The Journal of Systems and Software 86 (2013) 1923–1932

Contents lists available at SciVerse ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

An approach to software reliability prediction based on time series modeling

Ayman Amin [a,*], Lars Grunske [b], Alan Colman [a]

[a] Faculty of Information and Communication Technologies, Swinburne University of Technology, Australia
[b] Institute of Software Technology, University of Stuttgart, Germany

ARTICLE INFO

Article history:
Received 8 October 2011
Received in revised form 3 February 2013
Accepted 2 March 2013
Available online 22 March 2013

Keywords:
Reliability prediction
Software Reliability Growth Models
Time series ARIMA models

ABSTRACT

Reliability is the key factor for software system quality. Several models have been introduced to estimate and predict reliability based on results of software testing activities. Software Reliability Growth Models (SRGMs) are considered the most commonly used to achieve this goal. Over the past decades, many researchers have discussed SRGMs' assumptions, applicability, and predictability. They have concluded that SRGMs have many shortcomings related to their unrealistic assumptions, environment-dependent applicability, and questionable predictability. Several approaches based on non-parametric statistics, Bayesian networks, and machine learning methods have been proposed in the literature. Based on their theoretical nature, however, they cannot completely address the SRGMs' limitations. Consequently, addressing these shortcomings is still a very crucial task in order to provide reliable software systems. This paper presents a well-established prediction approach based on time series ARIMA (Autoregressive Integrated Moving Average) modeling as an alternative solution to address the SRGMs' limitations and provide more accurate reliability prediction. Using real-life data sets on software failures, the accuracy of the proposed approach is evaluated and compared to popular existing approaches.

1. Introduction

Software has been increasingly used in our daily life and has become a crucial part of critical and non-critical applications. Because of this increasing use and importance, the assurance of software quality becomes an issue of critical concern. Software quality can be expressed by quality requirements or attributes such as reliability, availability, safety, security, and performance. Among these software quality attributes, software reliability is generally considered as the most important factor (Palviainen et al., 2011). It quantifies software faults and failures, which can lead to serious consequences in safety-critical systems as well as in normal business (Lyu, 2007). Therefore, assessing, estimating, and predicting software reliability have been increasingly demanded in projects in order to achieve highly reliable software systems.

To address this demand, starting from the work of Jelinski and Moranda (1972) a considerable number of Software Reliability Growth Models (SRGMs) has been proposed. These models specify the form of a stochastic process that describes the software behaviour with respect to software failures in order to be used for reliability estimation, measurement of the current state, and prediction of the future state (Lyu, 1996). (For more details about these models, see Goel, 1985; Lyu, 1996.) These models require underlying assumptions to be applied, such as independence of time between failures, immediate correction of detected faults, and fault correction without introducing new faults. However, some of these assumptions seem unrealistic (Goel, 1985; Zeitler, 1991; Wood, 1997). In addition, SRGMs' applicability has become a critical issue because there is no single model which can be universally used in all the situations (Goel, 1985; Sharma et al., 2010). Regarding SRGMs' predictability as the main goal of using SRGMs, these models have been shown to be able to model and fit the past software failures data, however, they do not give accurate prediction (Xie et al., 1997).

To address these issues, approaches based on non-parametric statistics (Robinson and Dietrich, 1987; Barghout et al., 1998) and Bayesian networks (Neil and Fenton, 1996; Bai et al., 2005; Fenton et al., 2008; Wiper et al., 2012) have been proposed as solutions in the literature. However, all of these solutions are not complete (Goel, 1985); because if one solution is able to address one issue, it cannot address the others. For example, non-parametric approaches are able to address the unrealistic assumptions issue; however, they cannot completely address the applicability and predictability issues.

As a result, approaches based on neural networks and other machine learning methods (Karunanithi et al., 1992; Pai and Hong, 2006; Kiran and Ravi, 2007; Zaidi et al., 2008; Lo, 2009; Yang et al., 2010; Moura et al., 2011) have been introduced. However, the main

* Corresponding author.
E-mail addresses: aabdellah@swin.edu.au, aymanaamin@gmail.com (A. Amin), lars.grunske@informatik.uni-stuttgart.de (L. Grunske), acolman@swin.edu.au (A. Colman).

0164-1212/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2013.03.045

disadvantage of these approaches is that they require a large training data set as input/output examples to get a correct learning, which is computationally intensive and time consuming process. In addition, some authors (Singpurwalla and Soyer, 1985; Chatterjee et al., 1997; Ho and Xie, 1998; Xie and Ho, 1999) have tried to use ARIMA models as an alternative solution. However, these trials are conducted in an abstract way, and they have critical limitations in: (1) checking and satisfying the underlying assumptions of ARIMA models, which are necessarily required to get a correct ARIMA model for the given software reliability data; (2) how to practically construct in specific steps an ARIMA model for the given software reliability data. Consequently, proposing an approach which addresses these limitations and gives accurate reliability prediction is urgently needed to provide highly reliable software systems.

In this paper, we present a well-established prediction approach based on time series ARIMA modeling as a good alternative solution to address the limitations of SRGMs and the existing approaches and provide more accurate reliability prediction. The main advantage of this prediction approach is that it is a data-oriented approach and therefore does not require any restrictive assumptions on the environment of the software system under analysis. In addition, the approach utilizes statistical methods to check whether the given software reliability data satisfies the underlying assumptions of ARIMA models, and in the case of dissatisfaction it uses suitable statistical remedies. Moreover, the approach constructs an ARIMA model for the given software reliability data set according to specific and clear steps.

In brief, the proposed approach consists of six phases: (1) observing the software reliability data over a period of time, and checking for and satisfying ARIMA models assumptions; (2) identifying adequate ARIMA models to describe this data; (3) fitting the identified models using estimation methods; (4) checking the adequacy of these estimated models; (5) selecting the best model among the adequate models; and finally (6) using the constructed prediction model to predict future reliability values.

The rest of this paper is organised as follows. Related work including SRGMs and alternative solutions are discussed in Section 2. Section 3 introduces time series ARIMA models background. Our proposed prediction approach for software reliability is presented in Section 4. Evaluation of our approach and its comparison to selected existing approaches are presented in Section 5. Finally, Section 6 concludes the paper.

2. Related work

2.1. Software Reliability Growth Models

The goal of Software Reliability Growth Models (SRGMs) is to specify the form of a stochastic process that describes the software behaviour with respect to software failures. This stochastic process of failures behaviour can then be used for reliability estimation, measurement of the current state, and prediction of the future state (Lyu, 1996). SRGMs assume that corrections and modifications are made to the system as the testing progresses resulting in a decreasing failure rate and increasing reliability.

Starting from the work of Jelinski and Moranda (1972), SRGMs have received much attention where hundreds of publications have been written and more than 100 different forms of models have been proposed (Lyu, 2007). These proposed models can be classified into two groups: time between failures models and failure count models (Goel, 1985). The Jelinski-Moranda model (Jelinski and Moranda, 1972) is the earliest of the time between failures models, while the Goel–Okumoto Nonhomogeneous Poission Process (NHPP) model (Goel and Okumoto, 1979) is the earliest of the failure count models (Lyu, 1996).

SRGMs require *underlying assumptions* to be used. However, some of these assumptions are questionable and seem *unrealistic*. In the following, the most important assumptions are discussed.

1. Independence of time between failures: Time between failure models assume that successive failure times are independent of each other. This assumption can be satisfied if successive test cases are chosen randomly. In practice, systematic testing, especially functional testing, is not based on independent test cases, so that the test process is likely not random and has dependence (Goel, 1985). This in turn means that the time between failures are dependent.
2. Immediately correcting detected faults: Some models require this assumption and assume that faults are immediately corrected when they are discovered. This assumption is satisfied only if the testing process stops until the detected fault is repaired. In many testing situations, the testing process continues without correcting a detected fault, and the fault detection process behaves as if the fault had been corrected (Goel, 1985; Huang and Lyu, 2011; Kapur et al., 2011). In this case, if there are N faults when a fault is detected and testing continues, the fault detection rate will be different than assumed by the model since the software program will still have N faults rather than the assumed $(N-1)$ faults (Wood, 1997).
3. Correcting a fault without introducing new faults: This assumption is required to ensure that the failures process have a monotonic decreasing pattern. Thus, the only way to satisfy is to ensure that the correction process does not introduce new faults (Goel, 1985). However, it is agreed that fault correction can always introduce new faults (Wood, 1997). An argument can be made that if introduced faults constitute a very small fraction of the fault population, their practical effect on model results will be minimal (Goel, 1985).

It should be noted that any software development process is environment-dependent, and some assumptions may be realistic only in certain situations. More discussions about SRGMs assumptions are presented in Goel (1985), Zeitler (1991), Wood (1997). For example, Zeitler Zeitler (1991) presented a well-made argument about these assumptions, and the author concluded that the best solution to overcome the problem of unrealistic assumptions is to use time series ARIMA models. Also, the paper presented statistical proof that ARIMA models are able to accurately map the stochastic behaviour of software failures processes.

Applicability is another important issue in using SRGMs. Currently, there is no single model which can be universally used in all the situations (Goel, 1985). Hence, various models are applied, and then the best one is selected according to the given situation. This process is a time consuming and requires experts to choose the best model based on the development environment considerations.

Predictability is considered the main goal from using SRGMs. It can be defined as the ability to forecast future reliability values using the collected data, which is usually in the form of time between failures or the number of failures. When asking if a model is giving accurate reliability measures, it is really asking if it is predicting accurately (Brocklehurst et al., 1990). Another factor that shows the importance is that the ability to capture the past accurately does not necessarily imply an ability to predict the future accurately (Brocklehurst and Littlewood, 1992). Although SRGMs have been shown to be able to model and fit the past software failure data, they do not give accurate predictions (Xie et al., 1997).

2.2. Alternative solutions

The research community has presented many solutions to address the standard SRGMs' issues. Non-parametric statistics

(Robinson and Dietrich, 1987; Barghout et al., 1998) and Bayesian networks (Neil and Fenton, 1996; Bai et al., 2005; Fenton et al., 2008; Wiper et al., 2012) have been used to address the issue of unrealistic assumptions. However, these solutions do not remedy the other issues of applicability and predictability. In addition, neural networks and other machine learning methods (Karunanithi et al., 1992; Kiran and Ravi, 2007; Zaidi et al., 2008; Lo, 2009; Yang et al., 2010; Moura et al., 2011) have been applied to address some of the issues. Their main disadvantage is that they need extensive learning to give accurate measures and forecasts, which is a computationally intensive and time consuming process.

To solve the SRGMs applicability problem, some solutions have been presented using combination of models (Sharma et al., 2010; Lyu and Nikora, 1992; Popeniu and Boro, 1996; Raj Kiran and Ravi, 2008; Li et al., in press). The primary drawbacks of these solutions are that they are very complicated, and do not guarantee getting the best model. For example, a Supermodel (Popeniu and Boro, 1996) is a combination of SRGMs. This model needs intensive computations to be estimated, and it also has the same issues related to unrealistic assumptions and predictability.

A number of researchers have tried to apply time series models, especially ARIMA models, for reliability prediction (Singpurwalla and Soyer, 1985; Chatterjee et al., 1997; Xie and Ho, 1999). The main result is that time series models can address SRGMs' issues, and they have the ability to give more accurate forecasts. In addition, Junhong et al. Junhong et al. (2005) proved that the Goel–Okumoto NHPP model (Goel and Okumoto, 1979), the most commonly used in software reliability prediction, can be transformed into an autoregressive model of order one (AR(1)). This result states that the most important software reliability model is a special case of ARIMA models, and this opens a new research area how to exploit the ARIMA modeling advantages in software reliability prediction.

However, while time series ARIMA models have the ability to address SRGMs' limitations and provide more accurate forecasts, the existing applications of these models for reliability prediction are in the initial stage and face critical limitations:

1. *Checking and satisfying ARIMA models assumptions:* The existing applications do not discuss the underlying assumptions of ARIMA models, how these assumptions can be verified in practice, and if they are not satisfied what are the remedies which can be used to address this issue. This is critical task because satisfying these assumptions is a necessary requirement to construct a correct ARIMA model for the given software reliability data.
2. *Constructing ARIMA models:* Existing approaches do not explain how these models can be practically constructed for the given software reliability data in specific and clear steps which can help others, e.g. software engineers, in using these models.

Consequently, proposing an approach which can address these limitations and provide accurate reliability prediction is crucially required. In this paper, we propose a well-established prediction approach which can help a software reliability engineer construct the correct prediction model in an easier way, thereby providing much more accurate reliability predictions relatively to the other existing approaches.

3. Time series ARIMA models background

Autoregressive Integrated Moving Average (ARIMA) models were originally proposed by Box and Jenkins (1976), and they are commonly used in practice to model time series data to forecast the future values. These models to be used to model time series data; they require some assumptions time series data should satisfy to provide accurate forecasts. In this section, we present ARIMA models and discuss their main assumptions.

3.1. ARIMA models

The time series $\{y_t\}$ is said to be generated by an ARMA model of orders p and q, denoted by ARMA(p,q), if it satisfies:

$$\phi_p(B)y_t = \theta_q(B)\varepsilon_t \quad (1)$$

where $\{\varepsilon_t\}$ is a sequence of independent normal errors with zero mean and variance σ^2. The backshift operator B is defined as $B^d x_t = x_{t-d}$. The autoregressive polynomial is $\phi_p(B) = (1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p)$ with order p and $\theta_q(B) = (1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q)$ is the moving average polynomial with order q. The autoregressive and moving average coefficients are $\phi = (\phi_1, \phi_2, \cdots, \phi_p)^T$ and $\theta = (\theta_1, \theta_2, \cdots, \theta_q)^T$ respectively. The model (1) can be simplified and written as

$$y_t = \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t \quad (2)$$

where y_t is the current stationary observation, y_{t-i} for $i = 1, \ldots, p$ are the past stationary observations, ε_t is the current error, and ε_{t-i} for $i = 1, \ldots, q$ are the past errors.

If the original time series $\{z_t\}$ is non-stationary, then d differences can be done to transform it into a stationary one $\{y_t\}$. This differences-based transformation is denoted by $y_t = \nabla^d z_t$, where $\nabla^d = (1 - B)^d$ and B is the backshift operator. In this case, the time series $\{z_t\}$ is said to be generated by an Autoregressive Integrated Moving Average (ARIMA) model of orders p, d, and q and is denoted by ARIMA(p, d, q).

3.2. Assumptions of ARIMA models

The main assumptions of ARIMA models are normality, stationarity and invertibility (Box and Jenkins, 1976). These assumptions are discussed in some details in the following:

1. *Normality:* ARIMA model assumes that the given time series data is normally distributed, as all the computations of identifying, estimating, and evaluating its parameters are based on the normal distribution. In the case of non-normally distributed time series data, it can be approximated to normal distribution using specific power transformations, e.g., Box–Cox transformation (Box and Cox, 1964).
2. *Stationarity:* Roughly speaking, stationarity means the time series has constant mean and variance, no trend and same variation, over time. When time series has no trend, it is called stationary in the mean; and when it has same variation over time, it is called stationary in the variance. However, if the original time series is non-stationary, it can be transformed into a stationary in the mean using differences and into a stationary into the variance using power transformations.
3. *Invertibility:* ARIMA model to be used to forecast the future values it should be invertible. This means the error term in ARIMA model should be expressed in the other terms, i.e. $\varepsilon_t = y_t - \sum_{i=1}^{p} \phi_i y_{t-i} - \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$, to enable forecasting. For example, in the simplest case, MA(1), error term can be written as: $\varepsilon_t = y_t - \theta\varepsilon_{t-1}$, and it can be proved that they are equivalent to: $\varepsilon_t = y_t - \sum_{i=1}^{\infty} \theta^i y_{t-i}$. This model cannot be used for forecasting future errors unless ε_t has a finite value which requires that θ to be less than one, and this is the invertibility concept.

Investigating whether the given time series data satisfies these assumptions is a critical task, because falling to satisfy the assumptions leads to selecting incorrect ARIMA model that in turn will

provide significantly incorrect forecasts. In section 4 we discuss in detail how these assumptions can be verified, and if they are not satisfied, how we can deal with this issue through an illustrative example.

4. The proposed prediction approach

The proposed approach builds on the Box–Jenkins ARIMA models and proposes a parallelized modeling procedure. The main goal of the proposed approach is to check for and satisfy ARIMA models' assumptions, and construct the best ARIMA model for the given software reliability time series data in order to provide accurate predictions.

This approach is a data-oriented approach and does not impose any restrictive assumptions on the environment of the data, in contrast to existing SRGMs. Consequently, this approach can address the SRGMs' issues discussed above. It assumes the failure time series data is dependent, which is valid for time between failures and the number of failures, and uses this dependency to identify the best model. This implies that there is no need to arbitrarily assume a model prior to data analysis, which solves both unrealistic assumptions and applicability issues. Regarding predictability, it is known in the literature that ARIMA models are the most powerful technique for accurately predicting the future values (Box and Jenkins, 1976; Makridakis et al., 2008).

In summary, as depicted in Fig. 1 the approach first checks for the underlying assumptions of ARIMA models and in the case of dissatisfaction of these assumptions uses some statistical remedies and transformations. Second, the approach identifies a combination of adequate models based on Auto-Correlation Function (ACF) and Partial Auto-Correlation Function (PACF). Third, it fits these identified models using the maximum likelihood estimation (MLE) method. Fourth, it checks the adequacy of all the estimated models. Fifth, it selects the best model among the adequate models using Akaike's information criterion (AIC) (Akaike, 1974). Finally, it uses the selected model, which is the best model to describe the given data, to predict the future values. In the rest of this section the proposed prediction approach is introduced in detail and explained by a running example of a real-life data set on software failures which is called "System 40 data". This data set is time between failures data that has been published by Musa (1980).

4.1. (P1) Data preparation

Before ARIMA models can be constructed, the given time series data should satisfy the underlying assumptions described above. Consequently in this phase, we use some powerful statistical tests to check for these assumptions, and if they are not satisfied, we try to find the suitable transformation to make the data approximately fulfils the assumptions.

Example 1. For the "System 40 data" (Musa, 1980), which is labelled "sys40" in our example, the ARIMA model assumptions can be verified as follows:

Serial dependency: The approach has used the runs test (Gibbons and Chakraborti, 2003) to test whether the time series are mutually independent. It has concluded that it is significantly serially dependent where p-value equals to 0.021 (<0.05).

Normality: The approach used the K–S test (Gibbons and Chakraborti, 2003) to test whether the sys40 data is drawn from normal distribution. It has concluded that this data is not normally distributed, and the histogram depicted in Fig. 2(a) visualizes that and explains that the data is skewed to the right or positively skewed. To transform the data to be (approximately) normally distributed, the approach has used the Box–Cox transformation and concluded that the log of the data (referred as lsys40) is approximately normally distributed, as shown in Fig. 2(b).

Stationarity: Our approach used the KPSS test (Kwiatkowski et al., 1992) to test whether the lsys40 data is stationary, and found that it is not stationary where the p-value equals to 0.028 (<0.05) and this is explained in Fig. 2(c) where the ACF decays slowly. Using the first difference, the approach concluded that the differenced data (referred as dlsys40) is stationary where the p-value equals to 0.465.

4.2. (P2) Model identification

After the time series data is prepared, the model identification phase starts which selects the best suitable parameter p, d, and q for the ARIMA(p, d, q) model. The model identification is based on ACF and PACF (Box and Jenkins, 1976) as follows: If the ACF curve decays and the PACF curve cuts off, AR models are adequate to model the processed data. In contrast, if the ACF curve cuts off and the PACF curve decays, MA models are adequate. In addition, if both ACF and PACF curves cut off or decay, ARMA models are adequate. It is worth mentioning that our proposed prediction approach does not identify only one adequate model, but rather it identifies a combination of adequate models based on the dependency structure of the given failures data set.

Example 2. The ACF and PACF of the prepared data, dlsys40, are computed and plotted in Fig. 2(d). Based on these functions, the identified ARIMA models are: ARIMA(1, 1, 0), ARIMA(0, 1, 1), ARIMA(1, 1, 1), ARIMA(0, 1, 2), and ARIMA(1, 1, 2). It is worth noting that $d = 1$ is already determined in P1.

4.3. (P3) Model estimation

In the model estimation phase, values of the parameters of the identified models in phase P2 are analysed to provide the best fit to the given time series data. Non-linear least squares and maximum likelihood estimation (MLE) are the two main methods used to estimate ARIMA models. However, the latter one is generally the preferred technique, as it is faster and gives more accurate estimates (Box and Jenkins, 1976). Therefore, it is recommended to be used in our proposed approach. To explain the main idea of the maximum likelihood estimation method, suppose $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ is a time series of the ARMA model (2), given that $\varepsilon_t \sim \text{Normal}(0,\sigma^2)$, the likelihood function, l, is given by:

$$l \propto (\sigma^2)^{-\frac{n}{2}} \exp\left\{-\frac{1}{2\sigma^2}\sum_{t=1}^{n}(\varepsilon_t)^2\right\} \tag{3}$$

where $\varepsilon_t = y_t - \sum_{i=1}^{p}\phi_i y_{t-i} - \sum_{i=1}^{q}\theta_i\varepsilon_{t-i}$. Values of the parameters ϕ_i's and θ_i's that maximize the likelihood function (3) are called maximum likelihood estimates.

Example 3. After identifying the adequate ARIMA models in P2, the approach used the MLE method for each model to estimate the parameters, and the estimates are depicted in Table 1.

4.4. (P4) Model diagnostics checking

Model checking involves testing the diagnostics of the ARIMA models to identify whether they are satisfied. If one or more diagnostics are not satisfied, the current model is inadequate and should be removed from the set of models identified in P2. The main diagnostics that need to be performed to check the fitted ARIMA models are estimates significance, invertibility and stationarity conditions, and residuals randomness.

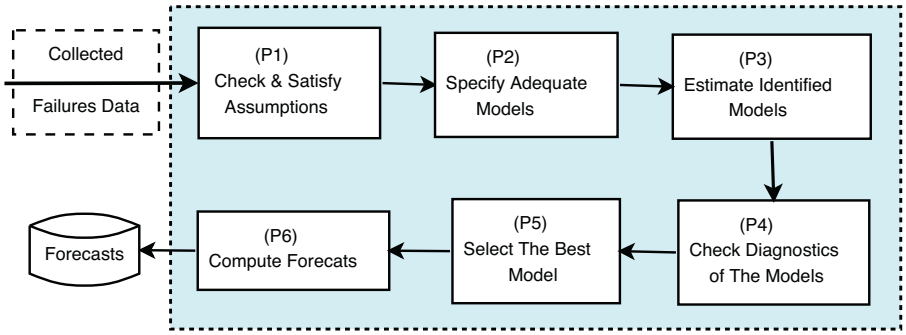

Fig. 1. The proposed prediction approach.

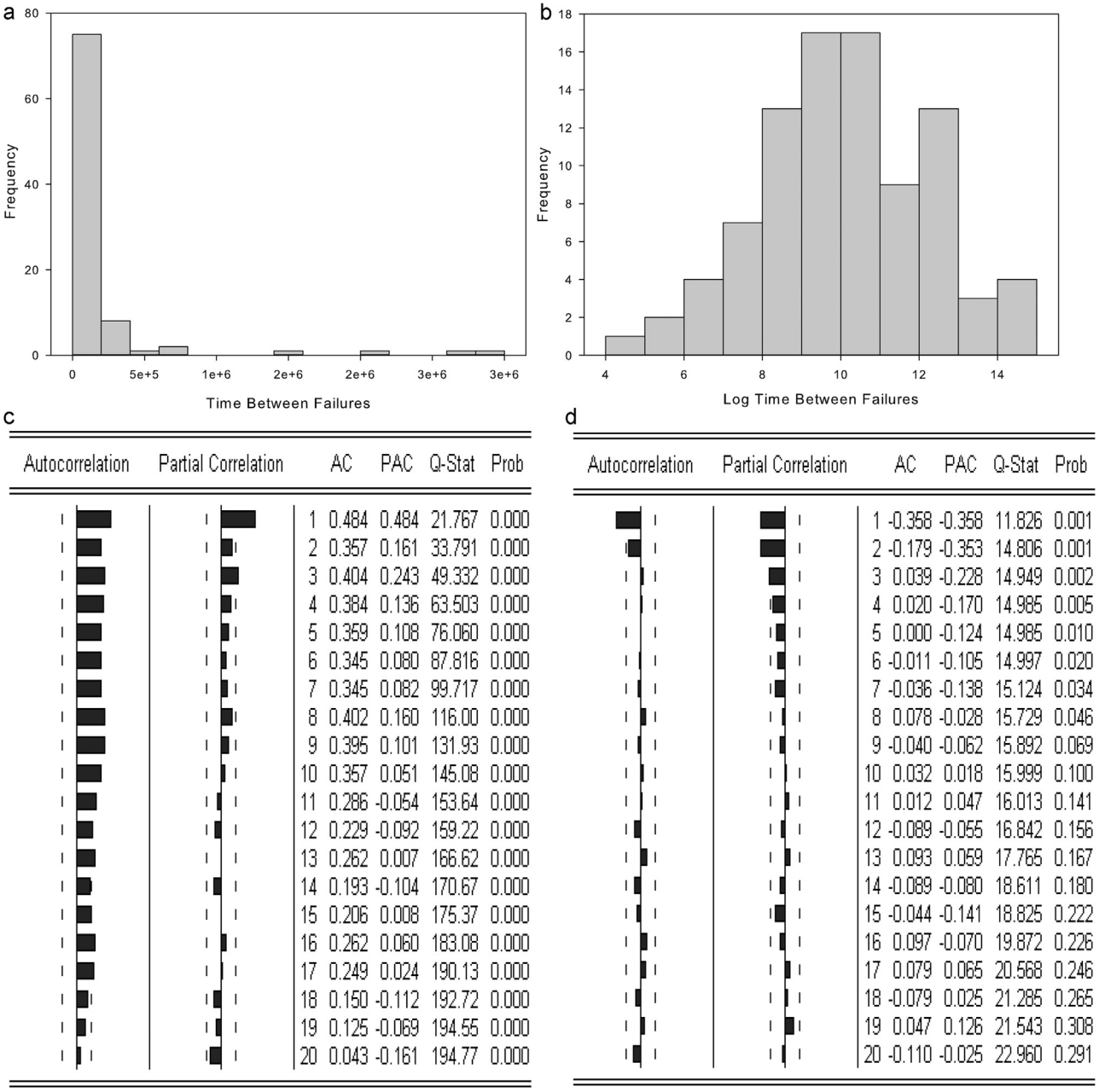

Fig. 2. Histograms, ACFs and PACFs of time between failures of system 40 data. (a) Histogram of time between failures data (b) Histogram of logged time between failures data. (c) ACF and PACF of logged time between failures data and (d) ACF and PACF of differenced logged time between failures data.

Table 1
The identified ARIMA models estimation.

Model	Parameter	Estimate	Std. error	*p*-value
I	AR(1)	−0.6964	0.0074	0.0000
II	MA(1)	−0.8426	0.0059	0.0000
III	AR(1)	0.0762	0.0204	0.0011
	MA(1)	−0.8582	0.0035	0.0000
IV	MA(1)	−0.9365	0.0198	0.0000
	MA(2)	−0.0621	0.0212	0.0054
	AR(1)	0.2986	0.0953	0.0013
V	MA(1)	−1.1302	0.0964	0.0000
	MA(2)	0.3122	0.0953	0.0170

4.4.1. (P4.1) Estimates significance test

Each estimate is tested to check whether it is statistically significant using a *t*-test:

$$t = \frac{\text{estimate value}}{\text{standard error of estimate}} \tag{4}$$

If the calculated *t* value for an estimate is significantly greater than a predetermined critical value (=1.96, for significance level = 0.05), it is significant and retained in the model. Otherwise, the model requires a recalculation using the remaining terms.

4.4.2. (P4.2) Invertibility and stationarity conditions satisfaction

In the model checking phase, invertibility and stationarity conditions can be checked as follows: (1) stationarity condition: the sum of the coefficients of AR model should be less than one, which means that $\sum_{i=1}^{p}\phi_i < 1$; and (2) invertibility condition: the sum of the coefficients of MA model should be less than one, which means that $\sum_{i=1}^{p}\theta_i < 1$.

4.4.3. (P4.3) Residuals randomness

Residuals of the well-fitted model will be random and approximately follow the normal distribution. For this condition, residuals can be analysed by plotting their ACF and PACF to observe whether these functions are statistically equal to zero. In addition, a hypothesis test, i.e. the Box–Pierce test (Box and Pierce, 1970), is performed to make statistically significant decision regarding this condition.

Example 4. The approach computed the *t*-test for all the estimates and its *p*-values are depicted in Table 1. It is clear that all the models' estimates are significant. Also, from Table 1 it is evident that all the models, except the last model, satisfy invertibility and stationarity conditions. To analyse the residuals, the approach has used Box–Pierce test (Box and Pierce, 1970) and concluded that all the models' residuals are uncorrelated except the last model's residuals, where the *p*-value equals to 0.035 (<0.05).

4.5. (P5) Best model selection

Once the identified models as a combination have been estimated and checked, the best model is selected based on Akaike's information criterion (AIC) (Akaike, 1974):

$$\text{AIC} = 2k - 2ln(L) \tag{5}$$

where *k* is the number of parameters in the estimated model, and *L* is the maximized value of the likelihood function for the estimated model. The best model is the one that has the minimum AIC value.

Example 5. Based on the diagnostics checking in P4, the first four estimated models are adequate to fit and forecast the future values of dlsys40. The AIC values for these models are: 15.2151, 15.1852, 15.1998, and 15.2054, respectively. Accordingly, the second model, ARIMA(0, 1, 1), is the best model that can be used to predict the future values of System 40 data.

4.6. (P6) Model based prediction

After selecting the best model, it is ready to be used for forecasting the future values of the given data. For more illustration, ARIMA model (2) after the estimation can be written as follows:

$$\hat{y}_t = \sum_{i=1}^{p}\hat{\phi}_i y_{t-i} + \sum_{i=1}^{q}\hat{\theta}_i\hat{\varepsilon}_{t-i} + \hat{\varepsilon}_t \tag{6}$$

To predict one-step-ahead values, it is moved from (t) to $(t+1)$:

$$\hat{y}_{t+1} = \sum_{i=1}^{p}\hat{\phi}_i y_{t+1-i} + \sum_{i=1}^{q}\hat{\theta}_i\hat{\varepsilon}_{t+1-i} + \hat{\varepsilon}_{t+1} \tag{7}$$

and similarly it can be moved to predict multi-step-ahead values.

Example 6. The best model selected in P5 can be rewritten as follows:

$$\hat{z}_t = z_{t-1} - (0.84263)\varepsilon_{t-1} + \varepsilon_t \tag{8}$$

Using this model, the one-step ahead predictions of the last eleven observations are given in Table 2.

5. Evaluation

In this section, we discuss how our proposed approach addresses the critical limitations discussed in Section 2 and outperforms the SRGMs and the existing time series based approaches.

5.1. Experiment setup

We identify the main research questions of our experiments as follows:

*RQ*1: Does the proposed approach outperform traditional SRGMs and improve the prediction accuracy?

*RQ*2: Does the proposed approach outperform existing time series and machine learning based approaches and improve the prediction accuracy?

In order to address *RQ*1, we apply our proposed prediction approach and the Jelinski–Moranda (JM) model to a benchmark of time between failures (TBF) data sets of sixteen real-world software systems (Musa, 1980). The list, description, and number of failures of these software systems are reported in Table 3.

After that, for each data set prediction we compute the mean squared errors (MSE) measure as follows:

$$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n}[y_t - \hat{y}_t]^2, \tag{9}$$

where y_t and $\hat{y}_t$ are the real and predicted values, respectively. We analyse these errors using boxplots and descriptive statistics. To test whether the proposed approach significantly outperforms the JM model in terms of prediction accuracy, we use the non-parametric Mann-Whitney test (Gibbons and Chakraborti, 2003). Additionally, we measure the relative prediction accuracy improvement using a metric RAI_{MSE} which is defined as:

$$\text{RAI}_{\text{MSE}} = \frac{\text{MSE}_{\text{JM}} - \text{MSE}_{\text{Ours}}}{\text{MSE}_{\text{JM}}} \times 100, \tag{10}$$

where MSE_{Ours} and MSE_{JM} refer to the MSE values produced by our proposed approach and JM model, respectively.

To address *RQ2*, we review the existing time series based and machine learning based approaches and classify them into four classes:

Table 2
Predictions of the logged time between failures.

Failure number	Actual value	Predicted value	Failure number	Actual value	Predicted value
91	4.937	4.395	97	5.552	4.585
92	4.948	5.157	98	5.715	4.281
93	6.259	3.646	99	5.539	5.011
94	3.619	4.027	100	4.496	3.501
95	3.505	4.501	101	5.424	4.944
96	5.299	4.472			

Table 3
Software reliability data project information.

System code	Application	Size (delivered object code instructions)	Number of failures
1	Real time command &control	21,700	136
2	Real time command &control	27,700	54
3	Real time command &control	23,400	38
4	Real time command &control	33,500	53
5	Real time commercial	2,445,000	831
6	Commercial subsystem	5,700	73
14C	Real time	Hundreds of thousands	36
17	Military	61,900	38
27	Military	126,100	41
40	Military	180,000	101
SS1A	Operating system	Hundreds of thousands	112
SS1B	Operating system	Hundreds of thousands	375
SS1C	Operating system	Hundreds of thousands	277
SS2	Time sharing system	Hundreds of thousands	192
SS3	Word processing system	Hundreds of thousands	278
SS4	Operating system	Hundreds of thousands	196

1. Random coefficient autoregressive process based approach, i.e. Singpurwalla and Soyer (1985).
2. Autoregressive process based approach, i.e. Chatterjee et al. (1997).
3. ARIMA process based approach, i.e. Xie and Ho (1999).
4. Support vector regression based approach, i.e. Moura et al. (2011).

After that, we apply the proposed approach to the same real-life TBF data sets that are used in the original articles of these existing approaches and compute the predictions. Based on the prediction results of the aforementioned approaches, we compute the MSE and RAI_{MSE} metrics.

5.2. Results

5.2.1. RQ1. Improving prediction accuracy compared to traditional SRGMs

Boxplots of relative prediction errors are depicted in Fig. 3, while MSE values of the proposed approach and the JM model are reported separately for each system in Table 4. The boxplots highlight that the proposed approach can provide more accurate predictions than the JM model, where the prediction errors exhibited by our approach are significantly lower than those of the JM model. This result is confirmed by the Mann–Whitney test where the p-value < 0.05. More precisely, as seen from Table 4, the proposed approach produces MSE = 3.00 comparing to 3.93 produced by the JM model. This is a relative accuracy improvement (RAI) of about 22.2%.

The Boxplots show another observation that the variation of prediction errors exhibited by the proposed approach is significantly lower than that of JM model. Where, 50% of the prediction errors are within (4.21, 12.29) with range = 8.08 for the proposed approach, and within (4.77, 16.83) with range = 12.05 for the JM model. This implies that the proposed approach's prediction accuracy is more stable across different data sets than that of the JM model.

For each system, it is clear from Table 4 that the proposed approach outperforms the JM model in all the cases. This result is also confirmed by the Mann–Whitney test, where all the p-values < 0.05. The lower and higher accuracy improvement of the approach comparing to the JM model are in the cases of system "SS3" with RAI_{MSE} is about 6.32% and system "5" with RAI_{MSE} is about 66.25%, respectively.

5.2.2. RQ2. Improving prediction accuracy compared to existing time series and machine learning based approaches

In the following we compare our approach to the existing time series and machine learning based approaches.

(1) *Our approach vs. random coefficient autoregressive process based approach*. Singpurwalla and Soyer (1985) have proposed an

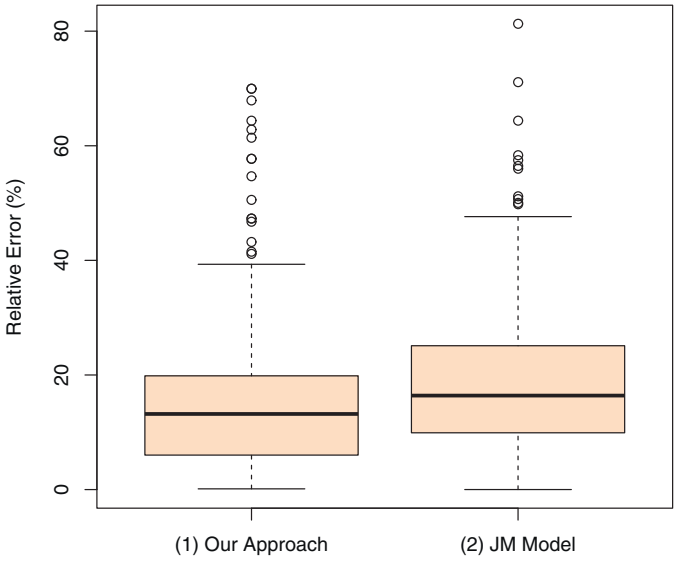

Fig. 3. Boxplots of relative prediction errors.

Table 4
MSE and RAI_{MSE} values for our approach vs. JM model.

System code	MSE values		RAI_{MSE} values
	Our approach	JM model	
1	1.65	2.04	19.22
2	0.93	1.16	20.43
3	1.71	2.32	26.42
4	1.08	1.56	30.62
5	2.64	7.84	66.25
6	2.38	2.87	17.13
14C	2.26	3.86	41.55
17	1.31	1.65	20.96
27	2.24	2.64	14.98
40	2.71	3.67	26.16
SS1A	3.34	3.58	6.59
SS1B	3.18	4.52	29.59
SS1C	3.78	4.90	22.97
SS2	4.74	5.56	14.77
SS3	8.10	8.64	6.32
SS4	5.87	6.79	13.51
Average	3.00	3.93	22.16

autoregressive model to fit and predict time between failures as a metric of reliability growth. Mainly, they assumed that the parameters (or coefficients) of these models are random, which means they have different values at each point of time. They evaluated the accuracy of this model using System 40 data (Musa, 1980), and their forecasting results are available in Singpurwalla and Soyer (1985). (It is worth noting that they applied their approach to the transformed, not original, data set using natural logarithm).

We have applied our approach to the same data set as explained in detail in our running example, and the prediction results of the last eleven observations are depicted in Table 2. In addition, we have computed the MSE measure for the prediction results. The MSE values are 2.71 produced by our approach comparing to 5.03 produced by Singpurwalla and Soyer's approach, which indicates that our approach gives more accurate predictions than their approach with relative accuracy improvement is about 46.1%.

(2) *Our approach vs. autoregressive process based approach.* Chatterjee et al. (1997) applied autoregressive models to predict the future values of software reliability. They evaluated their approach using two real life data sets of System 5 and System 40 (Musa, 1980). Therein, they propose the following two autoregressive models:

$$\hat{z}_t = (0.38094)z_{t-1} + (0.25714)z_{t-3} + \varepsilon_t \quad (11)$$

$$\hat{z}_t = (0.12988)z_{t-3} + (0.12098)z_{t-6} + (0.10243)z_{t-13} + (0.13355)z_{t-17} + \varepsilon_t \quad (12)$$

to predict the future values of the System 5 and System 40 data sets, respectively. (It is worth noting that they applied their approach to the transformed, not original, System 40 data set using the base-10 logarithm). We have used these two autoregressive models to re-produce Chatterjee et al.'s predictions and applied our approach to the same data sets to compute our predictions. We have computed the MSE measure for Chatterjee et al.'s and our approach's prediction results, as given in Table 5. It is evident that our approach gives more accurate predictions than their approach with relative accuracy improvement is on average about 20%.

Table 5
MSE and RAI_{MSE} values for our approach vs. Chatterjee et al.'s approach.

System code	MSE values		RAI_{MSE} values
	Our approach	Chatterjee et al.'s	
5	2.64	3.18	16.98
40	2.53	3.28	22.87
Average	2.59	3.23	19.93

Table 6
Fitted ARIMA models for the four data sets by Xie and Ho Xie and Ho (1999).

Data set	Model
1	$\hat{z}_t = z_{t-1} + \varepsilon_t - (0.96)\varepsilon_{t-1}$
2	$\hat{z}_t = z_{t-1} + \varepsilon_t - (1.16)\varepsilon_{t-1} + 25.3$
3	$\hat{z}_t = (1.27)z_{t-1} + (0.47)z_{t-2} - (0.73)z_{t-3} + \varepsilon_t$
4	$\hat{z}_t = (0.95)z_{t-1} + (0.05)z_{t-2} + \varepsilon_t - (0.087)\varepsilon_{t-1}$

Table 7
MSE and RAI_{MSE} values for our approach vs. Xie & Ho's approach.

Data set	MSE values		RAI_{MSE} values
	Our approach	Xie and Ho's	
1	257.99	320.61	19.53
2	105712.90	395646.00	73.28
3	24.10	230.39	89.54
4	22.27	70.84	68.56
Average	26504.32	99066.96	62.73

(3) *Our approach vs. ARIMA process based approach.* Xie and Ho (1999) have applied ARIMA models to predict the future values of software failures. They evaluated their approach using four real-life data sets (Xie and Ho, 1999). To predict the future values of the four data sets, they have proposed four ARIMA models, which are depicted in Table 6.

Using the proposed ARIMA models by Xie and Ho and our approach, we compute predictions for the four data sets and their MSE values, as depicted in Table 7. We can see from Table 7 that our approach gives on average MSE value less than Xie and Ho's by about 62.7%, which implies that our approach provides much more accurate predictions. Moreover, it is worth noting that Xie and Ho's models do not satisfy the stationarity and invertibility conditions, e.g. for data set 2 the absolute parameter estimate value (=1.16) is greater than one.

(4) *Our approach vs. support vector regression based approach.* Moura et al. (2011) have proposed a support vector regression (SVR) method for reliability prediction. This SVR method is a kind of learning method that builds its approximation function based on underlying concepts that rise from statistics theory as well as on a set of input/output examples that come up from the process under analysis (Moura et al., 2011). While this method can provide more accurate predictions than the other existing approaches, it requires relatively a large training data set as a set of input/output examples and consumes time to get correct learning and approximation function of the given process. This is basically the main disadvantage of most of learning methods (Pachowicz and Bala, 1991).

Moura et al. have evaluated their approach using two real-life reliability data sets, which are time between failures for turbochargers in diesel engines and miles-between-failures for a car engine. Therein, they compared their approach with the existing learning methods including ARIMA model as statistical learning method, and they concluded that their approach outperforms ARIMA model in term of prediction accuracy. The MSE values for their prediction results using SVR are presented in Table 8.

Table 8
MSE and RAI_{MSE} values for our approach vs. Moura et al.'s approach.

Data set	MSE values		RAI_{MSE} values
	Our approach	Moura et al.'s	
1	0.0079	0.0094	14.84
2	0.280	0.222	−26.13
Average	0.144	0.116	−5.65

We have applied our approach to compute the predictions for these two data sets, and the MSE values are reported in Table 8. From this Table we can conclude generally that our approach has comparable prediction accuracy to the SVR approach, whereas Moura et al. Moura et al. (2011) concluded in their work that SVR outperforms ARIMA models. In particular, in the case of data set 1 our approach outperforms the SVR approach, and our justification is that this data set consists of only 40 observations which is insufficient as a training set for the SVR approach. In contrast, the SVR approach outperforms our approach in the case of data set 2, which consists of 100 observations. Consequently, it is worth noting that the required training data set and consumed time for our approach to construct an adequate prediction model for the process under analysis is dramatically less than that required by SVR methods. This is because our approach's methodology is based on standard statistics theory and probability distributions that can be straightforwardly implemented without consuming time in learning.

6. Conclusion

Among software quality attributes, reliability is generally considered to be the most important factor because it quantifies software faults and failures, which can lead to serious consequences in software systems. To estimate and predict this reliability, Software Reliability Growth Models (SRGMs) are widely used. However, these models have many shortcomings related to their unrealistic assumptions, environment-dependent applicability, and questionable predictability. In the literature, researchers have suggested many solutions to address these shortcomings based on different methodologies, such as non-parametric statistics, neural networks, and Bayesian networks. However, none of these solutions addresses all the SRGMs' shortcomings.

In this paper we proposed a well-established prediction approach based on time series ARIMA models as an alternative solution to address these SRGMs' shortcomings and the limitations of the existing approaches, and thereby provide more accurate reliability prediction. The main advantage of this approach is that it is a data-oriented approach and does not impose any restrictive assumptions on the environment of the software system that generates the data under analysis. Using real-life data sets on software failures, the accuracy of proposed approach was evaluated and compared to that of the SRGMs and existing time series based approaches. This comparison showed that our proposed approach performed better than other ARIMA-based approaches, and was compatible in performance and less costly than the SVR approach.

References

Akaike, H., 1974. A new look at the statistical model identification. IEEE Transactions on Automatic Control 19 (6), 716–723.

Bai, C., Hu, Q., Xie, M., Ng, S., 2005. Software failure prediction based on a Markov Bayesian network model. Journal of Systems and Software 74 (3), 275–282.

Barghout, M., Littlewood, B., Abdel-Ghaly, A., 1998. A non-parametric order statistics software reliability model Software Testing. Verification and Reliability 8 (3), 113–132.

Box, G.E.P., Cox, D.R., 1964. An analysis of transformations. Journal of the Royal Statistical Society Series B (Methodological) 26 (2), 211–252.

Box, G., Jenkins, G., 1976. Time Series Analysis: Forecasting and Control. Holden Day, San Francisco.

Box, G.E.P., Pierce, D.A., 1970. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. JASA 65 (332), 1509–1526.

Brocklehurst, S., Littlewood, B., 1992. New ways to get accurate reliability measures [software]. IEEE Software 9 (4), 34–42.

Brocklehurst, S., Chan, P., Littlewood, B., Snell, J., 1990. Recalibrating software reliability models. IEEE Transactions on Software Engineering 16 (4), 458–470.

Chatterjee, S., Misra, R., Alam, S., 1997. Prediction of software reliability using an auto regressive process. International Journal of Systems Science 28 (2), 211–216.

Fenton, N., Neil, M., Marquez, D., 2008. Using Bayesian networks to predict software defects and reliability. Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability 222 (4), 701–712.

Gibbons, J.D., Chakraborti, S., 2003. Nonparametric Statistical Inference. CRC.

Goel, A., Okumoto, K., 1979. A time-dependent error-detection rate model for software reliability and other performance measures. IEEE Transactions on Reliability 28 (3), 206–211.

Goel, A., 1985. Software reliability models: assumptions, limitations, and applicability. IEEE Transactions on Software Engineering 11, 12.

Ho, S., Xie, M., 1998. The use of ARIMA models for reliability forecasting and analysis. Computers and Industrial Engineering 35 (1-2), 213–216.

Huang, C., Lyu, M., 2011. Estimation and analysis of some generalized multiple change-point software reliability models. IEEE Transactions on Reliability 60 (2), 498–514.

Jelinski, Z., Moranda, P., 1972. Software reliability research. In: Statistical Computer Performance Evaluation. Academic Press Inc, New York, USA, pp. 465–484.

Junhong, G., Hongwei, L., Xiaozong, Y., 2005. An autoregressive time series software reliability growth model with independent increment. In: Proceedings of the International Conference on Mathematical Methods and Computational Techniques In Electrical Engineering, WSEAS, pp. 362–366.

Kapur, P., Pham, H., Anand, S., Yadav, K., 2011. A unified approach for developing software reliability growth models in the presence of imperfect debugging and error generation. IEEE Transactions on Reliability 60 (1), 331–340.

Karunanithi, N., Whitley, D., Malaiya, Y., 1992. Using Neural Networks in Reliability Prediction. IEEE Software 9 (4), 59.

Kiran, N., Ravi, V.,2007. Software reliability prediction using wavelet neural networks. In: iccima. IEEE Computer Society, pp. 195–199.

Kwiatkowski, D., Phillips, P., Schmidt, P., Shin, Y., 1992. Testing the null hypothesis of stationarity against the alternative of a unit root. Journal of Econometrics 54, 159–178.

H. Li, M. Zeng, M. Lu, Exploring AdaBoosting Algorithm for Combining Software Reliability Models, ISSRE.

Lo, J., 2009. The implementation of artificial neural networks applying to software reliability modeling. In: Proceedings of the 21st annual international conference on Chinese control and decision. IEEE Press, China4385–4390.

Lyu, M., Nikora, A., 1992. Applying reliability models more effectively. IEEE Software 9 (4), 43–52.

Lyu, M., 1996. Handbook of Software Reliability Engineering. McGraw-Hill, Inc., Hightstown, NJ, USA.

Lyu, M.R., 2007. Software reliability engineering: a roadmap. In: Future of Software Engineering (FOSE'07), pp. 153–170.

Makridakis, S., Wheelwright, S., Hyndman, R., 2008. Forecasting Methods and Applications. Wiley-India.

Moura, M., Zio, E., Didier Lins, I., Droguett, E., 2011. Failure and reliability prediction by support vector machines regression of time series data. Reliability Engineering and System Safety 96 (11), 1527–1534.

Musa, J., 1980. Software reliability data. Data and Analysis Center for Software, Griffis Air Force Base, NY.

Neil, M., Fenton, N., 1996. Predicting software quality using Bayesian belief networks. In: Proceedings of 21st Annual Software Engineering Workshop, Citeseer.

Pachowicz, P., Bala, J., 1991. Texture recognition through machine learning and concept optimization. In: Tech. Rep. MLI 91-4, Machine Learning and Inference Laboratory, George Mason University, Fairfax, VA.

Pai, P., Hong, W., 2006. Software reliability forecasting by support vector machines with simulated annealing algorithms. Journal of Systems and Software 79 (6), 747–755.

Palviainen, M., Evesti, A., Ovaska, E., 2011. The reliability estimation prediction and measuring of component-based software. Journal of Systems and Software 84 (6), 1054–1070.

Popeniu, F., Boro, D., 1996. Software reliability growth supermodels. Microelectronics and Reliability 36 (4), 485–491.

Raj Kiran, N., Ravi, V., 2008. Software reliability prediction by soft computing techniques. Journal of Systems and Software 81 (4), 576–583.

Robinson, D., Dietrich, D., 1987. A new nonparametric growth model. IEEE Transactions on Reliability 36 (4), 411–418.

Sharma, K., Garg, R., Nagpal, C., Garg, R., 2010. Selection of optimal software reliability growth models using a distance based approach. IEEE Transactions on Reliability 59 (2), 266–276.

Singpurwalla, N., Soyer, R., 1985. Assessing (Software) reliability growth using a random coefficient autoregressive process and its ramifications. IEEE Transactions on Software Engineering 11 (12), 1456–1464.

Wiper, M., Palacios, A., Marí n, J., 2012. Bayesian software reliability prediction using software metrics information. Quality Technology and Quantitative Management 9 (1), 35–44.

Wood, A., 1997. Software reliability growth models assumptions vs. reality. In: issre. IEEE Computer Society, p. 136.

Xie, M., Ho, S., 1999. Analysis of repairable system failure data using time series models. Journal of Quality in Maintenance Engineering 5 (1), 50–61.

Xie, M., Hong, G., Wohlin, C., 1997. A study of the exponential smoothing technique in software reliability growth prediction. Quality and Reliability Engineering International 13 (6), 347–353.

Yang, B., Li, X., Xie, M., Tan, F., 2010. A generic data-driven software reliability model with model mining technique. Reliability Engineering and System Safety 95 (6), 671–678.

Zaidi, S., Danial, S., Usmani, B., 2008. Modeling inter-failure time series using neural networks. In: IEEE International Multitopic Conference, pp. 409–411.

Zeitler, D., 1991. Realistic assumptions for software reliability models. In: IEEE International Symposium of Software Reliability Engineering, Citeseer.

Ayman Amin is currently a PhD student in the Faculty of Information and Communication Technologies at Swinburne University of Technology, Australia. Amin's research area is statistical monitoring and forecasting for quality of service (QoS) attributes of software systems. Amin received his BSC in Statistics and Social Science Computing (in June 2006) and MSC in Statistics (in October 2009) from Faculty of Economic and Political Science (FEPS), Cairo University, Egypt.

Lars Grunske is currently Professor at the University of Stuttgart, Germany. He received his PhD degree in computer science from the University of Potsdam (Hasso-Plattner-Institute for Software Systems Engineering) in 2004. He was Junior Professor at the University of Kaiserslautern, Boeing Postdoctoral Research Fellow at the University of Queensland from 2004–2007 and lecturer at the Swinburne University of Technology, Australia from 2008–2011. He has active research interests in the areas modelling and verification of systems and software. His main focus is on automated analysis, mainly probabilistic and timed model checking and model-based dependability evaluation of complex software intensive systems.

Alan Colman is a Senior Lecturer in the Faculty of Information and Communication Technologies at the Swinburne University of Technology in Melbourne, Australia. He is a member of the Centre for Computing and Engineering Software Systems where his research primarily focuses on the customisation, management and control of adaptable service-based systems. He has a Masters in Human-Computer Interaction and a PhD in Software Engineering from Swinburne University of Technology.