An Initial Study of the Growth of Eclipse Defects

Hongyu Zhang
School of Software
Tsinghua University
Beijing 100084, China
hongyu@tsinghua.edu.cn

ABSTRACT

We analyze the Eclipse defect data from June 2004 to November 2007, and find that the growth of the number of defects can be well modeled by polynomial functions. Furthermore, we can predict the number of future Eclipse defects based on the nature of defect growth.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics - *Product metrics.*

General Terms

Measurement, Reliability.

Keywords

Defect growth model, defect prediction, polynomial regression, Eclipse.

1. INTRODUCTION

It is important to understand the nature of software defects so that we can have better estimation of software quality and better control of software process. The ability of predicting the number of defects can also help us allocate limited resources effectively and efficiently.

Over the years, there has been much research on defect analysis and prediction. For example, the work described in [8, 10, 11] focuses on the construction of defect prediction models based on the measurement of static code attributes, and the work reported in [1, 6, 12] analyzes the distribution of defects over modules in a large software system.

In this research, we study the growth of software defects over time. We use the public Eclipse defect data that is available at the MSR 2008 Challenge website [5]. We analyze the component-level defects from June 2004 to November 2007 and compute the number of cumulative monthly and weekly defects. We find that the number of defects growth over time and the growth curve can be well modeled by a polynomial function. Having understood the nature of the growth of defects, we can predict the number of future defects. We build the polynomial regression based models using historical defect data from June 2004 to December 2006, and use the models to predict the number of defects reported from January 2007 to November 2007. The prediction results are satisfactory.

2. THE GROWTH OF ECLIPSE DEFECTS

For the component-level Eclipse defect data, we collect the monthly defect numbers and calculate the cumulative values from June 2004 to November 2007. June 2004 is the month when Eclipse 3.0 was released and November 2007 is the last month when the complete monthly data is available. For monthly data, there are total 42 data points during this period. Let $f(1)$ represent the number of defects reported in June 2004, $f(2)$ represent the number of defects reported in July 2004, and $f(42)$ represent the number of defects reported in November 2007, the monthly defect numbers can be represented as a series:

$$f(1), f(2), \ldots, f(42)$$

The cumulative number of defects can be also represented as a series, which describe the growth of defects over time:

$$f(1), f(1)+f(2), \ldots, \sum_{i=1}^{41} f(i) + f(42)$$

In the same way, we collect weekly defect data from June 2004 to November 2007, which consists of total 183 data points.

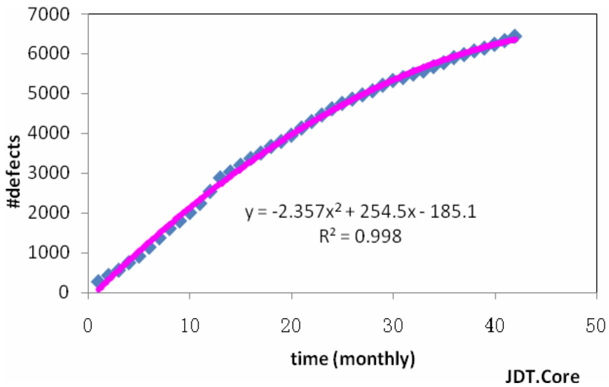
We analyze 14 major Eclipse components and plot their defect growth graphs for both monthly and weekly data. As an example, Figure 1 shows the defect growth graphs for components JDT.Core and Platform.Text. We can see that the numbers of defects continue growing over time. It is observed that all other components exhibit the similar behavior.

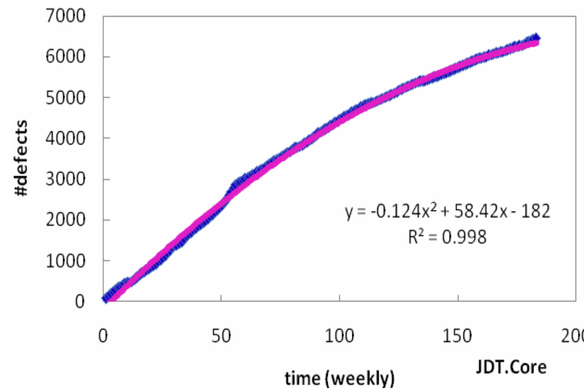
Polynomial regression is commonly used for curve fitting and trend analysis. It is a flexible method that can fit a wide range of curvature. We perform the polynomial regression analysis over the defect growth data, and obtain quadratic functions of the following form:

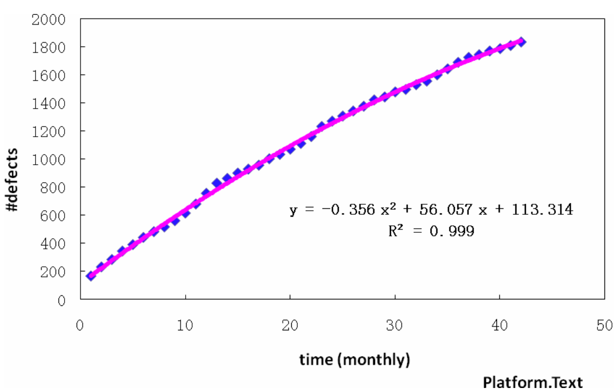
$$y = \beta_2 x^2 + \beta_1 x + \beta_0$$

where y is the number of cumulative defects, x represents the time (for monthly data x=1, 2, ..., 42, for weekly data x=1, 2, ..., 183), β_2, β_1, and β_0 are coefficients.

Table 1 shows the coefficients of polynomial models for all 14 Eclipse components we analyzed. The R^2 values range from 0.993 to 0.999, indicating good fitness of the models. All models are significant at the 0.001 level. We should note that for some components, other growth models such as exponential model or power model can be also applied. However, the polynomial model fits for all components with higher accuracy.







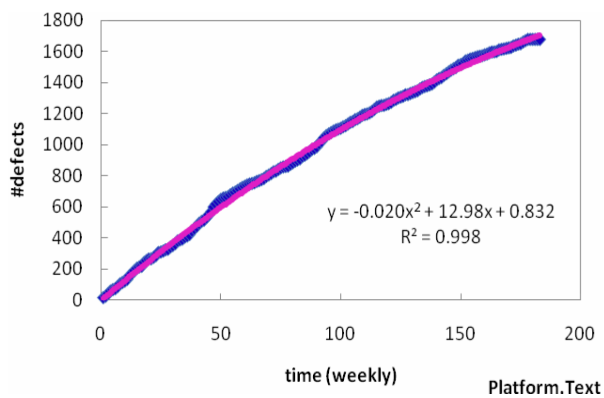


Figure 1. The growth of Eclipse defects from June 2004 to November 2007 (monthly and weekly)

	Monthly				Weekly			
	β_2	β_1	β_0	R^2	β_2	β_1	β_0	R^2
JDT.Debug	-0.427	85.242	-2.918	0.998	-0.028	20.42	-21.17	0.998
JDT.UI	-1.163	218.490	-73.986	0.996	-0.086	53.08	-116.10	0.997
JDT.Core	-2.357	254.50	-185.10	0.998	-0.124	58.42	-182.00	0.998
JDT.Text	-1.199	113.20	46.23	0.996	-0.064	26.19	46.27	0.996
Platform. Resource	-0.106	35.625	126.04	0.998	-0.011	9.102	103.9	0.998
Platform. Releng	0.001	21.384	-8.996	0.995	0.001	5.055	-9.077	0.997
Platform. IDE	0.222	4.404	4.095	0.995	0.011	1.757	1.436	0.995
Platform. Debug	-0.642	95.70	-27.562	0.995	-0.032	21.69	-34.88	0.997
Platform. Compare	0.035	11.804	16.288	0.997	0.008	2.281	23.55	0.995
Platform. Text	-0.356	56.057	113.314	0.999	-0.020	12.98	0.832	0.998
Platform. UI	-0.781	317.33	54.416	0.997	-0.044	73.50	56.93	0.998
Platform. SWT	-0.988	223.41	-32.717	0.998	-0.067	53.19	-56.46	0.999
PDE.UI	0.135	89.901	-46.731	0.993	0.014	19.70	-19.29	0.997
Equinox. Framework	0.729	-1.857	1.581	0.995	0.036	1.458	-17.03	0.995

Table 1. The coefficients of polynomial functions for studied Eclipse components

3. PREDICTING THE NUMBER OF FUTURE DEFECTS

Having understood the nature of the growth of Eclipse defects, we can then predict the number of future defects. For example, we build polynomial regression based models using defect data from June 2004 to December 2006 (total 31 data points), and use the models to predict the number of defects reported from January 2007 to November 2007 (total 11 data points).

To evaluate the accuracy of the prediction method, we use the metric MRE, which is defined as $MRE = |N - N^{\wedge}|/N$, where N and $N^{\wedge}$ are the actual and estimated values, respectively. The value of MRE is between 0 and 1. The commonly acceptable value is 0.25 (the smaller the value the better the estimation) [2].

Table 2 shows the prediction results for all studied Eclipse components. The actual and predicted numbers of defects from January 2007 to November 2007 are reported. For most of the components, the MRE values are below 25%, falling within the acceptable levels. The results show that we can predict the number of future defects reasonably well using the polynomial model built from historical data.

Table 2. Predicting the number of defects from January 2007 to November 2007 using data from June 2004 to December 2006

	Actual #defects	Predicted #defects	MRE	MRE <=25%?
JDT.Debug	522	595	13.94%	yes
JDT.UI	1237	1470	18.84%	yes
JDT.Core	1038	905	12.80%	yes
JDT.Text	378	253	33.07%	no
Platform. Resource	234	278	18.80%	yes
Platform. Releng	270	215	20.37%	yes
Platform. IDE	238	227	4.62%	yes
Platform. Debug	580	537	7.41%	yes
Platform. Compare	205	144	29.76%	no
Platfrom. Text	338	331	2.07%	yes
Platform. UI	2863	2863	0%	yes
Platform. SWT	1575	1664	5.65%	yes
PDE.UI	1226	1097	10.52%	yes
Equinox. Framework	475	565	18.95%	yes

4. CONCLUSION

Using the public Eclipse defect data, we have analyzed the growth of component-level defects over time, and found that the growth models can be well represented by polynomial functions. We have also shown that it is possible to predict the number of future defects based on the polynomial regression model built from historical data.

Models based on polynomial regression have limitations too. One major problem is that the extrapolation with polynomial could lead to unrealistic results [4]. Therefore predictions using polynomial models should be applied with cautions. In the future, we will explore if different model construction techniques or model selection criteria (e.g. [9, 3]) can help relieve this problem. For some Eclipse components, the factors that lead to large prediction deviations (MRE > 25%) need to be explored. We will also compare our method with related work on Eclipse defect prediction (e.g., [7]), and investigate if the findings reported in this paper can be generalized to other software systems.

5. ACKNOWLEDGMENTS

This research is supported by the 6th Key Researcher Program funded by the Tsinghua University, and the Chinese NSF grant 90718022. We thank M.Eng. student Lee Yunpeng for the computation of monthly and weekly Eclipse defect data.

6. REFERENCES

[1] C. Andersson and P. Runeson, A Replicated Quantitative Analysis of Fault Distributions in Complex Software Systems, *IEEE Trans. Software Eng.*, 33 (5), pp. 273-286, May 2007.

[2] L.C. Briand and I. Wieczorek, Resource estimation in software engineering, *Encyclopedia of Software Engineering*, (ed. J. Marcinak), Wiley, 2002. 1160-1196.

[3] J. R. Busemeyer and Y. Wang, Model comparisons and model selections based on generalization test methodology, *Journal of Mathematical Psychology*, June 1999.

[4] J. Cohen, *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences*, Lawrence Erlbaum Associates, 2003.

[5] Eclipse bug dataset, available at the MSR 2008 Mining Challenge website http://msr.uwaterloo.ca/msr2008/.

[6] N. Fenton and N. Ohlsson, Quantitative Analysis of Faults and Failures in a Complex Software System, *IEEE Trans. Software Eng.*, 26 (8), Aug 2000. pp. 797-814.

[7] H. Joshi, C. Zhang, S. Ramaswamy, C. Bayrak, Local and Global Recency Weighting Approach to Bug Prediction, *Proc. 4th Int'l workshop on Mining Software Repositories (MSR 2007)*, Minneapolis, USA, 2007.

[8] T. Menzies, J. Greenwald and A. Frank, Data Mining Static Code Attributes to Learn Defect Predictors, *IEEE Trans. Software Eng.*, vol. 32, no. 11, 2007.

[9] Rao, C. R., Prediction of Future Observations in Growth Curve Models, *Statistical Science*, vol. 2(4), 434-447, 1987.

[10] H. Zhang and X. Zhang, Comments on "Data Mining Static Code Attributes to Learn Defect Predictors", *IEEE Trans. on Software Eng.*, vol. 33(9), Sep 2007.

[11] H. Zhang, X. Zhang, M. Gu, Predicting Defective Software Components from Code Complexity Measures, *Proc. 13th IEEE Pacific Rim Int'l Symp. on Dependable Computing Conference (PRDC 2007)*, Melbourne, Australia, Dec 2007, IEEE Press, pp. 93-96.

[12] H. Zhang, On the Distribution of Software Faults, *IEEE Trans. on Software Eng.*, vol. 34(2), 2008.