed to meet

```
//Ratio quick repairs to slow repairs
const double x = 3;
//  Channels - mean times in hours
const double failure = ..
const double repair = ..
const double qrepair = ..
// Site environment
const double event = ..
const double site_failure = ..
const double fix_event = ..
//Rates
const double a = 1/failure;
const double b = 1/qrepair;
const double c = b/x;
const double d = 1/repair;
const double e = 1/event;
const double f = 1/fix_event;
const double g = 1/site_failure;

module Site_X_Tx  // channel pair
s0_X : [0..6];
[alarm_0_X]s0_X=0 -> 2*a:(s0_X'=1);   //SS
[alarm_0_X]s0_X=1 -> a:(s0_X'=2);     //SF
[]s0_X=1 -> b:(s0_X'=0) + c:(s0_X'=3); //SF
[]s0_X=2 -> b:(s0_XL'=1)  + c:(s0_X'=4); //FF
[alarm_0_X]s0_X=3 -> a:(s0_XL'=4);    //SM
[]s0_X=3 -> d:(s0_X'=0);  //SM
[]s0_X=4->b:(s0_X'=3)+c:(s0_X'=5)+d:(s0_X'=1);//FM
[]s0_X=5 -> d:(s0_X'=3); //MM
[alarm_f_X]true -> 1:(s0_X'=6); // E
[fix_X]true -> 1:(s0_X'=0); // E
endmodule

module Site_env_X
env_X : [0..2];
[alarm_e_X] env_X=0 -> e:(env_'=1);
[alarm_f_X] env_X=1 -> g:(env_'=2);
[fix_X] env_X=1 -> f:(env_'=0);
[fix_X] env_X=2 -> f:(env_'=0);
endmodule
```

Fig. 3: Generic Tx Site module and Site environment module.

service requirements, the company gave us access to their SAP incident ticketing system, which they employ for long term storage of data concerning logged failures. The data logs record failure occurrences and repair durations, as well as a textual description, which allow us to categorise events.

In the case study, inference of rates was by manual inspection of historical field data for failures, sector by sector, for nominated time periods. We note that longer term, we aim to influence the design of readouts and tickets, and subsequently we will automate the inference process.

As an example, we give results for a nominated sector, which we call *X*, over a one year period: February 2012 to February 2013. The data included 61 alarms and 24 site events. From this data we calculated inter failure times and then 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

rate	*inferred value*
Mean inter-failure time	452 h
Mean repair time	18 h
Response	57 m
Site event	1107 h
Percentage of quick repairs	15
Site failure	1 every 11.33 years

TABLE II: Inferred rates from historical.

rates. The final results are reported in Table II.

Our study of the field data for the nominated sectors confirms 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. Moreover, the inferred rates are of the expected orders of magnitude (again, as indicated by interviews with engineers).

However, our analysis of the field data raised some issues that require further consideration.

First, some events were impossible to detect 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. Second, rare events such as site failures did not occur in the time span covered by the data set; we had to look at 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 the contemporaneous failure of both the A and B channels, and
- deterministic events such as scheduled maintenance.

We will return to these issues in Section XII, now we turn our attention to the analysis of instantiations of the model.

VI. Propositions

Fundamental to any analysis is the set of propositions that are checked in a given state. Since our primary interest is level of service, i.e. the (monitoring) colour of a configuration as defined in Table I, we define propositions accordingly. Namely, we define propositions that indicate whether a channel/site/site environment/sector is serviceable/noservice/reduced redundancy thus:

$$serviceable\_chan(c) = (c = \text{SS})$$
$$serviceable\_env(e) = (e = \text{E0})$$
$$serviceable\_site(s) = serviceable\_chan(Tx_s) \wedge serviceable\_chan(Rx_s) \wedge serviceable\_env(Env_s)$$
$$serviceable\_sector(A) = \bigwedge_{s \text{ site in } A} serviceable\_site(s)$$

$$noservice\_chan(c) = (c = \text{FF}) \vee (c = \text{FM}) \vee (c = \text{MM}) \vee (c = \text{E})$$
$$noservice\_env(e) = (e = \text{E2})$$
$$noservice\_site(s) = noservice\_chan(Tx_s) \vee noservice\_chan(Rx_s) \vee noservice\_env(Env_s)$$
$$noservice\_sector(A) = \bigwedge_{s \text{ site in } A} noservice\_site(s)$$

Colour	*Model result*	*Result from historical data*
serviceable	88.46%	86.54%
reduced redundancy	11.53%	13.56%
no service	10^{-8}	0.00%

TABLE III: Comparison of model long run behaviour and manual analysis of historical data.

$$rr\_chan(c) = (c = \text{SF}) \vee (c = \text{SM})$$
$$rr\_env(e) = (e = \text{E1})$$
$$rr\_site(s) = \neg(serviceable\_site(s) \vee noservice\_site(s))$$
$$rr\_sector(A) = \bigvee_{s \text{ site in } A} rr\_site(s)$$

We also compute expected *rewards*, associated with states or transitions. For example, we use rewards to compute the expected number of alarms, over a period of time.

VII. Steady state properties

Steady state properties express *long run* behaviour and we use these properties for *validation* of the model. Typically we examine steady state behaviour for a given sector, computing the probability to be in a *service* state, a *reduced redundancy* state, or a *no service* state, in the long run. For sector *X*, the respective PRISM steady state properties are: $\mathcal{S}_{=?}[(rr\_sector(A))]$, $\mathcal{S}_{=?}[(serviceable\_sector(A))]$, and $\mathcal{S}_{=?}[(noservice\_sector(A))]$.

We checked these properties in the model for sector *X*, the results are given in the left hand column in Table III. We note that in the long run, the sector is serviceable for the majority of time (over 88%), and this accords with the experiences of the engineers we interviewed.

For further validation, we then analysed (by hand) the historical data for that sector (over one year), to calculate the percentage time spent in a *service* state, etc. These results are indicated in the right hand column in Table III. As can be seen, the model results align very well with actual performance in that sector over a one year period.

VIII. Transient properties

A. Transient properties for prediction

Transient properties are used to express the probability of reaching a state that satisfies a proposition within a period of time. In this system, we compute the probability of reaching *no service* in a given *sector* within time T, which is expressed for sector X by the property $\mathcal{P}_{=?}[\mathbf{F}^{\leq T} noservice\_sector(X)]$. We experiment in PRISM with instantiations of the variable T, to give a plot of how the probability changes over over time.

But the property alone is not sufficient to characterise criticality in a deployed system: the key question when considering transient properties is what is the *state* from which we compute the probability? In other words, we calculate probabilities of reaching certain states *from* given state(s). In standard model checking, the initial state is the pre-defined, 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

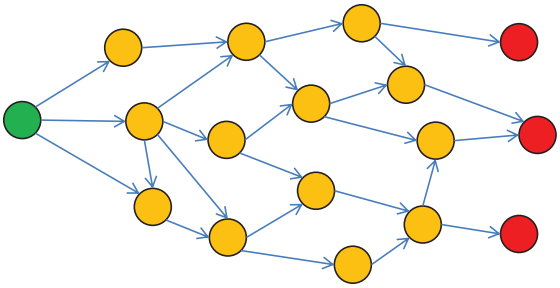

Fig. 4: Distinguishing degraded configurations: time to a no service configuration depends on the current configuration.

A	*B*	*C*
W	W	N
W	N	N
W	W	R
W	R	R
R	R	R
R	R	N
R	N	N

TABLE IV: Selected degraded configurations for sector *X*.

and the interesting cases are the degraded, amber configurations. Specifically, once we have reduced redundancy, we want to be able to quantify the criticality of the situation and take informed decisions – in other words, do I need to fix a fault now, or can I wait? And how long?

B. Degraded configurations

The amber colour coding denotes a *degraded* configuration in which, through the occurrence of a fault, there is some degree of reduced redundancy, though of course the service is still available. One goal of our analysis is to enable us to *distinguish* between different amber configurations. It is important to note that the length of the path to a no service state number 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 between 10 and 15 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 our choice of the time interval. Figure 4 gives a pictorial representation: on the left (the source state) we have the initial green state and on the right (the sink states) the red no service states, the degraded configurations states are the majority of states in between these two extremes. We aim is to distinguish subsets of these states and quantify the probability of reaching a no service state from them.

C. Results

The degraded configurations we examined for the example sector *X* with three sites *A*, *B*, and *C* are given in Table IV. We refer to Table I for the definitions of W, R and N. Observe that both N and R can be the result of many different site configurations, and we randomly select one of those for each occurrence of R and N. 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.

Figures 5, 6, and 7 give the results for the probability of reaching a no-service configuration, from different degraded configurations, over a time interval of 48 hours.

We are considering a critical service in a safety-critical domain and so we expect probabilities to be very low. However, observe the orders of magnitude difference on the Y axis. In Figure 5, the scale is $10^{-}7$, whereas in Figure 6, the scale is $10^{-}4$, and in Figure 7, the scale is $10^{-}3$. Also, observe that in Figure 5 the steepest trajectory is WWN, which contains one no service site, and in Figure 7, 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 sector topologies, we observe the contribution of additional sites increases service availability, however that increase is inversely proportional to the number of additional sites. The example in Figure 8 illustrates this: the difference between $3-$ and $4-$ary sites 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 n of sites, when $n > 3$. This implies the plots for the ternary site given in Fig. 5 to 7 can be used as a good approximation for sectors with more sites.

Finally, we remark that channel redundancy *within* a site is also 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 Figure 6, in which the plot for WRR is effectively flat, and similarly in Figure 7, in which the plots for WRN and WNN are also effectively flat.

IX. 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:

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 can 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 solid line in Figure 9. We remark that this solid line denotes the expected behaviour *if no assumptions are changed in the system*, that is, if we assume the current failure and repair rates. Now consider the shaded area in Figure 9, which indicates the probabilities above the safety threshold. The prediction shows that the system is likely to become

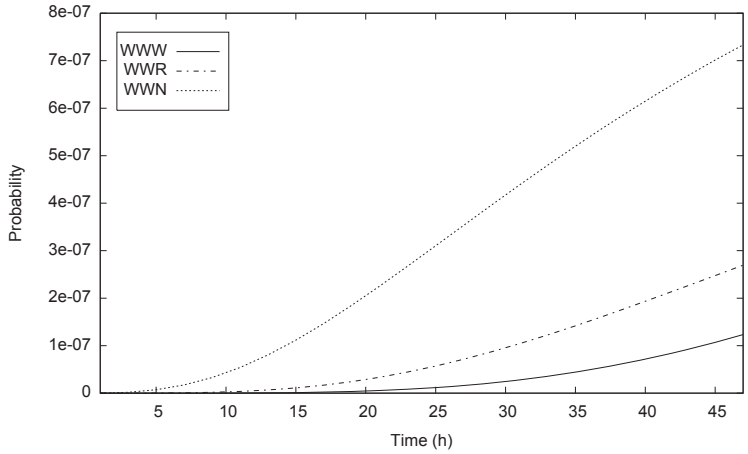

Fig. 5: Comparison of prediction of no-service from WWW, WWR and WWN configurations.

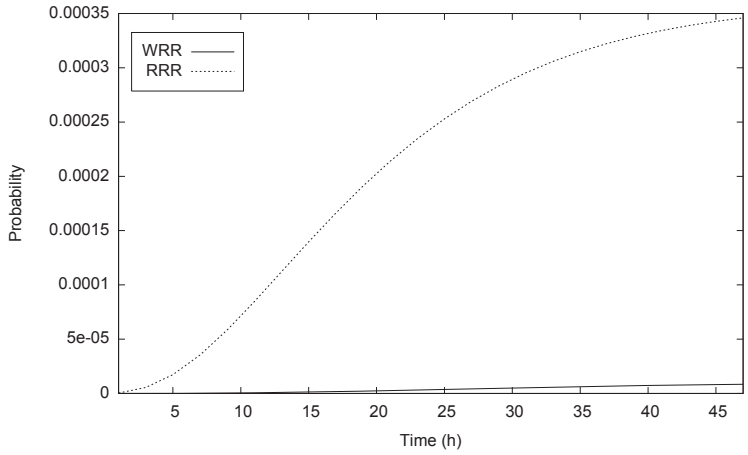

Fig. 6: Comparison of prediction of no-service from WRR and RRR configurations.

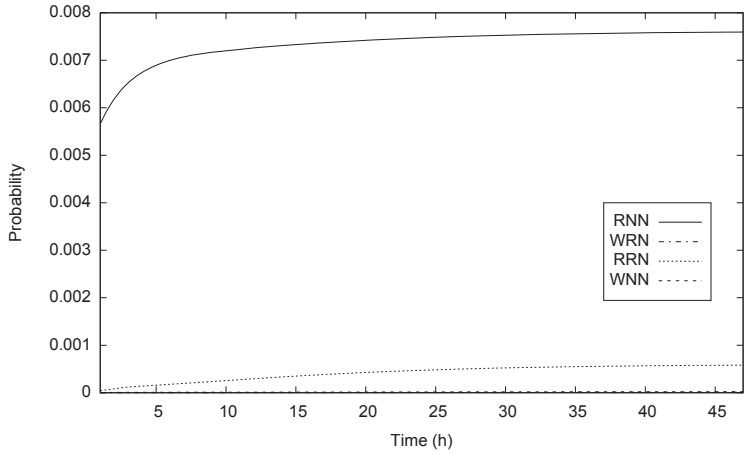

Fig. 7: Comparison of prediction of no-service from RNN, WRN, RRN and WNN configurations.

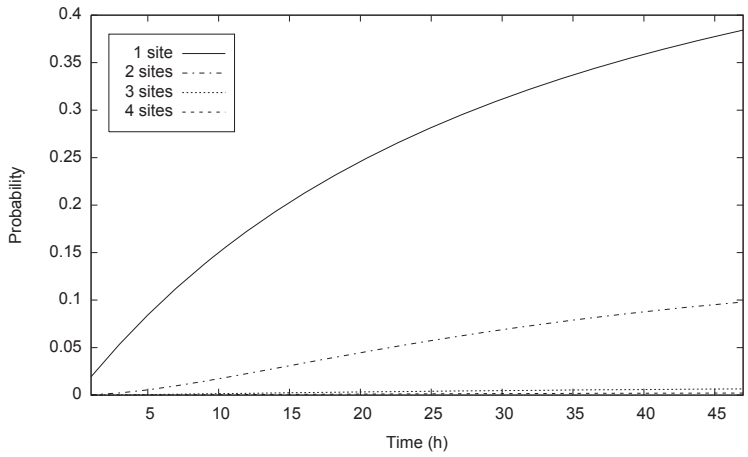

Fig. 8: Comparison of prediction of no-service for 1 to 4-ary sector topologies.

unsafe after 20 hours. We reach the conclusion that within 20 hours, we want to be on *another trajectory*, which is below the system safety threshold. We can do this by altering one or more rates so as to, in effect, transition the system into an alternative, more favourable "current" configuration, i.e. one that is less likely to lead to no service, within the given time frame.

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 behaviour 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 Figure 9 with the dashed line. Now consider the behaviour from 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 behaviour is indicated by the dotted line in Figure 9. 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 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 solid line in Figure 10. The dotted 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. Figure 11 gives a graphical illustration of our decision with reference to the state space: we make a discrete transition to another (more favourable) state.

We employ a similar approach to predict the behaviour of the system after certain 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 these cases, we are moving the trajectory *up*, instead of *down* at the the discontinuity, i.e. we are increasing likelihood of no service.

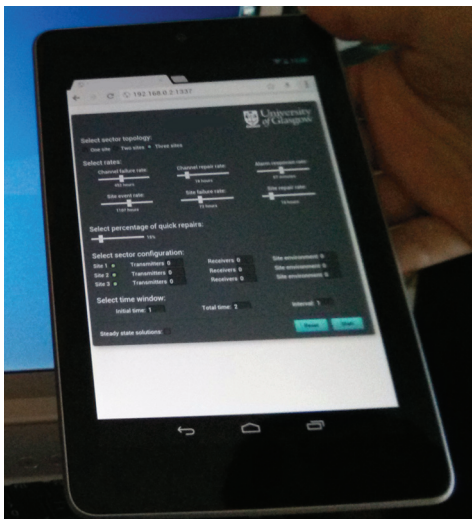

Fig. 12: Tablet GUI for setting rates and topologies.

X. Implementation and GUI

Briefly, the system is a client-server architecture based on a nodejs web server and a web interface. The models are parameterised by rates and the topologies of each sector. To make the models accessible to engineers in the company, we developed a a bespoke GUI, which runs on several web browsers and on a tablet, as illustrated in Figure 12. Default rates and topologies are altered using sliders on the interface.

XI. The modelling and analysis framework

Our overall framework is depicted in Figure 13 and summarised as follows. Model definition and analysis is indicated by solid lines, feedback from the analysis is indicated by the dashed lines.

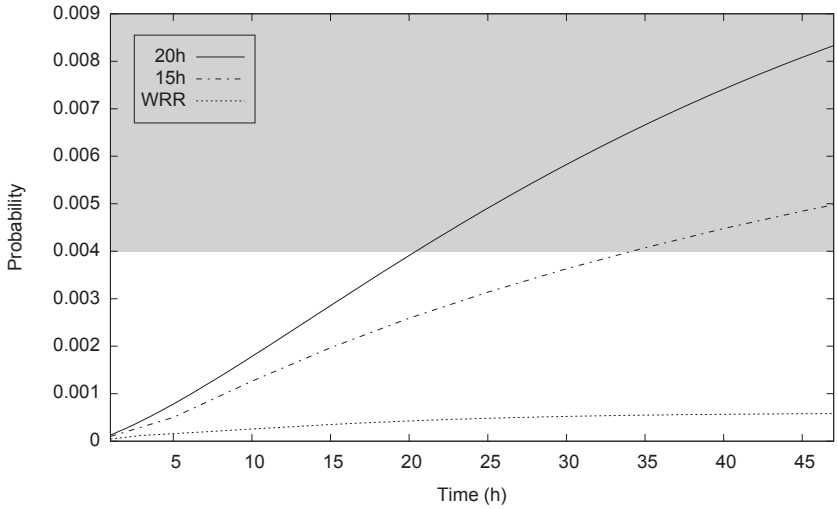

Fig. 9: Transient property for service availability and a system safety threshold.

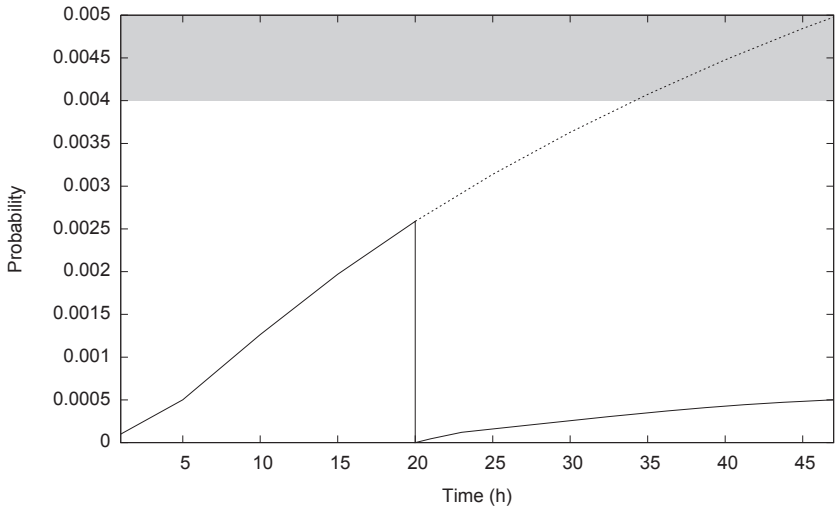

Fig. 10: Transient property for service availability before and after discrete transition to a new state, in context of 4×10^{-3} system safety threshold.

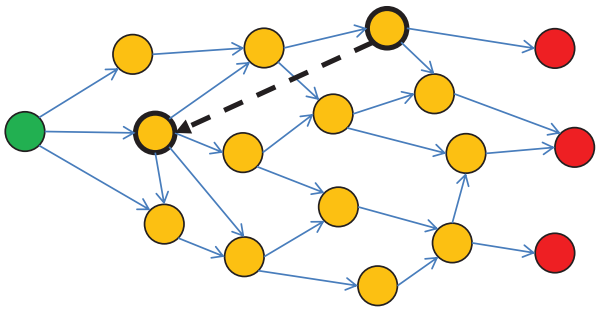

Fig. 11: Changing current state after analysis of transient property for service availability.

We model an event-based (sub) system with the Markovian property by a CTMC parameterised by event rates and system topology. The model relates component level failures to service availability. Event rates can be instantiated to investigate *system design* (e.g. from safety and business cases), or to investigate *operational behaviour* (e.g. from field data). A GUI aids parameter setting. We *validate* the model by examining the results of steady state temporal logic properties, and comparing them with the expected (or required) results from the safety and business cases, and with the observed, operational results inferred from the field data (dashed lines). This is indicated on the left hand side of Figure 13. An example result of steady state property analysis is given in the bottom left hand side of the Figure, in the bar chart format produced by our GUI and PRISM.

On the right hand side of the Figure we indicate how the model is used to inform decision making by providing *quantified predictions*, which are the result of transient temporal logic properties. A typical example is the probability of reaching *no service* within a fixed time frame. An example result of a tranisent property analysis is given in bottom right hand side of the Figure, in the graphical format produced by our GUI and PRISM.

A decision is a system intervention (dashed line), for example making a repair or scheduling maintenance, within another time frame. Recall there are numerous factors to consider when considering which new rates and configurations are plausible (e.g. cost or physical assessability) and the consequences of each, in terms of system behaviour.

In general, there are four ways in which this framework can be used:

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

In our case study, the original interest by the company was the second and the fourth options, as their primary concern was the current operational system. However, they are now showing considerable interest in the first option, for future developments. We note they have little interest in the real-time option; moreover, they have little interest in behaviours much beyond the 48 hour time frame.

We have chosen CTMCs as our underlying semantics, but there are other possibilities for discrete systems such as hybrid or probabilistic timed automata, or models such as bigraphs that incorporate spatial aspects. Regardless of the semantics, our overall approach remains the same: to analyse a component based system and consider quantify behaviour from selected degraded configurations. While we have applied our approach here to communications link status, our approach is applicable to any critical service with discrete events and failures, for example systems in process control or signalling.

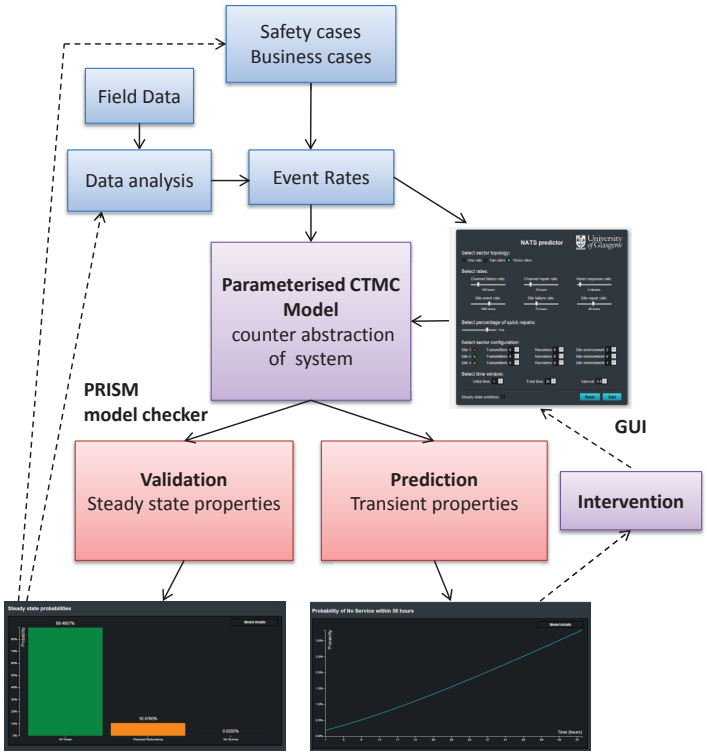

Fig. 13: Modelling and analysis process.

XII. Reflections on the Case Study

As indicated in Section V, when we inspected the historical data, we found evidence of dependencies between channel A and B faults, and evidence of scheduled maintenance. Neither of these issues had been raised with us previously, and so while on the one hand it was disappointing to uncover possible omissions in the model, on the other hand it demonstrated the value of inspecting historical data. Concerning the first, the cause of dependencies is as yet unclear, in part due to the formats for recording faults and the use of free text. Possible contributory factors are transmitters and receivers are usually commissioned at the same time (and therefore failures occur a similar times), and more likely, communications network failures that affect both channels simultaneously. Determining causes requires further investigation, however, modifications to the model are relatively straightforward and would involve the introduction of synchronised Tx and Rx failures. Concerning the second, we could model scheduled maintenance with a new event and suitable rate(s), or as deterministic, timed events (which would require enhanced model semantics). Given the data we have seen for scheduled maintenance is so far relatively sparse, we have not yet incorporated maintenance into the model, this will be further work.

Our system has huge variation in orders of magnitude of rates of events, and in likelihoods of the properties we analyse. We note though that this has not caused any numerical problems (e.g. stiffness).

XIII. Related Work

While there has been work in formal modelling for safety-critical systems, especially in the context of formal system development [5], and runtime models for managing self adaptation and the complexity of evolving software behaviour while it is executing [6], there appears to be scant work on formal modelling to inform (human) operational decision making during the execution of safety-critical systems. One issue for quantitative analysis of dependable systems development is state space explosion and numerical simulation difficulties in the presence of rare events [7]. We note we have not encountered state space explosion problems nor numerical difficulties, because our modelling approach is a 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).

If we choose in future to model scheduled maintenance by deterministic, timed events, as mentioned in Section XII, then the result will be a model with both stochastic and deterministic transitions. Such models have been considered 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 is applied to carry out quantitative analysis. This may provide a suitable semantic framework for our further work.

We note an approach with some similarity is our work for domestic network management [9] in which, as the system is running, we generate a simulation trace in real time, consisting of formal, bigraph models. These are analysed, in real time, according to various state properties, and notification of violations are fed back to the system and to human users.

XIV. Conclusions and Future Work

We have proposed that predictive modelling and probabilistic temporal logic reasoning can inform operational decision making in complex systems with component failures and monitoring, by providing temporal analysis of predicted risks. To investigate this hypothesis we developed an event based, parameterised, counter abstraction model of a deployed industrial communications link monitoring system; the model is parameterised by event rates and sector topologies. We used steady state properties to validate instances of the model against expected and historical behaviour, and transient properties to quantify criticality of reduced redundancy configurations, from different sets of *degraded* configurations. We indicated how transient property results could inform decision making, giving examples based on accepted system safety thresholds, in the context of event rates inferred from actual (historical) field data for the case study.

We have implemented the entire modelling and analysis framework in the PRISM language and model checker. The models and analysis techniques are parameterised; these are modified easily through a GUI that facilitates interaction.

While our original motivation was behaviour in an operational system, the framework also incorporates analysis of how an architecture is designed to meet service requirements, based on event rates derived from safety and business cases.

Further work includes new representations for scheduled maintenance and dependent events; analysis of further sectors and frequencies in the case study, including examination of more field data and incorporation of spatial aspects.

Acknowledgment

We thank our industrial collaborators for working with us on this interesting problem. This work was partially funded by the EPSRC grant *Verifying Interoperability Requirements in Pervasive Systems* EP/F033206/1 and the University of Glasgow EPSRC funded Impact Acceleration Account.

References

[1] 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)*, ser. LNCS (Tutorial Volume), M. Bernardo and J. Hillston, Eds., vol. 4486. Springer, 2007, pp. 220–270.

[2] 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.

[3] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, ser. LNCS, G. Gopalakrishnan and S. Qadeer, Eds., vol. 6806. Springer, 2011, pp. 585–591.

[4] R. Alur and T. A. Henzinger, "Reactive Modules," *Formal Methods in System Design*, vol. 15, no. 1, pp. 7–48, 1999.

[5] A. Galloway, F. Iwu, J. A. McDermid, and I. Toyn, "On the formal development of safety-critical software," in *VSTTE*, 2005, pp. 362–373.

[6] U. Aßmann, N. Bencomo, B. H. C. Cheng, and R. B. France, "Models@run.time (Dagstuhl Seminar 11481)," *Dagstuhl Reports*, vol. 1, no. 11, pp. 91–123, 2012. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2012/3379

[7] 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.

[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] M. Calder, A. Koliousis, M. Sevegnani, and J. Sventek, "Real-time verification of wireless home networks using bigraphs with sharing," *Science of Computer Programming*, vol. 80, Part B, pp. 288–310, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167642313001974