Improving Risk Management

RIESGOS-CM

Análisis, Gestión y Aplicaciones

P2009/ESP-1685

Technical Report 2010.02

Bayesian Reliability Analysis for Hardware/Software Systems

Javier Cano, Javier Moguerza, David Rios Insua

http://www.analisisderiesgos.org

Bayesian Reliability Analysis for Hardware/Software Systems

Javier Cano and **Javier M. Moguerza**

Rey Juan Carlos University
Department of Statistics and
Operations Research
Camino del Molino s/n
28943 Fuenlabrada (Madrid), Spain
(javier.cano@urjc.es)

David Ríos Insua

Royal Academy of Sciences
Valverde 22
28004 Madrid, Spain
(david.rios@urjc.es)

Abstract

We provide a Bayesian analysis for a class of models used to evaluate and forecast the reliability of complex hardware/software systems, described through Reliability Block Diagrams. Blocks referring to hardware components are modelled through Continuous Time Markov Chain models, whereas blocks referring to software components are modelled through a mixture of Software Reliability Growth models. Inference and forecasting tasks with such models are described and illustrated with an example.

Keywords: Reliability Block Diagram, Software Reliability Growth, Model Selection, Continuous Time Markov Chains, Multinomial - Dirichlet model, Exponential - Gamma model, Bayesian Analysis.

1 Introduction

There is a growing interest in reliability analysis of systems composed of several hardware and software (HW/SW) subsystems. This is specially important in safety critical systems which, more and more frequently, appear in organizations, see Dale and Anderson (2009), Dunn (2003) or Cukic and Chakravarthy

(2000). As an example, consider a university resource planning system which facilitates access through Internet to users (students, lecturers, administrators,...) to undertake various administrative processes (registering for a course, checking a research project status, assessing the human resources policy,...). This system would typically include an Internet server and a machine hosting the planner, which will be a program built on top of a database system. Our aim is to describe a class of models relevant in analyzing such complex systems.

Stemming from work by Goel and Soenjoto (1981), there has been interest in HW/SW systems reliability. Many papers have used general Continuous Time Markov Chains (CTMCs) models for such purpose, including Welke et al. (1995), who incorporate a Nonhomogenous Poisson Process (NHPP) software reliability model into a CTMC hardware reliability model. Pukite and Pukite (1998) and Xie et al. (2004) describe simpler models for HW/SW reliability. Recently, more sophisticated models have been developed, which take into account dependencies or dynamic or functional aspects, through stochastic Petri nets, see e.g. Lollini et al. (2009); Dynamic Reliability Block Diagrams (DRBDs), see e.g. Distefano and Puliafito (2009); or modified Reliability Block Diagrams (RBDs), see e.g. Levitin (2007).

The approach we adopt here combines several conventional models in a novel way, see Trivedi (2001) or Trivedi et al. (2006) for related proposals. Specifically, we assume that the HW/SW system may be described in terms of an RBD, see e.g. Birolini (2007). Pending from each block, one may have standard models, like those based on the exponential, or more sophisticated ones, based on mixtures of Software Reliability Growth Models (SRGM), see e.g. Singpurwalla and Wilson (1999), if it is a software component, or on a CTMC, see e.g. Ross (2007), if it is a hardware component. Sharing of parameters allows us to model some block dependencies. We provide Bayesian analysis for such models, thus taking advantage of all information available.

In Section 2, we describe a general procedure to perform reliability forecasting with RBDs, taking into account uncertainty in block parameters, and computational strategies to deal with complex systems. Section 3 describes issues referring to software blocks with the aid of a novel mixture-based SRGM selection strategy. As an illustration, we have included Power Law, Delayed S-Shaped and Schneidewind models within our mixture model. In Section 4, we show how to analyze hardware components, through CTMCs, with the aid of phase-type distributions. Section 5 provides a realistic example which sketches our university resource planning system. We end up with some discussion.

2 Reliability Forecasting with Reliability Block Diagrams

We shall assume that we may model a complex HW/SW system through an RBD, see Birolini (2007) for a full description: the system may be described with c blocks, some of them referring to HW components, the rest to SW components. Let $R_1, R_2, \ldots, R_c$ be the corresponding block reliabilities. The underlying reliability model for block i will depend on parameters θ_i, so that we shall actually have reliabilities conditional on parameters θ_i, $R_i|\theta_i,\ i = 1, 2, \ldots, c$. To simplify the notation, we shall use $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_c)$ when convenient. Some of the θ_i parameters might be shared between several blocks.

We assume we have a computational procedure f that allows us to compute the system reliability $R|\boldsymbol{\theta}$, conditional on $\boldsymbol{\theta}$, from the component reliabilities, that is,

$$R|\boldsymbol{\theta} = f\left(R_1|\theta_1, R_2|\theta_2, \ldots, R_c|\theta_c\right). \tag{1}$$

General results to approximate the system reliability from block reliabilities may be seen in Ball (1995), including state-based and path and cut-based methods. For simpler systems with only parallel and series connections, see Monga and Zuo (2001).

Example. The following diagram, shown in Figure 1, provides a sketch of the Rey Juan Carlos University resource planning system, composed of an Internet server, connected in series with an Enterprise Resource Planner (ERP), which comprises the ERP infrastructure (typically a load balancer together with several servers), and the ERP software (typically with several modules) which is built on top of a database system.

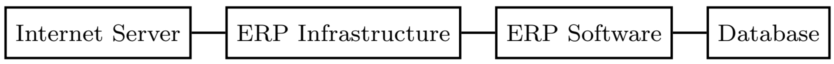


Figure 1: RBD example

In this case, dropping parameter dependence, we have that:

$$R = f\left(R_{is}, R_{erp}, R_{sw}, R_{db}\right) = R_{is} \cdot R_{erp} \cdot R_{sw} \cdot R_{db},$$

where $R_{is}, R_{erp}, R_{sw}, R_{db}$ are, respectively, the reliabilities of the Internet server, the ERP infrastructure, the ERP software module and the database system. $\triangle$

The standard approach of estimating reliability parameters, typically through MLEs, plugging the estimates into the model and using it for reliability prediction, will usually underestimate uncertainty in predictions, since the uncertainty in model parameters is not taken into account, see Glynn (1986), Draper (1995) or Berger and Ríos Insua (1998).

Alternatively, once we have computed the posterior distribution $\pi\left(\boldsymbol{\theta}|data\right)$ of the RBD parameters, we could compute the predictive system reliability through

$$\begin{aligned} R|data &= \int R|\boldsymbol{\theta}\ \pi\left(\boldsymbol{\theta}|data\right) \mathrm{d}\boldsymbol{\theta} \\ &= \int f\left(R_1|\theta_1, R_2|\theta_2, \ldots, R_c|\theta_c\right) \pi\left(\boldsymbol{\theta}|data\right) \mathrm{d}\boldsymbol{\theta}. \end{aligned} \tag{2}$$

To do the required computations, we may appeal to several procedures, depending on the precision of the involved posteriors and the complexity of the system:

1. When there is little (posterior) uncertainty about the parameters, we just need to obtain estimates $\hat{\boldsymbol{\theta}}$, say their posterior modes, and plug them into (1) to approximate the predictive system reliability through $\hat{R} = f(R_1|\hat{\theta}_1, \ldots, R_c|\hat{\theta}_c)$. Note, however, that we need to check how the uncertainty propagates over the system reliability. One possible approach would consider the extremes of Highest Posterior Density (HPD) regions for $\boldsymbol{\theta}$, would compute the corresponding reliabilities at such extremes and, if little differences are appreciated, use $\hat{R}$ as an estimate of the predictive system reliability.

2. When there is big uncertainty about $\boldsymbol{\theta}$, as reflected in the previous procedure, in general, given the difficulties in computing (2), we shall need to use Monte Carlo simulation to approximate it through

$$R|data \simeq \frac{1}{N}\sum_{\eta=1}^{N} f\left(R_1|\theta_1^\eta, R_2|\theta_2^\eta, \ldots, R_c|\theta_c^\eta\right),$$

   where $\left\{\boldsymbol{\theta}^\eta = \left(\theta_1^\eta, \theta_2^\eta, \ldots, \theta_c^\eta\right)\right\}_{\eta=1}^{N}$ is a sample from $\pi\left(\boldsymbol{\theta}|data\right)$. We would complete the study with standard Monte Carlo based uncertainty assessments about our estimates, see e.g. Schmeiser (1990).

3. Note, though, that the evaluation of the integrand in (2) may be too expensive computationally, specially as the number of blocks and the complexity of the system increases. In this case, we may opt for using a Reduced Order Model (ROM), see Grigoriu (2009) for details. To do so, we approximate the posterior $\pi(\boldsymbol{\theta}|data)$ by an appropriate simple distribution $\{\boldsymbol{\theta}^k, q^k\}_{k=1}^{\kappa}$. The size κ of the ROM is essentially defined by the computational budget available which bounds the maximum number of evaluations of the integrand in (2). We proceed as follows:

   (a) Obtain a sample $\left\{\boldsymbol{\theta}^\eta\right\}_{\eta=1}^{N}$ from $\pi(\boldsymbol{\theta}|data)$.

   (b) Cluster $\left\{\boldsymbol{\theta}^\eta\right\}_{\eta=1}^{N}$ in κ clusters and spread the centroids to obtain the ROM range $\left\{\boldsymbol{\theta}^k\right\}_{k=1}^{\kappa}$.

   (c) Compute optimal ROM probabilities by solving the system

$$\begin{aligned} \min_{q^1,\ldots,q^\kappa} \quad & e(q^1, \ldots, q^\kappa) \\ \text{s.t.} \quad & \sum_{k=1}^{\kappa} q^k = 1, \\ & q^k \geq 0, \qquad k = 1, \ldots, \kappa, \end{aligned}$$

where e is a distance between the ROM and the posterior distribution, typically trying to approximate their moments and their distribution functions.

(d) Approximate the posterior predictive probability through

$$\sum_{k=1}^{\kappa} q^k f(R_1|\theta_1^k, \ldots, R_c|\theta_c^k).$$

This describes our general computational strategy. Some of the blocks could have pending standard reliability models, such as those based on the exponential or Weibull distributions, see e.g. Zacks (1992). In complex HW/SW systems, it is likely that we shall need to employ more sophisticated models, as we describe below.

3 Dealing with software blocks

As we have mentioned, some of our blocks will be associated with software components. There is a very abundant literature on software reliability, see Singpurwalla and Wilson (1999) or Kuo (2005) for reviews. In particular, there is a plethora of SRGMs, which are specially relevant in our context, as we deal with not necessarily mature software. An important and inherent issue in software reliability modelling is how to choose among the many SRGM available. Kuo (2005) describes several approaches, based e.g. on the standardized deviation, the prequential conditional predictive ordinate or the predictive likelihood for the partial block.

We rely here on an alternative model selection strategy for SRGM, different to that used in Kuo (2005) based on superposition of SRGMs. Ours uses a mixture approach. In it, mostly as an illustration, we include three models that we have found relevant in the cases we have analyzed: Power Law (PL), Delayed S-Shaped (DSS) and Schneidewind (1975) (SCH).

Here, we outline computations with Schneidewind's model. Bayesian analyses of the other models may be seen in Guida et al. (1989), Ruggeri (2006), Kuo et al. (1997) and Kuo (2005), among others. We then describe model selection computations.

3.1 Description of models

All the models used here assume that software failures follow a NHPP, with mean function $m(t)$ and failure rate $\lambda(t) = m'(t)$, see Ruggeri (2006) for further details. The three models included have the following description and interpretation, summarized in Table 1, where $R(t|a, b) = 1 - \Pr\{T \leq t|a, b\}$ is the reliability for time t, i.e., the probability that the system will remain ON for a time longer than t, provided it is ON at time 0, and T represents the time passed until system failure, i.e., until a new software error is detected, assuming the parameter values are a, b.

Table 1: Description of models

	PL	DSS	SCH
$m(t)$	at^b	$a\left[1-(1+bt)\,e^{-bt}\right]$	$\frac{a}{b}\left(1-e^{-bt}\right)$
$\lambda(t)$	abt^{b-1}	ab^2te^{-bt}	ae^{-bt}
a	Exp. failures at $t=1$	Exp. failures	Failure rate at $t=0$
b	Growth/decay rate ($b \lesseqgtr 1$)	Error detection rate	Relative failure rate
$R\,(t\vert a,b)$	e^{-at^b}	$e^{-a\left[1-(1+bt)e^{-bt}\right]}$	$e^{-\frac{a}{b}\left(1-e^{-bt}\right)}$

With our choice, we provide great versatility to our analysis, as we have chosen models belonging to different categories of SRGM: DSS and SCH belong to the NHPP-I type, whereas PL belongs to the NHPP-II type, see Kuo (2005). They do not overlap in their features and they can reflect different software maturity degrees. For instance, NHPP-II models consider the possibility of introducing new faults during debugging (in this case, we have $m\,(t) \to +\infty$ as $t \to \infty$), whereas NHPP-I ones do not do so (and, in consequence, $m\,(t) < +\infty$ as $t \to \infty$). This also has, as further consequence, that the reliability of NHPP-I models does not tend to zero as $t \to \infty$. Note that the Power Law model includes the case in which the failure rate is constant ($b = 1$), typical of mature software.

3.2 Computing the posteriors

We describe now how to compute the posterior distribution for the SCH model. Assume we test until we observe n failures and we observe $\{t_1, t_2, \ldots, t_n\}$ as times between failures: the first failure occurs at time $s_1 = t_1$; the second one occurs at time $s_2 = t_1 + t_2$; and so on, denoting $s_n = t_1 + t_2 + \ldots + t_n$.

If we assume gamma prior distributions for a and b

$$a \sim \mathcal{G}(\alpha_1, \beta_1), \qquad b \sim \mathcal{G}(\alpha_2, \beta_2),$$

we get, after several computations,

$$\pi\,(a, b|data) \propto a^{\alpha_1+n-1} e^{-\beta_1 a}\; b^{\alpha_2-1} e^{-b\left(\beta_2+\sum_{i=1}^{n} s_i\right)} \times e^{-\frac{a}{b}\left(1-e^{-bs_n}\right)}.$$

Observe now that:

$$\pi\,(a|b, data) \propto a^{\alpha_1+n-1} e^{-a\left[\beta_1+\frac{1}{b}\left(1-e^{-bs_n}\right)\right]},$$

which we identify as a gamma distribution with parameters $\alpha_1 + n$ and $\beta_1 + \frac{1}{b}\left(1 - e^{-bs_n}\right)$. Similarly,

$$\pi\,(b|a, data) \propto b^{\alpha_2-1} e^{-b\left(\beta_2+\sum_{i=1}^{n} s_i\right)} e^{-\frac{a}{b}\left(1-e^{-bs_n}\right)},$$

which cannot be identified as a standard distribution. To sample from this conditional posterior, we may use a Metropolis step as follows,

loop

Sample $b_{cand} \sim \mathcal{N}\left(b^{(i)}, \sigma^2\right)$.

Make $b^{(i+1)} \leftarrow b_{cand}$ with probability

$$\begin{aligned} p &= \min\left\{1, \left(\frac{b_{cand}}{b^i}\right)^{(\alpha_2-1)} e^{-\left(b_{cand}-b^i\right)\left(\beta_2+\sum_{i=1}^n s_i\right)} \times \right. \\ &\quad \left. \times\, e^{-\frac{a}{b_{cand}}\left(1-e^{-b_{cand}s_n}\right)+\frac{a}{b^i}\left(1-e^{-b^i s_n}\right)}\right\}. \end{aligned}$$

Otherwise, set $b^{(i+1)} \leftarrow b^{(i)}$.

σ^2 is chosen as a suitable value of the standard deviation, meaning that the acceptance rate of the Metropolis step is between 20% and 50%, see Gamerman and Lopes (2006) for details. A hybrid MCMC algorithm may then be defined, with Gibbs steps when sampling from $\pi(a|b, data)$ and Metropolis steps when sampling from $\pi(b|a, data)$, with convergence following arguments in French and Ríos Insua (2000).

Similar algorithms may be defined for PL and DSS models, respectively, and may be seen in Kuo et al. (1997) and Kuo (2005).

3.3 Model selection

As we have mentioned, there are many SRGMs in the literature. Because of the inherent model selection problem when dealing with SRGMs, we shall use a Bayesian model selection strategy within a mixture model to deal with reliability of software blocks. Specifically, we shall assume that with probability γ_1, the interfailure data come from a PL model; with probability γ_2, they come from a DSS model; finally, with probability γ_3, they come from an SCH model. We will then compute the posterior probability of each model as well as the posterior predictive reliability.

To do so, let $M = 1, 2, 3$ designate, respectively, the PL, DSS and SCH models. Then, the posterior probability of the i-th model is

$$\gamma_i^{post} = \Pr\{M = i|data\} = \frac{\gamma_i \pi(data|M = i)}{\sum_{j=1}^3 \gamma_j \pi(data|M = j)},$$

which can be used for model selection or averaging: one selects the model with the largest posterior probability, or performs forecasting by averaging across models based on their posterior probabilities. We have that

$$\pi(data|M = i) = \iint \pi(data|a_i, b_i, M = i) \times \pi(a_i, b_i)\, \mathrm{d}a_i\, \mathrm{d}b_i.$$

If we assume gamma priors for the parameters in the i-th model, that is, $\pi(a_i, b_i) = \mathcal{G}(\alpha_1^i, \beta_1^i)\mathcal{G}(\alpha_2^i, \beta_2^i)$ we may see that, for example, for the SCH model, denoting $(a, b) \equiv (a_3, b_3)$, $(\alpha_1, \beta_1) \equiv (\alpha_1^3, \beta_1^3)$ and $(\alpha_2, \beta_2) \equiv (\alpha_2^3, \beta_2^3)$, and after integrating out a,

$$\begin{aligned}\pi(data|M=3) &= K \int h(b) \frac{(\beta_2 + \sum_{i=1}^n s_i)^{\alpha_2}}{\Gamma(\alpha_2)} \times \\ &\quad \times b^{\alpha_2 - 1} e^{-b\left(\beta_2 + \sum_{i=1}^n s_i\right)} \, \mathrm{d}b,\end{aligned}$$

where

$$h(b) = \left[\beta_1 + (1 - e^{-bs_n})/b\right]^{-(n+\alpha_1)}$$

and

$$K = \frac{\beta_1^{\alpha_1}\Gamma(n+\alpha_1)}{\Gamma(\alpha_1)} \left(\frac{\beta_2}{\beta_2 + \sum_{i=1}^n s_i}\right)^{\alpha_2}.$$

This can be approximated by Monte Carlo simulation through

$$\hat{\pi}(data|M=3) \simeq \frac{K}{N} \sum_{\eta=1}^{N} h(b_\eta),$$

where $\{b_\eta\}_{\eta=1}^N$ is a sample from a $\mathcal{G}(\alpha_2, \beta_2 + \sum_{i=1}^n s_i)$ distribution. In a similar way, we may compute the marginal likelihood for the PL and DSS models.

Note now that, for fixed parameters γ, a, b, we may compute the reliability of the software block as

$$R(t|\gamma, a, b) = \sum_{i=1}^{3} \gamma_i R_i(t|a_i, b_i),$$

with $R_i(t|a_i, b_i)$ as in Table 1. Then, its unconditional posterior reliability would be

$$R(t|data) = \iiint R(t|\gamma, a, b) \times \pi(\gamma, a, b|data) \, \mathrm{d}\gamma \, \mathrm{d}a \, \mathrm{d}b,$$

typically computed through simulation

$$R(t|data) \simeq \frac{1}{N} \sum_{\eta=1}^{N} \sum_{i=1}^{3} \gamma_i^{post} R_i(t|a_i^\eta, b_i^\eta), \tag{3}$$

for posterior samples $\{a_i^\eta\}_{\eta=1}^N$ and $\{b_i^\eta\}_{\eta=1}^N$, $i = 1, 2, 3$.

4 Dealing with hardware blocks

We are concerned now with hardware components, which we assume can be modelled with CTMCs, with m possible states, the first l corresponding to ON states, the remaining corresponding to OFF configurations. We denote by X_t

the state the system is in at time t. The behavior of the CTMC is characterized by the permanence rates $\boldsymbol{\nu} = (\nu_1, \ldots, \nu_m)$, where $1/\nu_i$ are the means of the exponential random variables representing the time spent by the system at state i before leaving it, and the $m \times m$ transition probability matrix, $P = (p_{ij})$, where p_{ij} is the probability that, given that there is a transition out of state i at time t, it leads to state j, with $\sum_j p_{ij} = 1,\ \forall i$, and $p_{ii} = 0$, see Ross (2007) for a full description. Clearly, for physical or logical reasons, some additional p_{ij} matrix entries could be zero.

4.1 Inference for the CTMC parameters

Unless based on a specific parametric model, we proceed as follows. For the permanence rates, $\nu_i,\ i = 1, \cdots, m$, we assume that the time until state i is left follows an exponential distribution, $T_i \sim \mathcal{E}(\nu_i)$. We also assume a gamma prior $\nu_i \sim \mathcal{G}(\alpha_i, \beta_i)$. Then, provided we have collected data about transition counts, $n_{ij},\ i, j = 1, \ldots, m,\ j \neq i$, from state i to state j, and sojourn times, $t_{i1}, t_{i2}, \ldots, t_{in_i}$, in state i, the posterior is

$$\nu_i | data \sim \mathcal{G}\left(\alpha_i + n_i, \beta_i + \sum_{j=1}^{n_i} t_{ij}\right),$$

with $n_i = \sum_j n_{ij}$, see, e.g., French and Ríos Insua (2000).

Regarding the transition probabilities, we assume that the entries in the $i-$th row of P follow a Dirichlet distribution

$$(p_{i1}, \ldots, p_{i,i-1}, p_{i,i+1}, \ldots, p_{im}) \sim \\ \sim \mathcal{D}\left(\delta_{i1}, \ldots, \delta_{i,i-1}, \delta_{i,i+1}, \ldots, \delta_{im}\right),$$

where δ_{ij} will be zero in case the corresponding p_{ij} is known to be zero. Then, the posteriors are:

$$(p_{i1}, \ldots,\ p_{i,i-1},\ p_{i,i+1}, \ldots,\ p_{im}) | data \sim \\ \sim \mathcal{D}(\delta_{i1} + n_{i1}, \ldots,\ \delta_{i,i-1} + n_{i,i-1},\ \delta_{i,i+1} + n_{i,i+1}, \ldots,\ \delta_{im} + n_{im}),$$

see French and Ríos Insua (2000) for details. More sophisticated models which take into account possible row dependence may be seen in, e.g., Diaconis and Rolles (2006).

4.2 Computing the posterior equilibrium distribution and estimating the intensity matrix

For fixed values of the p_{ij}'s and ν_i's, the equilibrium distribution $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_m)$ is obtained, if existing, through the solution of the system

$$\begin{aligned} \nu_j \pi_j = \sum_{i \neq j} r_{ij} \pi_i; \qquad & \forall j \in \{1, \ldots, m\}, \\ \sum_j \pi_j = 1; \qquad & \pi_j \geq 0, \end{aligned} \tag{4}$$

where the $r_{ij} = \nu_i p_{ij}$ are designated jumping intensities from state i to state j, see Ross (2007) for details. We define the intensity matrix, $\Lambda = (r_{ij})$, which we shall need later on, with $r_{ii} = -\sum_{j \neq i} r_{ij} = -\nu_i, \ i = 1, \ldots, m$. The π_j's may be interpreted as long-term time fractions that the system spends at various states.

Following the same reasoning as in Section 2, we consider three scenarios, depending on the precision of the involved posteriors and the number m of chain states.

1. When the posterior distributions of ν_i, p_{ij} are, respectively, very peaked around certain values, say their posterior modes, we could substitute the parameters by their corresponding estimates,

$$\hat{\nu}_i = \frac{\alpha_i + n_i - 1}{\beta_i + \sum_{j=1}^{n_i} t_{ij}}; \qquad \hat{p}_{ij} = \frac{\delta_{ij} + n_{ij} - 1}{\sum_{l \neq i} (n_{il} + \delta_{il}) - m + 1};$$

$$\hat{r}_{ij} = \hat{\nu}_i \hat{p}_{ij},$$

   for $i \neq j$. When $i = j$, we set $\hat{r}_{ii} = -\hat{\nu}_i, \ i = 1, \ldots, m$. Then, we solve system (4), to obtain the appropriate solution $\left\{\hat{\pi}_i\right\}_{i=1}^m$.

2. When there is big uncertainty about the process parameters or about the predictive equilibrium distribution (as could be asserted by following a similar exploratory analysis to that of Section 2 for the HPD regions of $(\boldsymbol{\nu}, P)$), we may proceed through sampling. Based on samples from the posteriors $\left\{\boldsymbol{\nu}^\eta, P^\eta\right\}_{\eta=1}^N$, we, consequently, obtain a sample $\left\{\boldsymbol{\pi}^\eta\right\}_{\eta=1}^N$ through the repeated solution of (4), and summarize it appropriately, if needed, through, e.g., its posterior mean

$$\hat{\pi}_i = \frac{1}{N} \sum_{\eta=1}^N \pi_i^\eta, \ i = 1, \ldots, m.$$

   At no extra cost, we use the relationship $r_{ij} = \nu_i p_{ij}$ to obtain samples from the posterior $\{r_{ij}^\eta\}_{\eta=1}^N$, $i \neq j$. For $i = j$, we use the posterior sample $\{r_{ii}^\eta = -\nu_i^\eta\}_{\eta=1}^N$, $i = 1, \ldots, m$. If needed, we could also summarize all samples appropriately, through, e.g., their means. The procedure above entails solving N linear systems with the structure of (4), something which can be computationally expensive as the number of states grows. Several algorithmic procedures have been proposed to solve that issue, being the GTH Grassmann et al. (1985) algorithm one of the most efficient and easy to implement, see Stewart (2007) or Grassmann (1990) for details. The GTH algorithm is a direct method and actually computes an LU factorization of Λ, whose computational cost is bounded above by $\mathcal{O}(m^3)$ flops. Given that Λ is rather sparse in most real systems, this bound is usually significantly reduced. Therefore, for m small or medium, this is perfectly affordable. However, for large values of m, iterative methods are known to perform better than direct ones, see Stewart (2007). Within these methods, the coefficients matrix is used only in terms of matrix-vector products and, therefore, are especially useful for large sparse systems. Such methods start with an initial guess and compute a sequence of approximations which converges to the solution of the linear system at hand, see Golub and Van Loan (1996). We may take further advantage of the theoretical properties of iterative methods. Given that system (4) has to be solved N times for similar coefficients matrices, the solution obtained at a given replica may be used as seed for the solution of the next replica.

3. Solving repeatedly the above system may be still too costly computationally, and we may opt for using a ROM, following the same scheme as in Section 2, to obtain an estimate of the predictive distribution defined by

$$\pi_i = \sum_{k=1}^{\kappa} q^k \pi_i^k, \ i = 1, \ldots, m.$$

4.3 Reliability forecasting with CTMCs

We describe now how to estimate reliabilities in the HW block, distinguishing the case in which we know the initial ON state, and that in which we only know that the system is initially ON.

For each ON state $i \in \{1, 2, \ldots, l\}$, assume we may compute the reliability $R_i(t|\boldsymbol{\nu}, P) = \Pr\{T \geq t|\boldsymbol{\nu}, P, X_0 = i\}$, conditional on parameters $\boldsymbol{\nu}, P$, where T is the random variable which represents time passed until system failure. The unconditional posterior reliability is

$$R_i\left(t|data\right) = \iint R_i\left(t|\boldsymbol{\nu}, P\right) \pi\left(\boldsymbol{\nu}, P|data\right) \,\mathrm{d}\boldsymbol{\nu}\, \mathrm{d}P, \tag{5}$$

which will typically be approximated through simulation

$$R_i\left(t|data\right) \simeq \frac{1}{N} \sum_{\eta=1}^{N} R_i\left(t|\boldsymbol{\nu}^\eta, P^\eta\right), \tag{6}$$

for a sample $\{\boldsymbol{\nu}^\eta, P^\eta\}_{\eta=1}^N$ from the posteriors computed in 4.1. Should the computational cost of evaluating the integrand in (5) be too high, we could opt for a ROM, as explained in Section 2.

Suppose now that we do not know which ON state we are currently in. We could undertake the following approach. Let π_i be the unconditional posterior equilibrium probability for the i-th ON state, $i = 1, \ldots, l$. Then, we could do

$$R\left(t|data\right) = \frac{1}{N} \sum_{\eta=1}^{N} \sum_{i=1}^{l} \frac{\pi_i^\eta}{\bar{\pi}^\eta} \, R_i\left(t|\boldsymbol{\nu}^\eta, P^\eta\right),$$

where $\bar{\pi}^\eta = \sum_{j=1}^{l} \pi_j^\eta$, and $\{\pi_i^\eta\}_{\eta=1}^{N}$ is a sample from the posterior calculated in Section 4.2, associated with $\{\boldsymbol{\nu}^\eta, P^\eta\}_{\eta=1}^{N}$.

The key issue is then how to compute the reliabilities $R_i(t|\boldsymbol{\nu}, P)$, $i = 1, \ldots, l$. To do so, we adopt the strategy of subsuming all OFF states into an absorbing state which we designate a, conveniently redefining the transition probabilities, see Figure 2.

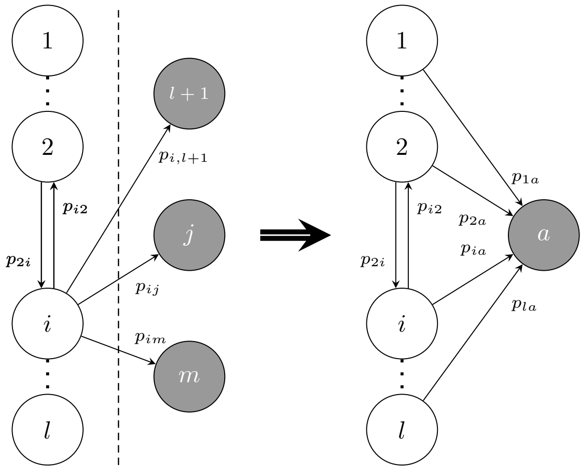


Figure 2: Subsuming the OFF states in state a

The transition probability matrix of the modified chain is

$$P_1 = \left(\begin{array}{c|c} \Pi_1 & \boldsymbol{p} \\ \hline \mathbf{0}^T & 1 \end{array} \right),$$

where $\Pi_1 = (p_{ij})$, $p_i = \sum_{k=l+1}^{m} p_{ik}$, and $i, j = 1, \ldots, l$. The intensity matrix of such chain will be designated

$$\Lambda_1 = \left(\begin{array}{c|c} \Omega_1 & \boldsymbol{\omega} \\ \hline \mathbf{0}^T & 0 \end{array} \right),$$

with $\Omega_1 = (r_{ij})$, $\omega_i = -\sum_{j=1}^{l} r_{ij}$, and $i, j = 1, \ldots, l$.

Note now that the time until failure coincides with the time until absorption (by state a) of our modified Markov chain, τ, which is known to have a phase-type distribution, given $\boldsymbol{\nu}, P$, see e.g. Bladt (2005):

$$\tau|\boldsymbol{\nu}, P \sim PH(\boldsymbol{\pi}^{(\mathbf{0})}_{\text{ON}}, \Omega_1|\boldsymbol{\nu}, P),$$

where $\boldsymbol{\pi}^{(\mathbf{0})}_{\text{ON}}$ is an initial state probability l-vector over ON states, see Cano et al. (2010) for details. The system reliability is then

$$R(t|\boldsymbol{\nu}, P) = \left(\boldsymbol{\pi}^{(\mathbf{0})}_{\text{ON}}\right)^T \exp(\Omega_1 t|\boldsymbol{\nu}, P)\boldsymbol{e},$$

where $\boldsymbol{e} = (1, \ldots, 1)^T$ is the l-vector of 1's. The computation of $R_i(t|\boldsymbol{\nu})$ is straightforward, simply setting $\boldsymbol{\pi}^{(\mathbf{0})}_{\text{ON}}$ as the vector of 0's with its i-th entry equal to 1.

5 Case study

We provide now an analysis of our university resource planning system. The system is composed by four blocks, connected in series. The Internet server and the database are blocks dealt with standard models. The ERP infrastructure is a HW block which will be modelled through a 9-state CTMC, whereas the software ERP modules are nonmature pieces of software which will be dealt with the SRGM mixture-based selection strategy in Section 3.3.

We have collected data of the system from March 2009 to June 2009, including some peak periods, e.g. issuing the employees' Withholding Tax Certificates in March, examinations periods for the students in May/June, or dismissing/hiring staff in June, among others. For each subsystem, we shall consider operation until the last failure is registered. This will simplify analysis by neglecting data censoring. Time units will be hours and the operation of the system will be daily, in the sense that the whole system is used offline for a couple of hours everyday for backup purposes, while the rest of the time is online. In all cases, we shall use relatively vague proper priors. We first consider each block and, then, the whole system.

5.1 The Internet server

The Internet server is a fairly stable system which may be analyzed through a standard exponential model of parameter λ_{is}. We used a weak gamma prior reflecting also that we expect about one failure every week, that is one failure every 154 hours. We model this with a $\mathcal{G}(1/(154 \cdot 1540), 1/1540)$, whose mean and variance are 1/154 failures/hour and 10 (failures/hour)^2, respectively.

Over the operation time we registered $n_{is} = 30$ failures, with interfailure time sum equal to 2879.17 hours. Interfailure times are available in Figure (3a). The sample coefficient of variation was 1.0178. The MLE was 0.0104. The posterior is

$$\lambda_{is}|data \sim \mathcal{G}\left(\frac{1}{154 \cdot 1540} + 30, \frac{1}{1540} + 2879.17\right).$$

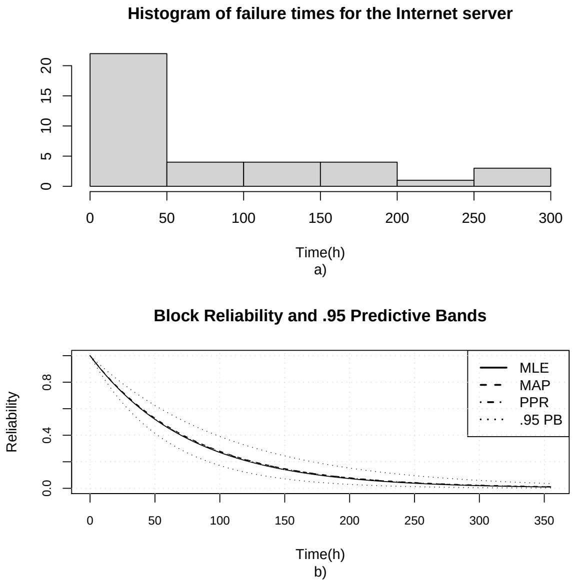


Figure 3: Time histogram and reliability for the internet server

The posterior mode is 0.0101. Figure (3b) represents the reliability $e^{-\lambda_{is} t}$ based on the MLE, based on the MAP, and the posterior predictive reliability

$$\left(\frac{\beta_{is} + \sum_j t_{j_{is}}}{\beta_{is} + \sum_j t_{j_{is}} + t} \right)^{\alpha_{is} + n_{is}},$$

where $(\alpha_{is}, \beta_{is})$ are the gamma prior parameters of λ_{is}, and .95 predictive probability bands, showing the relevance of assessing uncertainty, since the bands are not concentrated around the posterior predictive reliability.

5.2 The ERP infrastructure

We consider now the multiserver system functioning to support our university ERP. We shall briefly outline the basic details, referring to Cano et al. (2010) for a full description. The general architecture of our ERP is shown in Figure 4. A user makes a petition to the system. The petition passes through an active/active web-cache (WC) cluster balancer which distributes the load between four application (AP) servers. The balancer works if at least one of its two WC servers is up. The four AP servers work on a 2-out-of-4 basis, accessing the database and completing the service to the user.

The transition diagram of our system is shown in Figure 5, along with jumping intensities $r_{ij} = \nu_i p_{ij}$. Each ON state in the Markov chain is defined by two

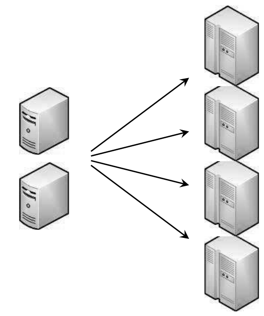


Figure 4: Architecture of our university ERP infrastructure

values, indicating the number of nonfunctioning WC and AP servers, respectively. For example, state 2 indicates that 1 WC and 0 AP servers are down, but the whole system is ON. On the other hand, OFF states (in grey) mean that all WC servers are down (State 7; Failure type I); that there are more than two AP servers down (State 8; Failure type II); or that an error in the balancer detection process has occurred (State 9; Failure type III); respectively.

The λ's and the μ's are, respectively, the failure and repair rates of the different components. Specifically, λ_{WC} and μ_{WC} are the failure and repair rates of the WC servers; λ^i_{AP}, $i = 2, 3, 4$ is the (constant) failure rate when there are i AP servers up, satisfying $\lambda_4^{\text{AP}} < \lambda_3^{\text{AP}} < \lambda_2^{\text{AP}}$; μ_{AP} is the repair rate of the AP servers. Only one server (WC or AP) can be repaired at a time. The WC cluster balancer also detects and disconnects failed AP servers, but such detection process has a probability α of success. When the system falls into an OFF state, due to too many WC or AP servers down, or to an imperfect detection process, the repair rate to recover the system from such breakdowns is ρ. The permanence rates and transition probabilities are easily recovered from the jumping intensities.

A summary of the data is displayed in Table 2, where we have displayed the transition counts n_{ij} and, in parentheses, the total sojourn time (in hours) that the system spent at state i before jumping into state j. For example, the second entry on the first row, 4(265.38), means that four transitions have occurred from State 1 into State 2, and that the system spent a total time of 265.38 hours in State 1 before jumping into State 2, over the four transitions.

We assign diffuse gamma priors to the failure and repair rates. Based on the data in Table 2, we get the corresponding posteriors. This information is summarized in Table 3. For convenience, we have expressed all rates in terms of failures/repairs per fortnight.
To compute the posterior of, e.g., λ_{WC}, the only transitions we have to take into account are (1,2), (2,7), (3,4), (4,7), (5,7), and (6,5), see Figure 5. Once

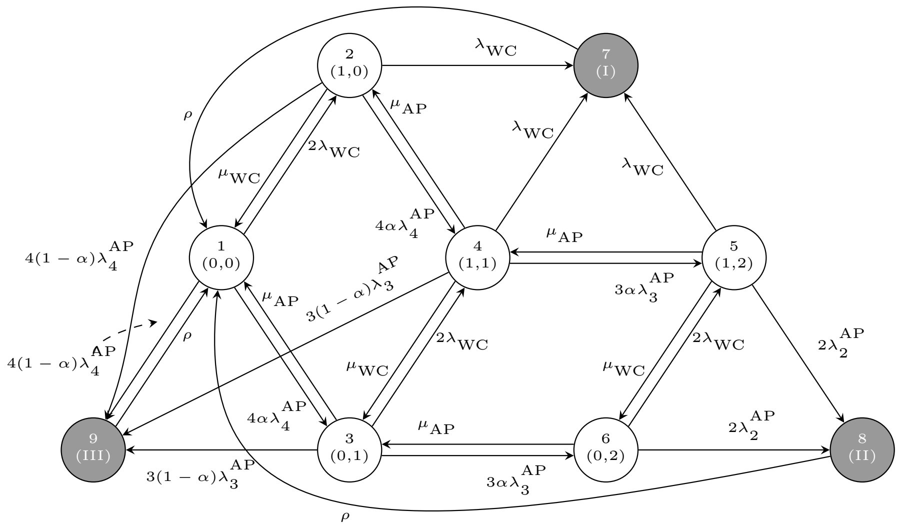


Figure 5: Transition diagram for university ERP infrastructure with jumping intensities

with the posteriors for the failure and repair rates, we can obtain samples from the posteriors for the ν's and the p_{ij}'s, using their relationships in terms of the failure and repair rates. Then, to obtain a sample from the posterior equilibrium distribution, we iteratively solve system (4), for various posterior samples of $\boldsymbol{\nu}$ and P.

Now, by adopting the analysis outlined in Section 4.3, we provide the reliability analysis in Figure 6, which plots $R_1(t|\hat{\boldsymbol{\nu}}, \hat{P})$, plugging in the MLEs (dashed line) and the posterior modes (solid line) as estimates of $(\boldsymbol{\nu}, P)$, along with .95 predictive bands (dotted lines) around the expected reliabilities (dotted-dashed line), as computed from (6). Note that there is a lot of uncertainty which is ignored through the standard approach based on plugging in parameter estimates. As we can observe, the predictive bands are quite separated from the posterior mean reliability, with relative errors close to 100% for time values around 250 hours (∼10 days). The values of the reliability conditional on the posterior modes of $(\boldsymbol{\nu}, P)$ are, in practice, the same than those of the posterior mean reliability. On the other hand, the reliability conditional on the MLEs of $(\boldsymbol{\nu}, P)$ provides considerably lower values, showing the poor quality of the MLEs when little data are available, as it is the case.

Table 2: Transition counts and sojourn times

		Final state								
		1	2	3	4	5	6	7	8	9
Initial state	1	–	4 (265.38)	3 (2390.23)	–	–	–	–	–	1 (24.32)
	2	3 (35.47)	–	–	–	–	–	1 (15.84)	–	–
	3	1 (18.03)	–	–	1 (13.35)	–	1 (4.09)	–	–	–
	4	–	–	1 (1.57)	–	–	–	–	–	–
	5	–	–	–	–	–	–	–	1 (0.02)	–
	6	–	–	–	–	1 (8.86)	–	–	–	–
	7	1 (13.59)	–	–	–	–	–	–	–	–
	8	1 (2.21)	–	–	–	–	–	–	–	–
	9	1 (11.18)	–	–	–	–	–	–	–	–

Table 3: Prior and posterior parameters of the failure and repair rates

	Priors		Posteriors	
	α	β	α^{post}	β^{post}
λ_{WC}	0.1	0.1	7.1	0.84
λ_4^{AP}	0.092	0.096	4.092	6.71
λ_3^{AP}	0.16	0.13	1.16	0.14
λ_2^{AP}	0.225	0.15	1.225	0.15
μ_{WC}	13	0.11	17	0.21
μ_{AP}	28	0.17	29	0.22
ρ	4.5	0.067	7.5	0.142

5.3 The database

The database system used is a well-known software, tested over many years, developed by a large company with a large number of installations and appropriate revision release controls. In fact, over the peak period there were only eight failures in relation with that software, with interfailure times 385, 236, 446, 288, 452, 155, 285 and 955 hours, respectively. We, therefore, go for a mature software exponential model with a diffuse $\mathcal{G}(2 \cdot 10^{-6}, 10^{-3})$ prior, which has mean 0.002 failures/hour, and variance 2 (failures/hour)^2. In consequence, the posterior is

$$\lambda_{db}|data \sim \mathcal{G}(2 \cdot 10^{-6} + 8, 10^{-3} + 3203).$$

The posterior predictive reliability is shown in Figure 7, along with .95 predictive bands. Once again, the convenience of assessing uncertainty becomes evident.

5.4 The planner

The planner is a piece of software built on top of a database system. It consists of several modules, which accomplish different tasks. For example, two relevant modules are the Human Resource module, which manages academic and administrative tasks, and the Financial Resources module, which deals with financial and accounting aspects.

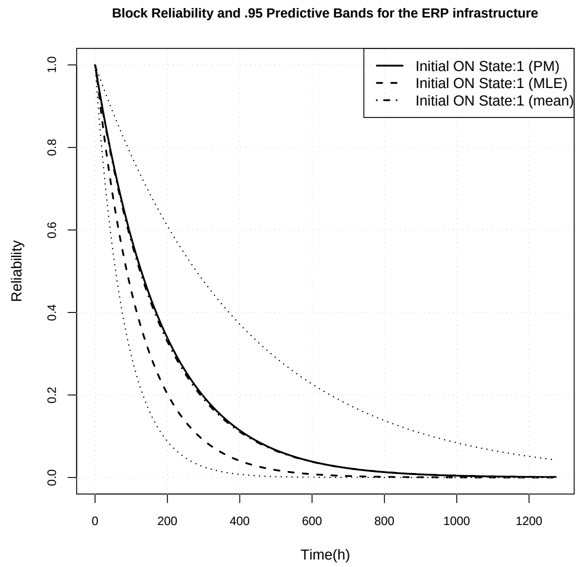


Figure 6: Reliability for the ERP infrastructure

The planner is a relatively new piece of software, designed by a relatively small company, having a relatively small number of installations. New versions are frequent as bugs are patched. We, therefore, doubt that they are mature software, as proved by frequent updates, and, therefore, shall use the model selection strategy in Section 3.3.

We have recorded 99 interfailure times for the planner, as shown in Figure (8a). We use $\gamma_1 = \gamma_2 = \gamma_3 = 1/3$. We use priors of model parameters to reflect that at peak times, we discover around one failure per day and we do not expect to discover more than three failures per day. We thus choose the following priors $(a_1, b_1) \sim \mathcal{G}(10, 10)\mathcal{G}(7, 10)$; $(a_2, b_2) \sim \mathcal{G}(100, 1)\mathcal{G}(1, 10)$; $(a_3, b_3) \sim \mathcal{G}(11, 10)\mathcal{G}(3, 10)$.

Then, with the strategy described in Section 3.3 we obtain the posterior weights $\gamma_1^{post} = 0.66$, $\gamma_2^{post} = 0.24$, $\gamma_3^{post} = 0.10$ and the reliability as calculated in (3), and reflected in Figure (8b). As with the rest of the blocks, .95 predictive bands around the mean values are also provided, showing in this case less uncertainty than with previous blocks.

5.5 System reliability

Finally, as parameters are not shared, the system reliability, given subsystem reliabilities, is computed through

$$R_{sys}|data = R_{is}|data \times R_{erp}|data \times R_{db}|data \times R_{sw}|data.$$

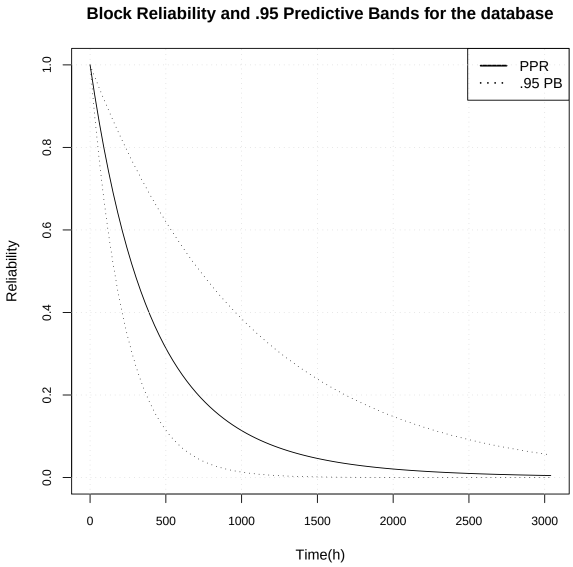


Figure 7: Reliability for the database

We plot in Figure 9 the system reliability for different times, obtained through simulation. The plot includes the posterior reliability mean together with .95 predictive reliability bounds. The picture clearly depicts the necessity of taking into account the uncertainty reflected in standard reliability analysis. As an example, suppose that we want to estimate the probability that the system will still be ON after two working days. The graphic predicts an expected reliability of 0.14, but with a .95 predictive band around it of $(0.11, 0.18)$, which amounts to relative errors of around 30%, clearly inadmissible if we are dealing with safety critical systems.

As the system has a series structure, therefore, its reliability will be influenced by the least reliable component. In that sense, a suitable sensitivity measure of the reliability of a given system with respect to its components is the partial derivative of the system reliability with respect to the component reliabilities. This informs us about which components are more critical in terms of reliability for the system. In our example, all the derivatives are straightforward and they lead to:

$$\frac{\partial R_{sys}}{\partial R_i} = \prod_{\substack{j \neq i \\ i,j \in \Omega}} R_j,$$

where $\Omega = \{is, erp, sm, db\}$. We plot the sensitivities in Figure 10.

As we can observe upon comparison of Figures 9 and 10c, our sensitivity analysis reveals that the least reliable component is the planner. Its removal

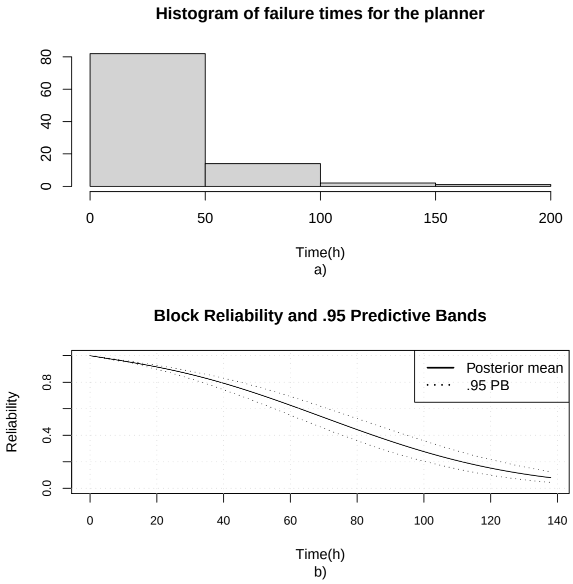


Figure 8: Time histogram and reliability for the planner

from the system configuration yields a remarkable increase in system reliability, specially relevant for larger times. Again, as an illustration, we consider system reliability after two days. If no component is removed, the expected system reliability is approximately 0.14, jumping up to 0.40 if the planner is removed. After 4 days, the situation is even more drastic, in terms of relative improvement: the expected value of system reliability leaps from 0.02 up to 0.16 when dropping the planner.

The other three blocks have similar sensitivities, as can be seen in Figures 10a, 10b and 10d. This type of information should be relevant when defining maintenance plans and service level agreements.

6 Conclusions and extensions

HW/SW systems are present in many fields of human activity. Estimating their reliability is therefore becoming increasingly important, specially for safety critical systems in areas such as finance, aerospace and energy, among others. Although there are several software packages available to support reasonably complex reliability analyses, they have several limitations, being possibly the most important, the fact that they provide little support to Bayesian analysis. With this drawback in mind, we have focused, in this paper, on Bayesian analysis of such systems.

We have developed a complete method to assess the reliability of complex

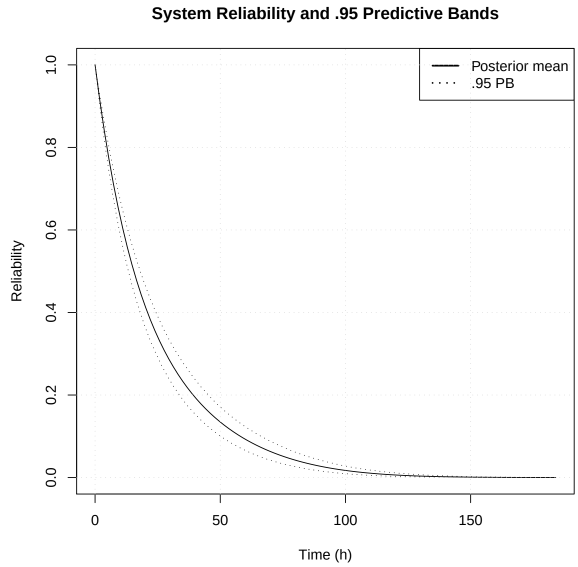


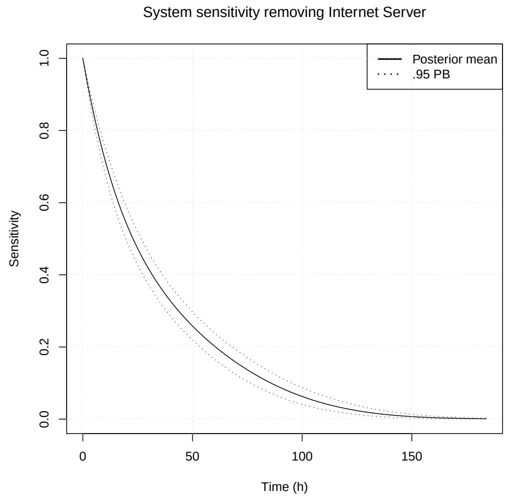
Figure 9: System reliability

HW/SW systems, devoting our attention to three key items: description of complex systems in terms of series and parallel RBDs with pending blocks, and forecasting their reliabilities; software block reliability modeling, through mixture based SRGM selection; and hardware block reliability modeling, through CTMCs.

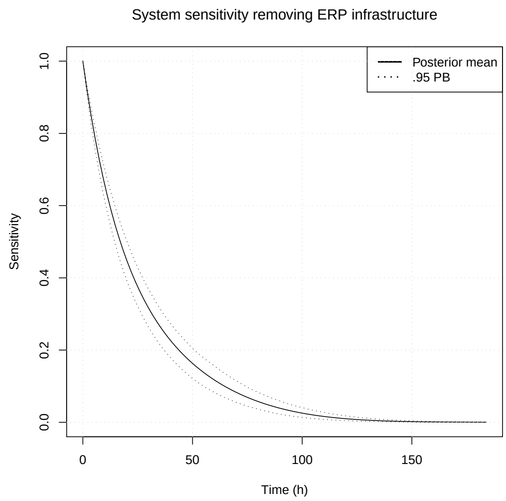
We are currently developing a computational environment to fully support the tasks we have to deal with in this analysis. We have illustrated the basic features of our design, with an example, consisting of the analysis of a schematic version of our university resource planner, paying attention to the computation of the reliability of each system component, and the reliability of the whole system.

A related aspect of interest is to analyze the dependence of system reliability with respect to the parameters which model each block, for both software and hardware components. As a further step in this direction, one would investigate the influence of prior distributions of the block parameters in the system predicted reliability, as typical in robust Bayesian analysis, see Ríos Insua and Ruggeri (2000).

Note also that RBDs essentially assume independent blocks. In this type of application, this condition might not be acceptable (e.g. because the HW components share location) and models taking into account such possible dependence should be developed.



(a) Removing Internet server

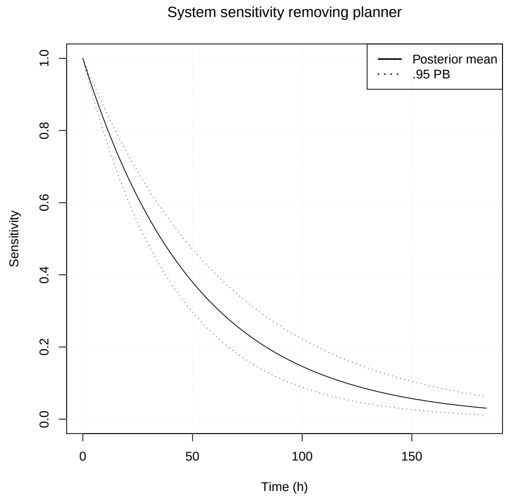


(b) Removing ERP infrastructure

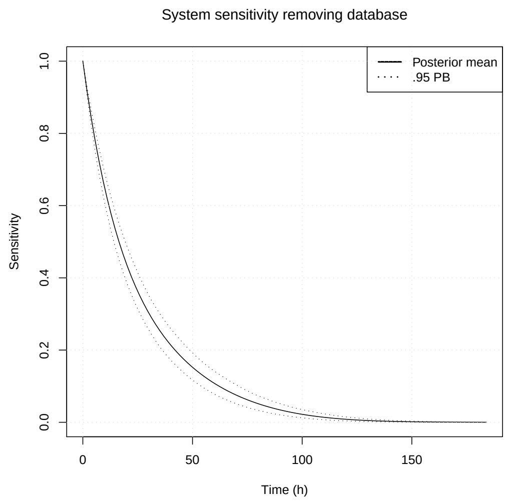
Figure 10: Sensitivity analysis

Acknowledgments

Work supported by projects from the Spanish Ministry of Science and Innovation program MTM2009-14087-C04-01, and the Government of Madrid RIESGOS-CM program S2009/ESP-1685. We are grateful to Luis Redondo, from MTP, for suggesting us this research area. The support of SAMSI and the ALGODEC



(c) Removing planner



(d) Removing database

Figure 10: (cont.) Sensitivity analysis

program are also gratefully acknowledged.

References

Dale C, Anderson T. Safety-Critical Systems: Problems, Process and Practice. Proc. of the 17th Safety-Critical Systems Symposium, Brighton, 2009.

Dunn W. Designing Safety-Critical Computer Systems. Computer 2003;36(11):40–6.

Cukic B, Chakravarthy D. Bayesian Framework for Reliability Assurance of a Deployed Safety Critical System. In: Proceedings of the Fifth IEEE International Symposim on High Assurance Systems Engineering (HASE'00), Albuquerque, 2000, p. 321–9.

Goel A, Soenjoto J. Models for Hardware-Software System Operational-Performance Evaluation. IEEE Trans Reliab 1981;30(3):232–9.

Welke S, Johnson B, Aylor J. Reliability Modeling of Hardware/Software Systems. IEEE Trans Reliab 1995;44(3):413–8.

Pukite J, Pukite P. Modeling for Reliability Analysis: Markov Modeling for Reliability, Maintainability, Safety, and Supportability Analyses of Complex Computer Systems. IEEE Computer Society Press; 1998.

Xie M, Dai Y, Poh K. Computing Systems Reliability: Models and Analysis. Kluwer Academic/Plenum Publishers; 2004.

Lollini P, Bondavalli A, Di Giandomenico F. A Decomposition-Based Modeling Framework for Complex Systems. IEEE Trans Reliab 2009;58(1):20–33.

Distefano S, Puliafito A. Reliability and Availability Analysis of Dependent-Dynamic Systems with DRBDs. Reliab Eng Syst Saf 2009;94(9):1381–93.

Levitin G. Block Diagram Method for Analyzing Multi-State Systems with Uncovered Failures. Reliab Eng Syst Saf 2007;92(6):727–34.

Trivedi K. Probability and Statistics with Reliability, Queuing, and Computer Science Applications, 2nd ed. New York: John Wiley; 2001.

Trivedi KS, Vasireddy R, Trindade D, Nathan S, Castro R. Modeling High Availability Systems. In: Proceedings of the Pacific Rim Dependable Computing Conference (PRDC), Riverside, 2006, p. 154–64.

Birolini A. Reliability Engineering: Theory and Practice. 5th ed. Springer Verlag; 2007.

Singpurwalla N, Wilson S. Statistical Methods in Software Engineering: Reliability and Risk. Springer; 1999.

Ross S. Introduction to Probability Models. Academic Press; 2007.

Ball MO. Network Reliability. In: Ball MO, Magnanti TL, Monma CL, Nemhauser GL, editors. Handbooks in Operations Research and Management Science, 7: Network Models, North Holland: Elsevier; 1995, p. 673–762.

Monga A, Zuo M. Optimal Design of Series-Parallel Systems Considering Maintenance and Salvage Value. Comput Ind Eng 2009;40(4):323–37.

Glynn P. Problems in Bayesian Analysis of Stochastic Simulation. In: Proceedings of the 18th Conference on Winter Simulation, Washington, 1986, p. 376–9.

Draper D. Assessment and Propagation of Model Uncertainty. J R Stat Soc B 1995;57(1):45–97.

Berger J, Ríos Insua D. Recent developments in Bayesian inference with applications in hydrology. In: Parent E, Hubert P, Bobée B, Miquel J, editors. Statistical and Bayesian Methods in Hydrological Sciences, Paris: UNESCO Press; 1998, p. 56–80.

Schmeiser B. Simulation Experiments. In: Heyman DP, Sobel MJ, editors. Handbooks in Operations Research and Management Science, 2: Stochastic Models, North Holland: Elsevier; 1990, p. 295–330.

Grigoriu M. Reduced Order Models for Random Functions. Application to Stochastic Problems. Appl Math Model 2009;33(1):161–75.

Zacks S. Introduction to Reliability Analysis. Berlin: Springer-Verlag; 1992.

Kuo L. Software Reliability. In: Dey D, Rao CR, editors. Handbook of Statistics, 25: Bayesian Thinking, Modeling and Computation, North Holland: Elsevier; 2005, p. 929–63.

Schneidewind N. Analysis of Error Processes in Computer Software. In: Proceedings of the International Conference on Reliable Software, Los Angeles, 1975, p. 337–46.

Guida M, Calabria R, Pulcini G. Bayes Inference for a Non-Homogeneous Poisson Process with Power Intensity Law. IEEE Trans Reliab 1989;38(5):603–9.

Ruggeri F. On the Reliability of Repairable Systems: Methods and Applications. In: Di Bucchianico A, Mattheij RMM, Peletier MA, editors. Mathematics in Industry, 8: Progress in Industrial Mathematics at ECMI 2004, Berlin: Springer; 2006, p. 535–53.

Kuo L, Lee JC, Choi K, Yang TY. Bayes Inference for S-shaped Software-Reliability Growth Models. IEEE Trans Reliab 1997;46(1):76–80.

Gamerman D, Lopes H. Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference. Chapman & Hall/CRC; 2006.

French S, Ríos Insua D. Statistical Decision Theory. Arnold; 2000.

Diaconis P, Rolles SWW. Bayesian Analysis for Reversible Markov Chains. Ann Stat 2006:34(3):1270–92.

Grassmann W, Taksar M, Heyman D. Regenerative Analysis and Steady State Distributions for Markov Chains. Oper Res 1985;33(5):1107–16.

Stewart W. Performance Modelling and Markov Chains. In: Bernardo M, Hillston J, editors. Lecture Notes in Computer Science Vol. 4486: Formal Methods for Performance Evaluation, Berlin: Springer; 2007, p. 1–33.

Grassmann, W. Computational Methods in Probability Theory. In: Heyman DP, Sobel MJ, editors. Handbooks in Operations Research and Management Science, 2: Stochastic Models, North Holland: Elsevier; 1990, p. 199–254.

Golub G, Van Loan C. Matrix Computations. Johns Hopkins University Press; 1996.

Bladt M. A Review on Phase-Type Distributions and their Use in Risk Theory. Astin Bull 2005;35(1):145–61.

Cano J, Moguerza JM, Ríos Insua D. Bayesian Reliability, Availability and Maintainability Analysis for Hardware Systems Described Through Continuous Time Markov Chains. *To appear in* Technometrics.

Ríos Insua D, Ruggeri F. Robust Bayesian Analysis. Springer; 2000.