An Application of Simulation in Software Reliability Prediction

By Vojo Bubevski
Agenda

- About Me
- About TATA Consultancy Services (TCS)
- Introduction
- Proof of Concept based on Real System
- TRPC Project Simulation (Hypothetical Case)
- Recommendations for Future Work
- Conclusion
- References
- Questions & Answers
About TATA Consultancy Services (TCS)

- **Mission:**
  - To help Customers achieve their business objectives, by providing Innovative, best-in-class consulting, IT solutions and services. To make it a joy for all stakeholders to work with us.
- **CMM™ Level 5 Organisation**
- **142 Offices in 42 countries**
- **84 Global Solution Centres in 9 countries**
- **Business Week ranks TCS among the Top 10 most profitable global IT companies**
- **Companies, Bloomberg: Ranking (last reported fiscal year)**
  - Revenue: TCS ranked #11
  - Net Income: TCS ranked #5
  - Market Capitalisation: TCS ranked #4
  - No. of Employees: TCS ranked #5

TCS is a Global Player in IT Services, Business Solutions and Outsourcing
About Me

• At High School, Macedonia, 1968 – 1971 (IBM™ 1130, 4KB memory):
  – In Macedonian Maths Squad, Yugoslav Maths Olympiads
  – FORTRAN Programming, Summer School for Advanced Students, 1969

• At the University of Zagreb, Croatia, 1972 – 1977 (UNIVAC™ 1110):
  – Advanced courses in Probability & Stats, Applied Maths, Numerical Methods, Operations Research and Systems Modelling & Simulation
  – Systems Modelling & Simulation Assignments:
    • Supermarket Simulation Model, SIMULA
    • Factory Simulation Model, GPSS

• At Alkaloid Pharmaceuticals, Skopje, Macedonia, 1978 – 1986 (IBM™ System/370):
  – FORTRAN Package for Mathematical Optimisation: Linear programming (Simplex, Dual Simplex and Gomory Integer algorithms) & Non Linear programming (quadratic algorithm)
  – Optimisation Models Implemented in Practice:
    • Optimal Production Planning to maximise profit (Published);
    • Optimal Cutting of X-Ray Film to minimise waste;
    • Optimal Packing to maximise use of space;
    • Optimal Mixture in Pharmaceutical Technology to minimise raw material costs (Published)

Experienced the evolution of IT, Optimisation and Simulation Tools
Introduction: Definitions & Terms

- **Software Reliability**: The probability of failure-free software operation for a specified period of time (American National Standards Institute – ANSI);

- **Software Reliability Engineering (SRE)**: Discipline which quantitatively studies the operational behavior of software systems with respect to reliability requirements of the user [1];

- **Software Reliability Estimation**: Determines current software reliability based on failure data obtained in the past [1];

- **Software Reliability Prediction**: Determines future reliability of software systems based upon software metrics data available now [1];

- **Cumulative Failure Function**: The mean cumulative failures at each point in time [1];

- **Failure Intensity Function**: The rate of change of the cumulative failure function [1];

- **Mean Time to Failure (MTTF)**: The expected time that the next failure will occur [1];

- **Software Code Size Measure**: Source Lines of Code (LOC); KLOC is one thousand LOC;

- **Defect** refers to fault (i.e. the cause of failure), or **failure** (i.e. the effect of fault);
Introduction: Subject Context

• Classical Approach to Software Reliability Prediction: Analytic models using statistical analysis; available in literature since early 1970s;
• Major analytic models reviewed by Lyu [1];
• Analytic models main characteristic: Unrealistic and oversimplified assumptions required to obtain a simple analytic solution [1, 2];
• Need of modern approach, i.e. simulation, recognized in 1993 by Von Mayrhoaser et al. [2];
• Since 1993, the application of simulation in SRE has emerged and substantial work has been published, e.g. the articles by Tausworthe, Lyu, Gokhale, and Trivedi [3, 4, 5, 6];
• Results from these works indicated that the simulation technique is apparently a more accurate prediction than can be obtained from analytic modeling techniques [3];
• Simulation models appeared to be subject to only a few fundamental assumptions, such as the independence of the cause of failures [1, 7].
Introduction: A Key Related Work of Others

• A very interesting piece of work on software reliability simulation was published by Tausworthe and Lyu [7] as a chapter in a handbook of SRE [1].

• Elaborates the application of simulation techniques to typical software reliability processes eliminating the simplifying assumptions needed for analytic models [7].

• Special-purpose simulation tools for software reliability were designed, built and used in simulation experiments on a real-world project; the Galileo project at the Jet Propulsion Laboratory [1].

• Simulation results were very close to the real system’s data and much better than the prediction results obtained from analytic models;

• Compared with the simulation model, the analytic models do not seem to adequately predict the reliability of the system [7].
Introduction: An Outline of My Work

• An application of Palisade™ @RISK® (a general-purpose simulation tool) in software reliability prediction using Monte Carlo simulation with the Poisson distribution.
• Demonstrates the practical aspects of software reliability simulation (the theory is referenced only).
• Proof of Concept is established by reliability simulation experiments of a real system. A unique method is applied to transform the raw unusable failure data into data usable for simulation, without affecting the software reliability principles. The experiments are to select the most suitable model first and then use this model for reliability prediction of the real system.
• Prediction of reliability of a hypothetical financial software system is elaborated. The simulation experiment uses the data of the supposed current release, in order to predict the reliability of the supposed next release of the system. Important feasibility assumptions are discussed for this simulation experiment.
• Future work is recommended for supporting the software projects in achieving reliability goals with minimal costs.
• In conclusion, the presented approach, including the unique data transformation method, is generic and applicable to any software project compliant with CMM™ Level 4. Palisade™ @RISK® can be used for software reliability simulation. The experimental results are satisfactory. Using @RISK® is much easier than using the special-purpose software reliability tools. Also, @RISK® provides for comprehensive data presentation and analysis, which is not the case with the special-purpose tools. The simulation models are simple, but could be easily upgraded for more complex reliability prediction.
Proof of Concept: An Outline

• To prove the concept, we experiment with real system data to simulate software reliability using @RISK® tools.
• The data of the Galileo project at the Jet Propulsion Laboratory [1] is used.
• We prove the concept as follows:
  1. Present and analyse the actual (raw) Galileo data;
  2. Transform the Galileo data for simulation, as the raw data is unusable;
  3. Simulate the reliability using two different simulation models;
  4. Compare the two simulations results with the actual data in order to select the better simulation model for future Galileo simulations;
  5. Use the selected model to show how we can predict the reliability at the end of testing, supposing that we are in the middle of the testing stage.
Proof of Concept: Galileo Actual Data & Analysis

The total number of defects detected and removed during 41 weeks of testing is 351.

Mean Time to Failure for Galileo Project in testing: MTTF = 41/351 = 0.1168 Weeks.

The numbers of failures detected in each time interval are independent Poisson random variables [1];

It is impossible to correlate the number of failures detected each week.

The failure intensity function exhibits a strong-zigzag decreasing behavior.

Consequently, the data is raw and the failure intensity function is not practical for simulation.

However, we can transform the raw data to be usable for simulation without changing the software reliability principal values (i.e. the number of failures detected in each time interval, the time period, the total number of failures detected and MTTF).

Figure 1: Galileo project actual failure intensity function
Proof of Concept: Raw Data Transformation Method

- Fact: The numbers of failures detected in each time interval are independent Poisson random variables [1]
- We can reorder the time intervals preserving: a) the number of failures detected in each interval; b) the time period; c) the total number of failures detected during the time period; and d) MTTF.
- The criterion for reordering the intervals is that the number of failures detected in each interval must be put in descending order.
- This will transform the failure intensity function from a strong-zigzag decreasing type to a smooth decreasing type, which is usable for simulation.

- It should be noted that the MTTF of the raw Galileo data is equal to the MTTF of sorted Galileo data, i.e. MTTF = 41/351 = 0.1168.

Figure 2: Galileo sorted failure intensity function
Proof of Concept: Galileo Simulation 1 - Exponential Model

Figure 3: Exponential failure intensity function

Figure 4: Distribution of the Galileo simulation 1

- Simulation 1 model uses the Poisson distribution with an exponential failure intensity function (Figure 3). The approximation of the failure intensity $y$ (i.e. failures per week) as a function of time $x$ (in weeks) is $y = 34.74e^{-0.0884x}$. The mean of the Poisson distribution is equal to the value of the failure intensity function for time $t$.

- Simulation 1 Results (Figure 4): The predicted total number of defects is 361, which is quite close to the actual value of 351, with Standard Deviation 19 (i.e. 5.3%).
Proof of Concept: Galileo Simulation 2 - Logarithmic Model

• Simulation 2 model uses the Poisson distribution with a logarithmic failure intensity function (Figure 5). The approximation of the failure intensity $y$ as a function of time $x$ is $y = -8.6744 \ln(x) + 32.687$. The mean of the Poisson distribution is equal to the value of the failure intensity function for time $t$.

• Simulation 2 Results (Figure 6): The predicted total number of defects is 352 with Standard Deviation 19 (i.e. 5.4%). This result is almost equal to the actual value of 351.
Proof of Concept: Comparing Results & Selecting the best Model

• The simulation results are as follows:

  1. Simulation 1: Predicted total number of defects is 361 with Standard Deviation 19 (i.e. 5.3%). Compared with actual 351 defects, the error is 2.85%.

  2. Simulation 2: Predicted total number of defects is 352 with Standard Deviation 19 (i.e. 5.4%). Compared with actual 351 defects, the error is 0.28%.

• Comparing the actual number of errors, i.e. 351, with the results of Simulation 1 and Simulation 2, it is obvious that the Simulation 2 result is much better. Thus, the Simulation 2 model with the logarithmic failure intensity function is selected for future simulations of Galileo.
Proof of Concept: Galileo Simulation 3 – 21 Weeks Testing

Suppose that we predict the reliability at the end of 41 weeks testing, after 21 weeks of testing. The total number of defects for 21 weeks is 272, i.e. MTTF = 21/272 = 0.0772 Weeks.

Simulation 3: Poisson distribution with the transformed logarithmic failure intensity function (Figure 7). The transformed failure intensity $y$ as a function of time $x$ is $y = -8.8835 \ln(x) + 32.149$. The mean of the Poisson distribution is equal to the value of the failure intensity function for time $t$.

Simulation 3 Results (Figure 8): The predicted total number of defects is 310 with Standard Deviation 17 (i.e. 5.5%). This result is not bad compared with the actual result of 351 defects.
Proof of Concept: Galileo Simulation 4 – 23 Weeks Testing

**Figure 9:** 23 Weeks Transformed failure intensity function

**Figure 10:** Distribution of the Galileo simulation 4

- Suppose that we predict the reliability at the end of 41 weeks testing, after 23 weeks of testing. The total number of defects for 23 weeks is 301, i.e. MTTF = 23/301 = 0.0764 Weeks.
- Simulation 4: Poisson distribution with the **transformed** logarithmic failure intensity function (Figure 9). The transformed failure intensity \( y \) as a function of time \( x \) is \( y = -8.5579 \ln(x) + 32.289 \). The mean of the Poisson distribution is equal to the value of the failure intensity function for time \( t \).
- Simulation 4 Results (Figure 10): The predicted total number of defects is 346 with Standard Deviation 19 (5.49 %). This result is very good compared with the actual result of 351 defects.
Proof of Concept: Summary

- The actual Galileo failure-count data is presented and analysed.
- The data is raw and the failure intensity function exhibits a strong-zigzag decreasing behavior, which is unusable for simulation.
- The raw data can be transformed to provide for simulation without changing the reliability principals.
- Raw Data Transformation Method: Reorder the time intervals in descending order of number of failures, preserving: a) the number of failures detected in each interval; b) the time period; c) the total number of failures detected, and d) the reliability measure MTTF.
- The method is feasible because the numbers of failures detected in each time interval are independent Poisson random variables [1].
- The method transforms the failure intensity function from a strong-zigzag decreasing type to a smooth decreasing type, which is usable for simulation.
- The reliability is simulated using the Poisson distribution with two different approximations of the failure intensity function: a) Exponential; and b) Logarithmic.
- Comparing the results, the Logarithmic model is the better simulation model for Galileo.
- The logarithmic model is used to predict Galileo reliability in 41 weeks testing, supposing that we are in the middle of the testing stage. Two predictions are presented for this purpose.
- First prediction is at the end of week 21, so the simulation is based on 21 weeks worth of data. The predicted total number of defects is 310 with Standard Deviation 17 (i.e. 5.5%).
- The second simulation is based on 23 weeks worth of data as the prediction is at the end of week 23. The predicted total number of defects is 346 with Standard Deviation 19 (i.e. 5.49 %).
- Compared with the actual total of 351 defects, the first result of 310 defects is not bad (i.e. -11.68% error) whereas the second result of 346 defects is very good (i.e. -1.42% error).
- The second prediction is better than the first prediction, simply because the simulation was carried out on data taken over a longer time period, i.e. the simulation is based on more available data.
- In conclusion, the experimental results are satisfactory and the concept is proven.
Next Release Simulation: TRPC Project (Hypothetical)

- This simulation experiment is hypothetical, so it is for illustration purposes only.
- It is supposed that we have collected data of two subsequent releases of the hypothetical financial software system – Project TRPC.
- In this reliability simulation, we use the supposed current release data to simulate the reliability of the supposed next release of the TRPC software system.
- The following is an outline of the approach to Next Release Simulation:

  1. Feasibility assumptions are discussed;
  2. TRPC Project two releases data are presented;
  3. Simple Simulation model is demonstrated;
  4. Simulation results and actual data are compared.

- Even though this experiment is hypothetical, it illustrates how we can simulate the next release of a software system, using the data of the current release.
Next Release Simulation: Feasibility Assumptions

• The software reliability simulation uses failure data collected in the past. However, having only the collected data is not sufficient to provide for feasibility of the software reliability simulation. Other much more complex criteria must be met as well.

• For example, if an organisation collects and has a history of data for their software projects, but does not meet the other criteria, the software reliability simulation is not feasible. The data provide for running a simulation, but any reliability prediction initiative is unrealistic and even dangerous because the simulation results are inconsistent and any decision made based on these results is very risky.

• The following defines the fundamental assumptions for the feasibility of the software reliability simulation. That is, Software Reliability Simulation and Prediction are feasible only if the software organisation and the software project are compliant with Capability Maturity Model – CMM™ Level 4.

• CMM™ Level 4 requires quantitative management of software processes and products within an organisation. The criteria are as follows: “Detailed measures of the software process and product quality (must be) collected. Both the software process and products (must be) quantitatively understood and controlled.” [8].

• Some aspects of software reliability prediction relating to CMM™ LEVEL 4 are discussed by Lakey [9].
Next Release Simulation: TRPC Current Release Data

Supposed Current Release Actual Data

1. New Code
   - Size KLOC: 29
   - Operation Defects in 40 Weeks: 342
   - Test & Fix Effort Man-Days: 1860
   - Defects Found in Test & Fixed: 1041
   - Total Defects: 1383
   - Total Defects: 47.69 per KLOC

2. Changed Code
   - Size KLOC: 16
   - Operation Defects in 40 Weeks: 116
   - Test & Fix Effort Man-Days: 955
   - Defects Found in Test & Fixed: 355
   - Total Defects: 471
   - Total Defects: 29.44 per KLOC
Next Release Simulation: The Model

- Expected size of the new code is 40 KLOC. We predict the new code size using a Normal distribution with a Mean of 40 and a Standard Deviation of 2 (5%).

  **New Code Size Prediction (Normal Distribution)**

  **New Code Predicted Size KLOC: 40.00**
  Mean Value (μ): 40.00
  Standard Deviation (σ): 2.00

- Expected size of the changed code is 25 KLOC. We predict the changed code size using Normal distribution with Mean of 25 and Standard Deviation of 1.25 (5%).

  **Changed Code Size Prediction (Normal Distribution)**

  **Changed Predicted Size KLOC: 25.00**
  Mean Value (μ): 25.00
  Standard Deviation (σ): 1.25
Next Release Simulation: The Model (continued...)

- Assumed Defect Injection Rate (DIR) for the new code will be equal to the current release rate, i.e. 47.69 Defects/KLOC. To predict this parameter we use a Normal distribution with a Mean of 47.69 and a 5% Standard Deviation (i.e. 2.38).

  **New Code Defect Injection Rate (DIR) Prediction** *(Normal Distribution)*

  Current Release DIR per KLOC: 47.69
  
  **New Code Predicted DIR per KLOC: 47.69**

  Mean Value (µ): 47.69; Standard Deviation (σ): 2.38

- For the changed code, the assumed Defect Injection Rate (DIR) will be equal to the current release rate, i.e. 29.44 Defects/KLOC. To predict this parameter we use a Normal distribution with a Mean of 29.44 and Standard Deviation of 1.47 (5%).

  **Changed Code Defect Injection Rate (DIR) Prediction** *(Normal Distribution)*

  Current Release DIR per KLOC: 29.44
  
  **Changed Code Predicted DIR per KLOC: 29.44**

  Mean Value (µ): 29.44; Standard Deviation (σ): 1.47
Next Release Simulation: The Model (continued...)

• We assume that for the new code the effort required to test and fix defects during testing is equal to the current release effort, i.e. 64.14 Man-Days / KLOC. To predict this parameter we use a Normal distribution with a mean value of 64.14 and Standard Deviation of 3.21 (5%).

**Effort for Testing & Fixing New Code Prediction (Normal Distribution)**

Current Release Man-Days per KLOC: 64.14

Test & Fix Predicted Man-Days per KLOC: 64.14

Mean Value ($\mu$): 64.14; Standard Deviation ($\sigma$): 3.21

• Similarly, for the changed code, we assume that the effort required to test and fix defects during testing is equal to the current release effort, i.e. 59.69 Man-Days / KLOC. To predict this parameter we use a Normal distribution with a mean value of 59.69 and Standard Deviation of 2.98 (5%).

**Effort for Testing & Fixing Changed Code Prediction (Normal Distribution)**

Current Release Man-Days per KLOC: 59.69

Test & Fix Predicted Man-Days per KLOC: 59.69

Mean Value ($\mu$): 59.69; Standard Deviation ($\sigma$): 2.98
Next Release Simulation: The Model (continued...)

- We expect that the new code Defect Removal Rate (DRR) will be the same as the current rate, i.e. 1.79 Man-Days / Defect. We predict this parameter using a Normal distribution with a mean value of 1.79 and Standard Deviation of 0.09 (5%).

  **New Code Defect Removal Rate (DRR) Prediction** *(Normal Distribution)*
  
  Current Rel. DRR Man-Days per Defect: 1.79

  **New Code Predicted DRR Man-Days per Defect:** 1.79

  Mean Value ($\mu$): 1.79; Standard Deviation ($\sigma$): 0.09

- Similarly, the changed code Defect Removal Rate (DRR) will be the same as the current rate, i.e. 2.69 Man-Days / Defect. We predict this parameter using a Normal distribution with a mean value of 2.69 and Standard Deviation of 0.13 (5%).

  **Changed Code Defect Removal Rate (DRR) Prediction** *(Normal Distribution)*
  
  Current Release DRR Man-Days per Defect: 2.69

  **Chang. C. Predicted DRR Man-Days per Defect:** 2.69

  Mean Value ($\mu$): 2.69; Standard Deviation ($\sigma$): 0.13
Next Release Simulation: The Model (continued...)

- Using the parameters above, we calculate the Defect Injection and Defect Removal Intensities for the new and changed codes, as given below. We simulate now the numbers of injected and removed defects (shown in Blue below) using the Poisson distribution with the mean value equal to the associated defect intensity. The total number of defects in operation (shown in bold Blue below) is the difference between injected and removed defects.

**Next Release Defect Totals Prediction (Poisson Distribution)**

- New Code (NC) Defect Injection Intensity: 1902.60
- Changed Code (CC) Defect Injection Intensity: 735.94
- NC Defect Removal Intensity: 1435.90
- CC Defect Removal Intensity: 554.72
- Predicted NC Defects Injected: 1903.00
- Predicted CC Defects Injected: 736.00
- Predicted NC Defects Removed: 1436.00
- Predicted CC Defects Removed: 555.00
- Predicted Total Defects in Operation: 648.00
Next Release Simulation: The Results

- For this simulation (Figure 11), the predicted total number of defects is 648 with Standard Deviation 164 (i.e. 25.31%).
- The high Standard Deviation of 25.31% is caused by the high number of random variables used in the model.
- **Simulation Results Vs Actual Data**
  - New Code Size KLOC: 38 / 40 / 5.27
  - Changed Code Size KLOC: 26 / 25 / -3.85
  - Total Code Size KLOC: 64 / 65 / 1.56
  - Total Defects in Operation: 690 / 648 / -6.09

- The results are quite good (-6.09% error for total number of defects), so our hypothetical experiment is successful.

**Figure 11: Distribution of the TRPC Project Next Release Simulation**
Next Release Simulation: Summary

• This simulation experiment is hypothetical; however it demonstrates how software project next release can be simulated, using the data of the current release.
• The simulation model is simple considering only the testing and operational phase of the software project.
• The model can be easily expanded to involve the analysis and design phase of the project if the failure data is available.
• The assumptions are discussed to establish the criteria for which the reliability simulation is feasible.
• Software Reliability Simulation and Prediction are feasible only if the software organisation and the software project are compliant with Capability Maturity Model – CMM™ Level 4.
• TRPC Project data of the “current release” are presented.
• The simulation model is demonstrated, which predicts the reliability of the “next release” using the “current release” data.
• The simulation results are compared with the actual “next release” confirming that the experiment is successful.
Future Work Recommendations

• The management objective is to achieve the software system reliability goals with minimal costs and schedule of projects.

• Software reliability simulation is very useful in supporting software project management to achieve this objective.

• For future work, it is recommended to develop optimization models for this purpose.

• For example, supposing that the management want to employ extra resources on the project, but the resources are limited.

• The problem here is to determine how to utilize the limited resources and improve the reliability in the most cost-effective way.

• An appropriate optimization model can provide for an optimal solution to this problem, i.e. to maximize the reliability improvement by optimal utilization of the limited resources.
Conclusion: Summary

• Experiments of software reliability simulation are presented using Palisade™ @RISK®.
• The simulation models use Monte Carlo sampling with the Poisson distribution.
• The purpose is to demonstrate the practical aspect of simulation; the theory is not discussed.
• Proof of concept is demonstrated with simulations of a real system, i.e. the Galileo project at the Jet Propulsion Laboratory [1].
• Unique method is elaborated and applied to transform raw unusable failure-count data into data usable for simulation. The method does not affect the software reliability principles.
• This method is generic and applicable to any software reliability simulation.
• Galileo Project Testing is simulated in the experiments and results are compared with the actual data.
• Experimental results are satisfactory, hence proving the concept
• Simulation of a hypothetical system (Project TRPC) is elaborated.
• The purpose of the experiment is to use the data of the current release to simulate the next release.
• Important feasibility assumptions are discussed and established.
• The simulation model is presented and the experiment results are compared.
• The simulation model is simple including only the testing and operation phases. The model can be expanded to consider the analysis and design failures if data is available.
• For future work, it is recommended to develop optimization models to support the management in achieving the software system reliability goals with minimal costs.
Conclusion: Major Points to Emphasise

• Firstly, the presented approach to software reliability prediction is generic and applicable to any CMM™ Level 4 software project.
• Secondly, the method for transforming the unusable raw failure-count data into data usable for simulation is unique and generic.
• Thirdly, the general purpose simulation tools such as Palisade™ @RISK® can be used for software reliability simulation. The experimental results presented in this paper are satisfactory. Using Palisade™ @RISK® is much easier than using the special-purpose tools. Also, Palisade™ @RISK® tools provide for comprehensive data presentation and analysis, which is not the case with the special-purpose tools.
• Finally, the presented simulation models are simple. However, the models can be upgraded to provide for more complex reliability simulation if data is available.
References

Questions & Answers