Defect rate profile in large software-systems

Elena-Ramona MODROIU and Ina SCHIEFERDECKER
FOKUS Fraunhofer Institute, Berlin, Germany

Abstract. Software reliability, an issue raised long time ago by software engineering researchers, is still a problem that must be faced nowadays. Our approach is to handle reliability of software products making use of a statistical distribution based model and historical data of similar projects. We analyze the software error metrics of large software projects developed in a multinational company in the last five years, searching for a Rayleigh pattern in the defect rate profile. If we can prove empirically such a pattern exists in certain environment, we have a rapid and powerful theoretical model to support project estimation decisions early in the development life cycle when historical data on similar projects is available.

Keywords. Project estimation, software reliability, error rate profile, knowledge acquisition, historical data

Introduction

Quality of a software product encompasses different aspects like stability, portability, maintainability, usability among which reliability plays an important role. Software reliability can be thought of as the probability that a software to function without failure for a specified period of time, in a specified environment [6]. To evaluate reliability, there are employed basic metrics over defect occurrences in time such as tracking defect counts in a cumulative manner. We expect to find a Rayleigh pattern in the defect occurences and thus to ensure a theoretical framework for modeling of defects during the software testing cycle. In the following sections we show that the results of our analysis confirm our expectation.

Earlier in 1982, by studying 10 error histories, Trachtenberg was among the first ones to point out there might be the Rayleigh curve underlying the defect rate behavior in software products [1]. Schick-Wolverton predicted Rayleigh-shaped failure rates proposing a software reliability model derived from the assumption that the error detectability is linearly increasing with time [3]. This last model is part of a set of classical reliability models classified as linear (Jelinski-Moranda, Shooman, Musa), geometrical (Moranda, Ramamoorthy-Bastani) and Rayleigh (Schick-Wolverton). Later in 1990, Trachtenberg proposed a general framework to unify all these classical reliability models and showed that also the general theory predicts Rayleigh-shaped failure rates assuming that workload is increasing linearly with time [5]. Also QSM company published a few white papers and articles on the in-house defect estimation approach as being based on the Rayleigh distribution function to forecast the discovery rate of defects as a function of time throughout the software development process [2].

One of our goals is to compare new software development to previous developments. To achieve this we propose to determine the characteristics governing our environment, analyzing the knowledge provided by the measurement programs integrated in the development process. We have the chance to confirm the existence of similar environments among organizations generalizing the results we got by making use also of the findings reported in the afore mentioned research works. In the same time we extend the observation to different types of testing corresponding to the main phases in the product building process. This study was inspired by our team research work in the domain of quality of tests.

The paper is organized as follows. Section 1 describes the projects under study and software metrics that were used to measure them. Section 2 gives a short mathematical overview of Rayleigh model and the algorithm used to non-linearly fit the raw data to the statistical distribution. The results of non linear fitting are presented in section 3. We discuss further the importance of the results we got in the context of software projects estimation and the restrictions in applying a Rayleigh model. Section 4 summarizes the conclusions and future work.

1. Experiment settings

In this paper the focus is on studying the defect rate throughout the development life cycle of large, heterogeneous, multi-release software projects in the world of modern communication networks. A similar case-study is reported in [4]. We have now the chance to confirm the results among different companies and different software development processes.

The projects under study are large. They consist of many modules, are developed around the world within a multinational company and the effort employed represents a few hundreds man-years. Each project is a subsystem of a more complex software product, a network platform for mobile communications. As the mobile communication market has continuously grown in the last 10 years, also the technology was improved and new features had to be implemented in the software product. This resulted in new releases of the product almost each 2 years. Each release was in the category of high complexity project class due to new innovative hardware or software technologies, new software tools to be used, new algorithms to be created or component interfaces to be defined. A basic measurement program - common to CMMI Level 2 and 3 organizations - was in place, so we had plenty of data on defects discovered along each testing phase in the validation and verification process.

Our software product is made up of four main subsystems and data from two consecutive releases in the last 5 years was available. This means we have under study a set of 8 projects.

We attempt to discover proper knowledge to support project estimation decisions from following categories of historical data: size metrics (effective lines of code ELOC not including empty lines or comments), defect metrics (error finding status EFS, defects per development phase DDP) and effort metrics.

2. Data Processing Method

The software developing process is based on the V-Model graphically represented in figure 1. V-Model defines what, how and when has to be done during software development. It became a standard for German federal administration and defense projects, but also many companies have chosen to use it. There are certain phases that make up the life cycle of a software product. First part of the V corresponds to the building of the system, while the second part consists of the main testing types, each test activity being mapped to a certain building phase. Entry and exit points of the execution phases are marked formally by definition of generic milestones as listed in table 1.

Milestone	Short Description
M1	Product Definition
M2	Completion of Analysis
M3	Completion of Design
M4	Completion of Implementation
M5	Completion of Unit Test
M6	Completion of Component Integration Test
M7	Completion of System Integration Test
M8	Completion of System Test
M9	Completion of Customer Acceptance Test

Table 1. Milestones definition.

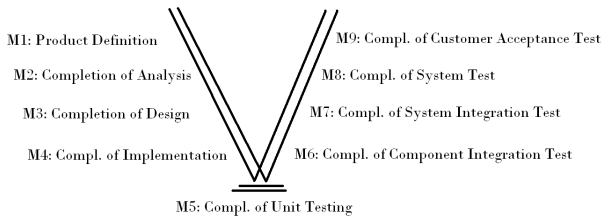


Figure 1. V-Model. Types of testing.

The error data, recorded per week in a cumulative manner during main testing activities - component integration test, system integration test, system test - will be analyzed against a particular case of Weibull distribution

$$W(m;k,\lambda) = 1 - e^{-(\frac{m}{\lambda})^k} \tag{1}$$

where $\lambda = \sqrt{2}\beta$ and $k = 2$ known as Rayleigh distribution function

$$R(x;\beta) = 1 - e^{-\frac{x^2}{2\beta^2}} \tag{2}$$

The fitting of the Rayleigh distribution to raw error data was done using Levenberg-Marquardt algorithm (LMA), present in almost any software that provides generic curve fitting tool. The LMA algorithm interpolates between Gauss-Newton algorithm (GNA) and the method of gradient descendant. It is an iterative procedure more robust than GNA, i.e. in many cases it finds a solution even if it starts very far off the final minimum.

3. Analysis of Results

In this section there are presented and analyzed the progress data collected independently by each test team. As the implementation is ready, first checks for code sanity are done by the code developers themselves. It is generally agreed that as late as a defect is found, more expensive will be to have it fixed. This phase is not going to be analyzed in this study, the focus being on defects discovered by specialized testing teams. Three main testing activities are well defined by verification and validation process and these are: component integration, system integration and system test. It follows then customer acceptance test, executed by clients, but this remains outside of the scope of our case-study.

The Rayleigh curve fitted to the defect data is

$$R(x) = a_0(1 - e^{-\frac{x^2}{2a_1^2}}) \tag{3}$$

where a_0 stands for the total number of errors to be discovered and a_1 is the scale parameter of the curve and represents the time when Rayleigh probability density function reaches a maximum.

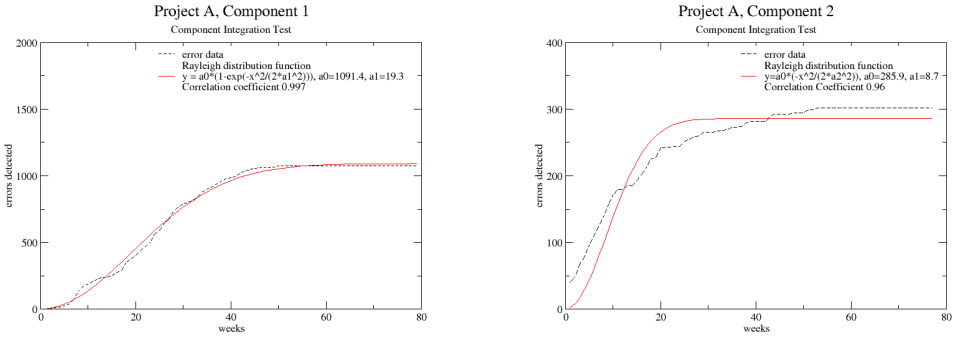
3.1. Component Integration Test

Component Integration Test is performed to integrate one subsystem of the larger project. As we are talking of very large subsystems, these were also made up of modules that have to be assembled and tested together to deliver a reliable component for the end product. The modules are already tested for stand alone functionality and correctness in the unit testing phase before to become available for component integration. This phase begins once milestone M5 is declared, and goes beyond M6 due to later change requests implementation and bug fixes of errors that did not represented a requirement criteria to M6 declaration.

Figure 2 presents the defect rate along component integration test for two projects: subsystem-1 and subsystem-2 of release A. The correlation coefficients between the fitted Rayleigh curve and the raw error data were 0.997 and 0.96. The results got for all projects in our study are listed in table 2.

Product	Project	Correlation coefficient to Rayleigh distribution
Release A	Component 1	0.997
	Component 2	0.96
	Component 3	0.962
	Component 4	0.984
Release B	Component 1	0.994
	Component 2	0.998
	Component 3	0.983
	Component 4	0.984

Table 2. Component Integration Test



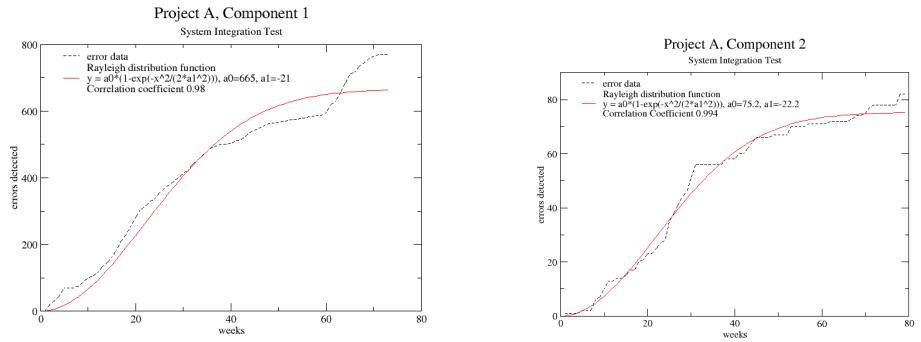
(a) (b)

Figure 2. Progress Data: Software defects

3.2. System Integration Test

System Integration Test is performed to integrate all subsystems and get the final software product. Test cases are executed to check the interoperability of the subsystems, focusing on testing the interfaces in between. The phase begins once milestone M6 is declared and continues until product is delivered to customer acceptance.

Figure 3 presents the defect rate along system integration test for two projects: subsystem-1 and subsystem-2 of release A. The correlation coefficients between the fitted Rayleigh curve and the raw error data were 0.98 and 0.994. Table 3 summarizes the results obtained for all 8 projects.



(a) (b)

Figure 3. Progress Data: Software defects

With an exception of a correlation coefficient of 0.94 in the case of component 4 in the release A, all the others coefficients were very high.

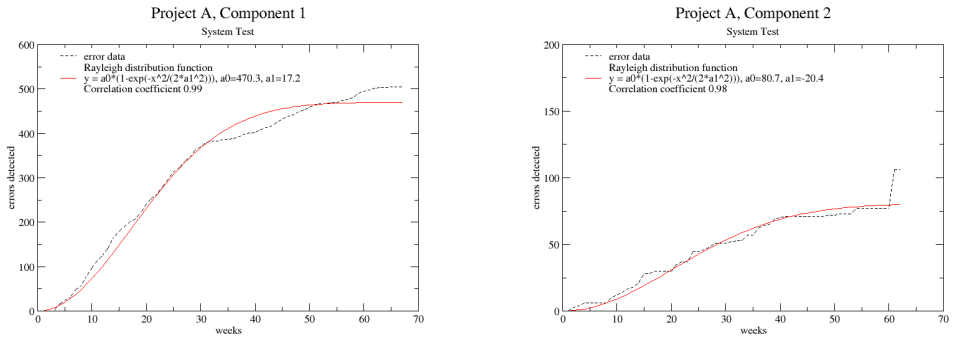
Product	Project	Correlation coefficient to Rayleigh distribution
Release A	Component 1	0.98
	Component 2	0.994
	Component 3	0.994
	Component 4	0.945
Release B	Component 1	0.995
	Component 2	0.973
	Component 3	0.977
	Component 4	0.981

Table 3. System Integration Test

3.3. System Test

In System Test, test cases are planned and executed to check for conformance with the requirements of the entire software product. Also non-functional checks such as performance, volume, stress, robustness are among major activities now. System Test begins once milestone M7 is declared and continues until product is delivered for customer acceptance tests.

Figure 4 presents the defect rate along system test activities of release A, for subsystem-1 and subsystem-2. The correlation coefficients between the fitted Rayleigh curve and the raw error data were 0.98 and 0.994. Table 4 summarizes all the results. It can be noticed, the highest correlation coefficients to Rayleigh curve were obtained for System Test phase.



(a) (b)

Figure 4. Progress Data: Software defects

3.4. Verification and Validation Phase

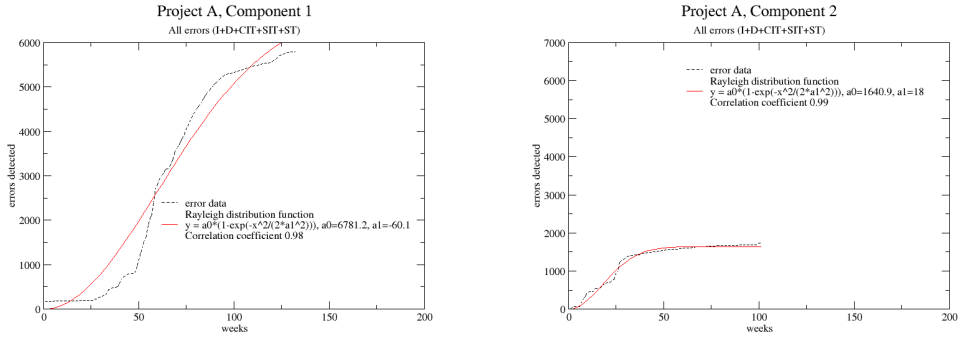
Verification and Validation corresponds to the second part of the software development process according to the V-Model and includes unit test, component integration test, system integration test, system test and customer acceptance test.

Figure 5 presents the defect rate along all testing phases for the same two projects, subsystem-1 and subsystem-2 of release A. The correlation coefficients between the fitted

Product	Project	Correlation coefficient to Rayleigh distribution
Release A	Component 1	0.99
	Component 2	0.98
	Component 3	0.98
	Component 4	0.98
Release B	Component 1	0.995
	Component 2	0.991
	Component 3	0.985
	Component 4	0.995

Table 4. System Test

Rayleigh curve and the raw error data were 0.98 and 0.994. Table 5 lists the results got for all 8 projects selected for this study. One could notice very high correlation coefficients to Rayleigh curve, comparable to system test phase.



(a) (b)

Figure 5. Progress Data: Software defects

Product	Project	Correlation coefficient to Rayleigh distribution
Release A	Component 1	0.98
	Component 2	0.99
	Component 3	0.98
	Component 4	0.99
Release B	Component 1	0.997
	Component 2	0.995
	Component 3	0.997
	Component 4	0.996

Table 5. Verification and Validation Phase

3.5. Decision Support in Project Estimation

This case study confirms there is a pattern in the defect rate profile and there is one governed by Rayleigh distribution function. The result was found for very large, heteroge-

neous software projects, as reported also in [4], and within an organization which has adopted V-Model as a standard development process.

Once proved the defect rate shows a certain distribution for projects of a company, this becomes a simple but powerful instrument for similar project estimations. It supports manager's decisions as when ready for product release ensuring a minimum required mean time to defect, what effort to pair up to testing to guarantee required quality in a given time interval. One way to achieve this target is by collecting for each project data sets as size, effort and β, where β is the Rayleigh distribution parameter computed for the project given the error progress reports. When a new project starts, assuming size and effort are estimated we can get from table the corresponding value for β and thus being able to plot an estimated Rayleigh-shaped error rate. Later on when data from the current project becomes available it can be compared to the estimated Rayleigh curve, having the possibility to identify deviations and analyze their cause, getting the project back on track. Also having such a table allows a manager what-if analysis. If the project size is about 2000 KLOC and an effort of 300 man-month is available, then using Rayleigh curve with the corresponding β from table it might be found that the project could be delivered in 2 years time. But what if the available effort is not only 300 man-month, but 400 man-month? What-if scenarios could be used by project managers if they had available data collected from previous similar projects. But this requires a big history of projects. It would be very useful if we could find a relationship between size, effort and then correlate it with β being able to interpolate the whole range of values size- effort-β. This is a future task do be done, when detailed effort data becomes available.

Similarity of two projects is decided by a series of attributes that could be classified into a few groups: domain of functionality, development environment and project complexity. We consider projects to belong to different domains of functionality such as banking applications, game applications, network communication platforms etc. Development environments are defined by the development model adopted within the organization, team experience and available software resources, while project complexity is given by the degree of innovation, size, multi-site need of development and duration of a project. The list of above mentioned attributes to characterize the similarity of two projects is not an exhaustive one, depending on the degree of similarity one would like to achieve, more detailed attributes could be added.

The Rayleigh model can be applied with two restrictions. The first one refers to the fact that defect rate during development phase must be positively correlated with defect rate in the field. The second one is about the errors' injection as being constant, and thus more defects discovered earlier means fewer defects are discovered later. Both this assumptions are related to the concept of "doing right the first time" [7].

4. Conclusions

The projects in this study presented a defect rate governed by a Rayleigh curve. This has been found valid for all testing activities seen as one single test phase and also for single testing types as component integration, system integration and system test. Higher correlation coefficients were obtained for system test and verification-validation scenario. This is in agreement with previous studies [2,3,4] which reported the same underlying pattern for all errors or for errors found by system test. Beside this, it has been shown

that also the other test types are characterized by a Rayleigh defect rate distribution. One could look at the verification and validation process as a sequence in time of 3 Rayleigh curves - having as starting points milestones M5, M6, respectively M7 - and in the same time the verification and validation process overall is represented by a Rayleigh curve.

It has been found that the Rayleigh curve is appropriate in modeling our environment of heterogeneous, complex projects developed in a multinational organization with cost of a few hundreds man-years. Other studies reported in the '80 the same model being appropriate for software development environments. Others reported the same findings more recently. This means a Rayleigh estimator can be applied among different organizations.

As a next step we would like to build a Rayleigh forecaster, a composite model built with data from same class of projects, and measure its performance when applied to new projects in the same class.

Certainly there are limitations in implementing such a forecaster, due to availability of data. For CMMI 2 or higher level organizations where measurements programs are implemented, this problem disappears. We see now that establishing a measurement program integrated in the development process helps any organization achieve an in-depth understanding of its software development environment, ensuring a solid background for further improvements in project planning and estimation.

References

[1] M. Trachtenberg, Discovering how to ensure software reliability, *RCA Engineer*, Jan.-Feb. 1982, 53-57.

[2] Lawrence H. Putnam and Ware Myres, Familiar Metric Management - Reliability, *www.qsm.com*.

[3] George J. Schick and Ray W. Wolverton, An analysis of Competing Software Reliability Models, *IEEE Trans. Software Eng. (1978)*, vol 4, no 2, 104-120.

[4] J.W.E. Greene, Ensuring Delivery of Highly Reliable Complex Software Releases, *Quantitative Software Management LTD, www.qsm.com 2003*.

[5] M. Trachtenberg, A General Theory of Software-Reliability Modeling, *IEEE Transactions on Reliability*, volume 39, number 1, April 1990.

[6] Dick B. Simmons and Newton C. Ellis and Hiroko Fujihara and Way Kuo, Software Measurement: A visualization Toolkit For Project Control and Process Improvement, *Prentice Hall PTR*, 1997.

[7] M. Neil and P. Krause and N. Fenton, Software Quality Prediction Using Bayesian Networks.