A comparative evaluation of using genetic programming for predicting fault count data

Wasif Afzal, Richard Torkar
Blekinge Institute of Technology,
S-372 25 Ronneby, Sweden
{waf,rto}@bth.se

Abstract

There have been a number of software reliability growth models (SRGMs) proposed in literature. Due to several reasons, such as violation of models' assumptions and complexity of models, the practitioners face difficulties in knowing which models to apply in practice. This paper presents a comparative evaluation of traditional models and use of genetic programming (GP) for modeling software reliability growth based on weekly fault count data of three different industrial projects. The motivation of using a GP approach is its ability to evolve a model based entirely on prior data without the need of making underlying assumptions. The results show the strengths of using GP for predicting fault count data.

1. Introduction

A key element of software quality is software reliability, defined as the ability of a system or component to perform its required functions under stated conditions for a specific period of time [13]. If the software frequently fails to perform according to user-specified behavior, other software quality factors matter less [19].

Software reliability growth modeling helps in deciding project release time and managing project resources. After the first software reliability growth model was proposed by Jelinski and Moranda in 1972 [14], there have been numerous reliability growth models following it. The existence of a large number of models requires a user to select and apply an appropriate model. For practitioners, this may be an unmanageable selection problem and there is a risk that the selected model is unsuitable to the particulars of the project in question. Some models are complex with many parameters. Without extensive mathematical background, practitioners cannot determine when it is applicable and when the model diverges from reality. Moreover, these *parametric* software reliability growth models are often characterized by a number of assumptions [9] which are necessary to develop a mathematical model. These assumptions are often violated in real-world situations (see e.g. [26]), therefore, causing problems in the long-term applicability and validity of these models. Even if the dynamics of the testing process are well known, there is no guarantee that the model whose assumptions appear to best suit these dynamics will be most appropriate [20].

Under this scenario, what becomes significantly interesting is to have modeling mechanisms that can exclude the pre-suppositions about the model and are based entirely on the fault data. In this respect, genetic programming (GP) could be used as an effective tool because, being a *non-parametric* method, GP does not conceive a particular structure for the resulting model and GP also does not take any assumptions about the distribution of the data.

In an earlier study, we investigated the suitability of using GP for building software reliability growth models [2]. In this paper, we present the results of comparison between models evolved using GP and three other traditional SRGMs based on weekly fault count data of three projects carried out by a large telecommunication company. We compare the models using measures of model validity, goodness of fit and residual analysis. The comparative results indicates that in terms of model validity, two out of three measures favored GP evolved models. The GP evolved model also represented comparatively better goodness of fit, while residual analysis showed that the predictions from the GP evolved model are comparatively less biased.

The remainder of this paper is organized as follows. Section 2 describes related work. Section 3 presents a brief introduction to genetic programming. In Section 4 we detail our research method along with a discussion of evaluation measures used. Sections 5 and 6 are comprised of experimental setup and results respectively. Validity evaluation is given in Section 7 while the discussion and future work appears in Section 8.

978-0-7695-3372-8/08 $25.00 © 2008 IEEE
DOI 10.1109/ICSEA.2008.9

2 Related work

Studies reporting the use of GP for software reliability modeling are few and recent. Costa et al. [8] presented the results of two experiments exploring GP models based on time and test coverage. The authors compared the results with other traditional and non-parametric artificial neural network (ANN) models. The results from the first experiment, which explored GP models based on time-between-failure (TBF) data, showed that GP adjusts better to the reliability growth curve. The results from the second experiment, which was based on test coverage data, showed that all metrics were always better for GP and ANN models. The authors later extended GP with boosting techniques for reliability growth modeling [21] and reported improved results. A similar study by Zhang and Chen [28] used GP to establish a software reliability model based on mean time between failures (MTBF) time series. The results of the study also confirmed that in comparison with the ANN model and traditional models, the model evolved by GP had higher prediction precision and better applicability.

There are several ways in which the present work differs from the aforementioned studies. Firstly, none of the previous studies used data sets consisting of weekly fault count data. In this study, our aim is to use the weekly fault count data as a means to evolve the reliability growth model using GP and perform comparisons with traditional reliability growth models. Secondly, we have avoided performing any pre-processing of data to avoid chances of incorporating bias. Thirdly, we remain consistent with using first 2/3 of the data to build the model and use the rest 1/3 of the data for model evaluation for all of our data sets. Lastly, in an attempt to provide a fair evaluation, we also remain consistent with using the same set of evaluation measures to compare GP evolved models with traditional models for all the data sets.

3 Background to genetic programming

The evolution of software reliability growth models using GP is an example of a symbolic regression problem. Symbolic regression is an error-driven evolution as it aims to find a function, in symbolic form, that fits (or approximately fits) data from an unknown curve [16]. In simpler terms, symbolic regression finds a function whose output matches some target values. GP is well-suited for symbolic regression problems as it does not make any assumptions about the structure of the function.

GP is an evolutionary computation technique (first results reported by Smith [24] in 1980) and is an extension of genetic algorithms. As compared with genetic algorithms, the population structures (individuals) in GP are not fixed length character strings, but programs that, when executed, are the candidate solutions to the problem. GP is a systematic, domain-independent method for getting computers to solve problems automatically starting from a high-level statement of what needs to be done [23]. Programs are expressed in GP as syntax trees, with the nodes indicating the instructions to execute and are called functions (e.g. min, $*$, $+$, $/$), while the tree leaves are called terminals which may consist of independent variables of the problem and random constants (e.g. x, y, 3). The fitness evaluation of a particular individual is determined by the correctness of the logical output produced for all of the fitness cases [3]. The control parameters limit and control how the search is performed like setting the population size and probabilities of performing the genetic operations. The termination criterion specifies the ending condition for the GP run and typically includes a maximum number of generations [7]. GP iteratively transforms a population of computer programs into a new generation of programs using various genetic operators. Typical operators include crossover, mutation and reproduction.

4 Research method

In this section we outline the research method used in the paper. We describe the data sets used, selection of traditional SRGMs for comparison, the formulated hypotheses and a description of the evaluation measures.

4.1 Fault count data sets

The data sets used in this study are based on the weekly fault count data collected during the testing of three large-scale software projects at a large telecom company. The motivation for selecting the fault count data from an industrial context is to be representative of real-world problem domain. The projects are targeted towards releases of three mature systems that have been on the market for several years. These projects followed an iterative development process. In this scenario, it becomes important for project managers to estimate the current reliability and to predict the reliability ahead of time, so as to measure the quality impact with continuous addition of new functionality and fixes of previously discovered faults. Appendix A shows the fault count data sets used in the study, but due to the proprietary nature of data, the number of faults are multiplied by a constant factor and are given for illustrative purposes only. Nevertheless, we believe that making the data sets available allows the research community to replicate results and to perform additional studies. The *results* of the evaluation measurements in the rest of the paper are, however, based on original data sets.

The fault count data from the three projects is divided into two portions. The first portion comprises of first $2/3$ data from each data set and is used for calibrating the software reliability model under consideration. The second portion comprises of later $1/3$ data from each data set and is used for comparing the predictions from each individual model according to the five different statistics (Subsection 4.4). This implies that we are able to make predictions on several weeks constituting $1/3$ of the data.

4.2 Selection of traditional SRGMs

Since we are interested in comparing predictions of weekly fault count data, therefore we selected three traditional SRGMs that represent the fault count family of models [9]. These three models are Goel-Okumoto non-homogeneous poisson process model (GO-NHPP) [10], Brooks and Motley's poisson model (BM) [5] and Yamada's S-Shaped growth model (YAM) [27]. We selected them because these models present a fair representation of fault count family of models and represents different forms of growth curves. In particular, GO-NHPP and BM are concave (or exponential) while YAM is S-shaped. Also we had limitations in terms of information requirements of certain models, so they were not selected for comparison, like Shooman exponential model's hazard function requires knowing the parameters of total number of instructions in the program and debugging time since the start of system integration [9].

4.3 Hypothesis

In order to formalize the purpose of this experiment, we define the following hypotheses:

$\mathrm{H}_{0-\mathrm{val}}$: The predictions of the GP evolved model are not significantly more valid as compared with traditional models.

$\mathrm{H}_{1-\mathrm{val}}$: The predictions of the GP evolved model are significantly more valid as compared with traditional models.

$\mathrm{H}_{0-\mathrm{gof}}$: The GP evolved model does not give significantly higher goodness of fit as compared with traditional models.

$\mathrm{H}_{1-\mathrm{gof}}$: The GP evolved model gives significantly higher goodness of fit as compared with traditional models.

$\mathrm{H}_{0-\mathrm{res}}$: There is no significant difference between the residuals of the GP evolved model as compared with traditional models.

$\mathrm{H}_{1-\mathrm{res}}$: There is a significant difference between the residuals of the GP evolved model as compared with traditional models.

In order to test the above hypotheses, we use different evaluation measures as detailed in the next section.

4.4 Evaluation measures

It is usually recommended to use more than one measure to determine model applicability, as in [20], because reliance on a single measure can lead to making incorrect choices. We used measures of model validity, model goodness of fit and distribution of residuals to compare GP evolved model with traditional reliability growth models.

Model validity is measured in terms of prequential likelihood ratio (PLR), the Braun statistic and the adjusted mean square error (AMSE). The PLR of two prediction systems, A and B, is the running product of ratio of their successive on-step ahead predictions $\hat{f}_j^A(t_j)$ and $\hat{f}_j^B(t_j)$ respectively [4]:

$$PLR_i^{AB} = \prod_{j=s}^{j=i} \frac{\hat{f}_j^A(t_j)}{\hat{f}_j^B(t_j)}$$

In our case, we select the actual time distribution of weekly fault count data as a reference and conduct pairwise comparisons of all other models' predictions against it. Then the model with the relatively smallest prequential likelihood ratio can be expected to provide the most trust worthy predictions. For further details on PLR, see [1, 4]. We complement the measure of prequential likelihood ratio with two measures of variability, namely the Braun statistic and AMSE. The Braun statistic can be used to measure the accuracy of fault count predictions and is give by the following formula [4]:

$$\text{Braun statistic}\{\hat{E}[N_k]; k = s, \ldots, r\} = \frac{\displaystyle\sum_{k=s}^{r}(n_k - \hat{E}[N_k])^2 x_k}{\displaystyle\sum_{k=s}^{r}(n_k - \bar{n})^2 x_k}$$

Where n_k is the actual fault count within successive time intervals, $x_k, k = s, \ldots, r$. $\hat{E}[N_k]$ represents the predicted fault count data and $\bar{n}$ represents the mean of the actual fault count data. AMSE is a simple measure based on the mean square error which takes into account the mean of the data sets and is given by the following formula [6]:

$$AMSE = \sum_{i=1}^{i=n} \frac{(E_i - \hat{E}_i)^2}{(\bar{E}_i) * \bar{\hat{E}}_i)^2}$$

Where E_i is the actual fault count data and $\hat{E}_i$ is the predicted fault count data.

To measure a particular model's bias, we examine the *distribution of residuals* to compare models as suggested in [15, 22]. The model's *goodness of fit* in our case was measured using Kolmogorov-Smirnov (K-S) test [12]. For K-S test, we used the significance level, $\alpha = 0.05$ and if the K-S statistic J is greater or equal than the critical value J_α, the null hypothesis of two samples having the same probability distribution is rejected in favor of the alternate hypothesis.

5 Experimental setup

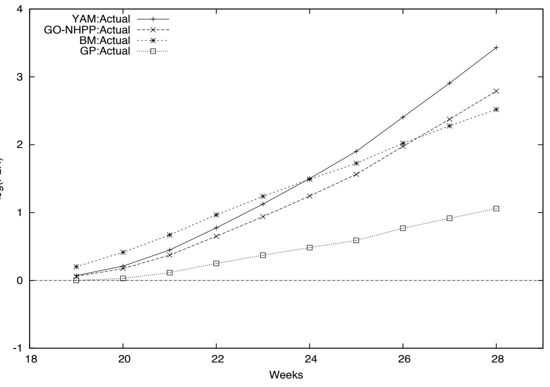
In this study we used MATLAB version 7.0 [18] and GPLAB version 3.0 [11] (a GP toolbox for MATLAB).

GPLAB allows for different choices of tuning control parameters. Initially we experimented with a minimal set of functions by keeping the terminal set containing the independent variable only. We incrementally increased the function set with additional functions and later on also complemented the terminal set with a random constant. For each data set, the best model having the best fitness was chosen from all the runs of the GP system with different variations of function and terminal sets. The function set for project 1 and project 3 data sets were the same, while a slightly different function set for project 2 gave the best fitness. The GP programs were evaluated according to the sum of absolute differences between the obtained and expected results in all fitness cases, $\sum_{i=1}^{n} \mid e_i - e_i^{'} \mid$, where e_i is the actual fault count data, $e_i^{'}$ is the estimated value of the fault count data and n is the size of the data set used to train the GP models. The control parameters that were chosen for the GP system are shown in Table 1.

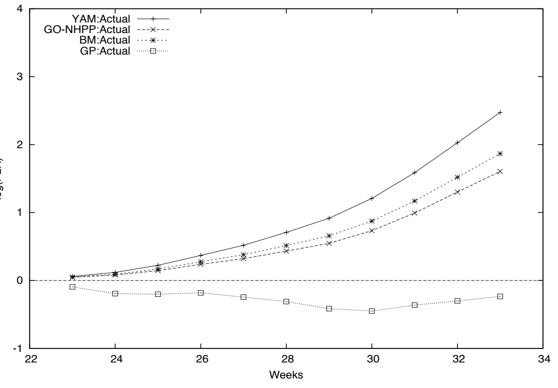
6 Results

Figure 1 shows the PLR analysis for the three data sets. The log(PLR) of actual time distribution of weekly fault count data is chosen as the the reference; and it is indicated as a straight line in the plots of Figure 1. It can be seen that the curve for the PLR of the GP model with the actual fault count data (GP:Actual) is closer to the straight line as compared with the same curves for the traditional models; confirming that GP predictions are better approaching reality as compared with traditional reliability growth models.

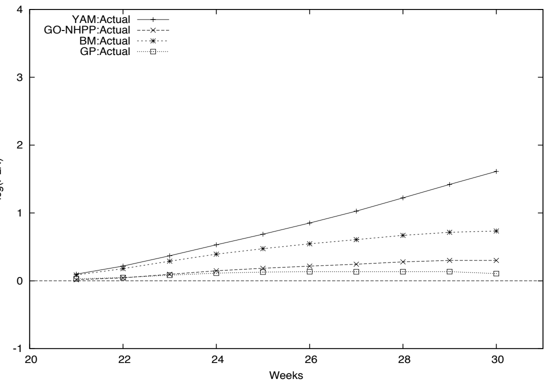
The variability measures of Braun statistic and AMSE obtained for each data set of all models were compared using matched paired two-sided t-test at significance level, α = 0.1. We compared the variability measures of the GP model with each of the traditional models. The null hypothesis was formulated as that there was no difference between the variability statistics of GP and that of the particular traditional model under comparison. The alternate hypothesis



(a) Log(PLR) plots for Project 1.



(b) Log(PLR) plots for Project 2.



(c) Log(PLR) plots for Project 3.

Figure 1. Log(PLR) plots for three projects.

Table 1. GP control parameters.

Control Parameter	*Value*
Population size	30
Number of generations	200
Termination condition	200 generations
Function set (for project 1 & 3)	$\{+, -, *, sin, cos, log\}$
Function set (for project 2)	$\{+, -, *, /, sin, cos, log\}$
Terminal set	$\{x\}$
Tree initialization	ramped half-and-half
Initial maximum number of nodes	28
Maximum number of nodes after genetic operations	512
Genetic operators	crossover, mutation, reproduction
Selection method	lexictour
Elitism	replace

Table 2. Statistical results for Braun statistic and AMSE.

Comparative models	*t-statistic*
Braun statistic, t_α=±2.42	
GP:BM	-3.97
GP:YAM	-4.80
GP:GO-NHPP	-1.64
AMSE statistic, t_α=±2.42	
GP:BM	-1.23
GP:YAM	-1.39
GP:GO-NHPP	-1.03

to test was then that there existed such a difference. Using normal quantile plot of the samples' variability differences to assess any radical departures from normal distribution showed that they had approximately normal distribution. The results of applying the matched paired two sided t-test are shown in Table 2.

The critical values of t for α=0.1 and degrees of freedom $n-1$ is $t_\alpha = \pm 2.92$. If the calculated t-statistic lied in the critical region, we were able to reject the null hypothesis of no difference between the samples.

We can observe from Table 2 that there is a statistical difference between GP and two of the traditional models (BM and YAM) for the Braun statistic. However, for the AMSE statistic, there is no statistical difference between GP and traditional models. This shows that the GP model, while optimizes the Braun statistic, degrades AMSE. This result strengthens the viewpoint of Mair et al. [17] that using a fitness function for GP that is not specifically tied to a single measure but takes into account multiple objectives may give overall better results for the GP model. Based on the results of applying PLR and Braun statistic, we are able to reject the null hypothesis, $H_{0-\mathrm{val}}$ in support of the alternative hypothesis, $H_{1-\mathrm{val}}$. On the other hand, using AMSE, we are not able to demonstrate the rejection of the null hypothesis, $H_{0-\mathrm{val}}$.

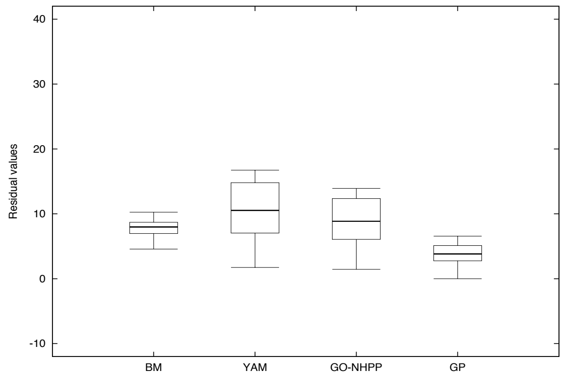
Table 3. Results of applying K-S test.

	J_{GP}	J_{BM}	J_{YAM}	$J_{GO-NHPP}$
Proj. 1, J_α=0.70	0.40	0.70	1.00	0.8
Proj. 2, J_α=0.64	0.27	0.73	0.82	0.54
Proj. 3, J_α=0.70	0.10	0.30	0.70	0.20

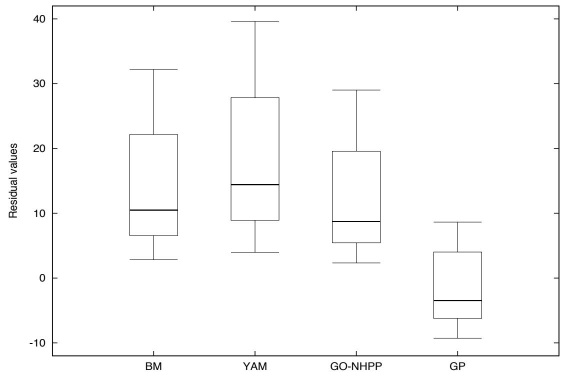
Table 3 shows the statistic J for the two sample K-S test performed on the validation fault count data ($1/3$ of the original data set) and the predictions by the GP and traditional reliability growth models. For project 1, we see that $J_{GP} < J_\alpha$, suggesting that the predicted fault count data, as provided by the GP model, fits quite well to the set of observations. On the other hand, the J statistic for all other traditional models are either equal to or greater than J_α. For project 2, the GP model along with GO-NHPP model have K-S statistic J less than J_α; and for project 3, GP model along with BM and GO-NHPP provide K-S statistic J less than J_α. While we see the traditional models giving statistically significant goodness of fit for project 2 and 3 on three occasions, neither of them gave statistics that were lower than the corresponding K-S statistic for the GP model. We can, thus reject the null hypothesis, $H_{0-\mathrm{gof}}$ in favor of the alternative, $H_{1-\mathrm{gof}}$.

Figure 2 shows the box plots of the residuals for all the models for the three projects. For project 1 (Figure 2(a)), all the box plots show the tendency of under-estimating; with the length of the box and tails of the GP model and BM model being smaller, indicating that the prediction bias is not severe. The tendency of the GP model incase of project 2 (Figure 2(b)) is to over-estimate but the bias is smaller as compared to other models. In case of project 3 (Figure 2(c)), all box plots represent a tendency to under-estimate while the GP model presents relatively less bias with residuals both above and below 0.

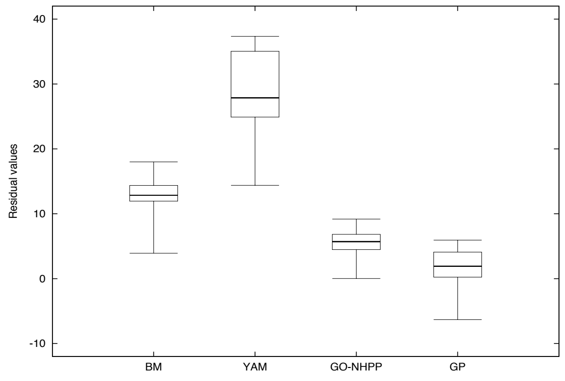
Since the box plots in Figure 2 are not significantly skewed, we applied matched paired t-tests of the residuals for each data set to compare the GP model with each of the traditional models. The results are presented in Table 4 and show that the residuals from the GP model are significantly different and less variable from the residuals for traditional models for each data set at α=0.05. Therefore, we are able to reject the null hypothesis, $H_{0-\mathrm{res}}$ in support of the alter-



(a) Box plots of residuals for Project 1.



(b) Box plots of residuals for Project 2.



(c) Box plots of residuals for Project 3.

Figure 2. Charts showing box plots of residuals for three projects.

Table 4. t-test results for residuals.

	$t_{GP:BM}$	$t_{GP:YAM}$	$t_{GP:GO-NHPP}$
Proj. 1, t_α=± 2.42	-32.18	-6.42	-6.59
Proj. 2, t_α=± 2.23	-7.76	-7.11	-7.53
Proj. 3, t_α=± 2.26	-23.43	-7.92	-4.56

native hypothesis, $\mathrm{H}_{1-\mathrm{res}}$.

7 Validity threats

Conclusion validity refers to the statistically significant relationship between the treatment and the outcome [25]. One of the threats to conclusion validity is that we might have missed applying a more suitable evaluation measure. However, to the authors' knowledge, the evaluation measures used in the study reflect the ones commonly used for evaluating prediction models. *External validity* is concerned with generalization of results outside the scope of the study [25]. The experiment is conducted on three different data sets taken from an industrial setting. However, these projects are carried out by one organization following similar development methods. We acknowledge that the generalizability of the research can be improved by experimenting with data sets taken from diverse projects employing different development methodologies. Also the significance of results can further be improved using larger data sets, thus giving an opportunity to evaluate the GP evolved models on greater number of data items.

8 Discussion and future work

In this paper, we compared the predictions of fault count data from models evolved using GP with three other traditional fault count software reliability growth models. The evaluation results show that prediction of fault count data using genetic programming is a promising approach.

We observed a considerable variation in the values of Braun statistic and AMSE for the GP model in three data sets; which can be attributed to the sensitivity of GP algorithm to changes in the training set. GP, being an adaptive algorithm, is able to discover pattern from a set of heterogeneous fitness cases.

In our case, we had one independent and one dependent variable. Hence, the GP algorithm generated good models efficiently within the termination criterion of 200 generations. However, it is common that efficiency and effectiveness of GP drops if the data tables contain hundreds of variables as the GP algorithm then can take a considerable amount of time in isolating the key features [23].

While measures of goodness of fit and predictive accuracy are important, we agree with Mair et al. [17] that these measures are not enough for a practical utility of a prediction system. The explanatory value (transparency of solution) and ease of configuration are also important aspects that require discussion. Since the output of a GP system is an algebraic expression, it has the potential of generating transparent solutions; however the solutions can become complex as the number of nodes in the GP solution increases. There is a trade-off in having more accurate predictions and less simplicity of the algebraic expressions but we believe that this tradeoff is manageable as achieving accurate models within acceptable thresholds is possible. In this respect, we also intend to evaluate GP evolved models against simpler regression models to compare predictions. In terms of ease of configuration of GP algorithm, different facets need to be determined, e.g. evaluation function, genetic operators and probabilities, population size and termination criterion to name a few. The parameter tuning problem is time consuming because the control parameters are not independent but interact in complex ways and trying all possible combinations of parameters is practically infeasible [23]. One possible way to reduce the effort is using adaptive parameter control during genetic programming run. Our future work intends to explore this possibility further.

An interesting area of future work is to compare the relative short-term and long-term predictive strength of the GP evolved model with traditional reliability growth models for different lengths of training data. Another possible dimension to explore is using a model that combines the results of GP evolved model and traditional SRGMs.

9 Conclusions

This paper presented the results of comparative evaluation of fault count data predictions from models evolved by genetic programming and traditional reliability growth models. Weekly fault count data of three different industrial projects was used in the study. The results have been evaluated in terms of model validity, goodness of fit and distribution of residuals. For evaluating model validity, the results of using prequential likelihood ratio and Braun statistic show favorability of the GP model. However, the results of AMSE did not show a statistically significant difference between the GP model and traditional software reliability growth models. The GP model was also found to have either an equivalent or better goodness of fit as compared to traditional models. The visual inspection of the box plots of residuals and matched paired t-tests further showed the GP model predictions to be less biased than traditional models.

References

[1] A. A. Abdel-Ghaly, P. Y. Chan, and B. Littlewood. Evaluation of competing software reliability predictions. *IEEE Transactions on Reliability*, 12(9):950–967, 1986.

[2] W. Afzal, R. Torkar, and R. Feldt. Prediction of fault count data using genetic programming. Submitted to ISSRE'08: The 19th International Symposium on Software Reliability Engineering, WA, USA.

[3] T. Bäck, D. Fogel, and T. Michalewicz. *Evolutionary computation 1—basic algorithms and operators*. Taylor & Francis Group, New York, USA, 2000.

[4] S. Brocklehurst and B. Littlewood. Techniques for prediction analysis and recalibration. In *Handbook of software reliability engineering, Editor M. R. Lyu*, Hightstown, NJ, USA, 1996. McGraw-Hill, Inc.

[5] W. D. Brooks and R. W. Motley. Analysis of discrete software reliability models. Technical report, IBM FEDERAL SYSTEMS, 1980.

[6] C. J. Burgess and M. Lefley. Can genetic programming improve software effort estimation? a comparative evaluation. *Information and Software Technology*, 43(14):863–873, 2001.

[7] E. K. Burke and G. Kendall. *Search Methodologies—Introductory Tutorials in Optimization and Decision Support Techniques*. Springer Science and Business Media, New York, USA, 2005.

[8] E. Costa, S. Vergilio, A. Pozo, and G. Souza. Modeling software reliability growth with genetic programming. In *ISSRE '05: Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering*, Washington, USA, 2005. IEEE Computer Society.

[9] A. L. Goel. Software reliability models: Assumptions, limitations, and applicability. *IEEE Transactions on Software Engineering*, SE-11(12):1411 – 1423, 1985.

[10] A. L. Goel and K. Okumoto. Time dependent error detection rate model for software reliability and other performance measures. *IEEE Transactions on Reliability*, R-28(3):206 – 211, 1979.

[11] GPLAB—A genetic programming toolbox for MATLAB. `http://gplab.sourceforge.net`.

[12] M. Hollander and D. A. Wolfe. *Non-parametric statistical methods*. John Wiley and Sons, Inc., 1999.

[13] IEEE Std 610.12-1990. *IEEE standard glossary of software engineering terminology*, 1990.

[14] Z. Jelinski and P. Moranda. Software reliability research. In *Statistical Computer Performance Evaluation, Ed. W. Freiberger*. Academic Press, USA, 1972.

[15] B. A. Kitchenham, L. M. Pickard, S. G. MacDonell, and M. J. Shepperd. What accuracy statistics really measure. *IEE Proceedings - Software*, 148(3):81–85, 2001.

[16] J. Koza. *Genetic programming: on the programming of computers by means of natural selection*. MIT Press, 1992.

[17] C. Mair, G. Kadoda, M. Lefley, K. Phalp, C. Schofield, M. Shepperd, and S. Webster. An investigation of machine learning based prediction systems. *Journal of Systems and Software*, 53(1):23–29, 2000.

[18] The MathWorks, Inc. `http://www.mathworks.com`. (Last checked 20 April 2008).

[19] J. Musa. *Software reliability engineering: more reliable software faster and cheaper*. AuthorHouse, 2004.

[20] A. P. Nikora and M. R. Lyu. An experiment in determining software reliability model applicability. In *Proceedings of the 6th International Symposium on Software Reliability Engineering*, pages 304–313, Oct. 1995.

[21] E. Oliveira, A. Pozo, and S. Vergilio. Using boosting techniques to improve software reliability models based on genetic programming. In *ICTAI '06: Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence*, Washington, USA, 2006. IEEE Computer Society.

[22] L. Pickard, B. Kitchenham, and S. Linkman. An investigation of analysis techniques for software datasets. In *6th International Software Metrics Symposium*, page 130, Los Alamitos, USA, 1999. IEEE Computer Society.

[23] R. Poli, W. B. Langdon, and N. F. McPhee. *A field guide to genetic programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008.

[24] S. F. Smith. *A learning system based on genetic adaptive algorithms*. PhD thesis, University of Pittsburgh, Pittsburgh, PA, USA, 1980.

[25] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering: an introduction*. Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[26] A. Wood. Software reliability growth models: assumptions vs. reality. In *ISSRE '97: Proceedings of the 8th IEEE International Symposium on Software Reliability Engineering*, Los Alamitos, CA, USA, 1997. IEEE Computer Society.

[27] S. Yamada, M. Ohba, and S. Osaki. S-shaped reliability growth modeling for software error detection. *IEEE Transactions on Reliability*, R-32(5):475 – 478, 1983.

[28] Y. Zhang and H. Chen. Predicting for MTBF failure data series of software reliability by genetic programming algorithm. In *Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications*, Washington, USA, 2006. IEEE Computer Society.

A Data sets used in the study

Project 1		Project 2		Project 3	
Week	Fault Count	Week	Fault Count	Week	Fault Count
1	9	1	15	1	3
2	9	2	18	2	12
3	24	3	24	3	18
4	24	4	30	4	30
5	27	5	39	5	60
6	27	6	60	6	93
7	39	7	69	7	138
8	45	8	72	8	186
9	54	9	87	9	210
10	54	10	126	10	240
11	54	11	129	11	258
12	57	12	132	12	279
13	57	13	144	13	297
14	57	14	147	14	312
15	57	15	156	15	348
16	66	16	162	16	357
17	66	17	171	17	372
18	69	18	171	18	399
19	75	19	174	19	414
20	81	20	180	20	429
21	90	21	186	21	459
22	99	22	192	22	486
23	102	23	207	23	519
24	105	24	210	24	540
25	108	25	222	25	552
26	120	26	234	26	570
27	120	27	237	27	588
28	123	28	249	28	612
		29	255	29	624
		30	279	30	630
		31	306		
		32	327		
		33	330		