Development, Role and Significance of Operational Profile in Software Reliability Estimation

Wali Ullah [1], Riyaz Ahmad Khan [2], S.P.Tripathi [3]

ABSTRACT

This paper provides detail description for developing the operational profile which is essential in any software reliability engineering application. An operational profile of software is a key factor to analyze failure behavior of software and the relationship between software reliability and operational profile has been recognized as a major issue in estimating software reliability.

Keywords: Reliability, failure and faults, Input variable, Operational architecture, Operational profile, environment, Input state, Input space.

1. INTRODUCTION

The functions of the computers depend on the correct operation of their software. Unlike computer hardware, whose reliability has improved dramatically with the advances in integrated circuit technologies, the improvements in the reliability of software have been lagging. A series of accidents caused by unreliable software proves that the reliability of the software is a matter of life and death. Software reliability is defined as the probability of failure free operations for a specified period of time in a specified environment. Failure rate of software depend critically on the environment in which it is executing. Software may frequently fail for a given input space, or in other words if software has faults, only input space will exercise that fault to affect failures. The input space causing failure will decide the software failure rate [1]. Reliability is based on the failure which in turn depends on the nature of input being used by the software during execution, reliability is clearly dependent on the operational profile of the software. If the operational profile of software changes dramatically, reliability of systems will need to be either recomputed or recalibrated [2].

2. SOFTWARE RELIABILITY

According to Bew Littlewood "Software reliability means operational reliability. Who cares how many bugs are in the program? We should be concerned with their effect on its operations" [3].

As per IEEE standard [IEEE 90]; "Software reliability is defined as the ability of a system or component to perform its required functions under stated conditions for a specified for a time". Software reliability is also defined as the probability that a software system fulfills its assigned task in a given environment for a predefined number of input cases, assuming that the hardware and the inputs are free of error. Hence it is the probability that the software will work without failure for a specified period of time in a given environment. Here environment and time is fixed. Reliability is for fixed time under given environment or stated conditions. So, reliability is always for a well defined domain.

[1] Department of Computer Science Mumtaz P.G. College, Lucknow (UP) Email Address:walishah@rediffmail.com

Faculty of Applied Science, Integral University, Lucknow riyazakhan68@yahoo.co.in

Institute of Engineering and Technology, UPTU, Lucknow tripathee_sp@yahoo.co.in

The most acceptable definition of software reliability: "It is the probability of a failure free operation of a program for a specified period of time in a specified environment [4]. A time sharing- system may have a reliability of 0.95 for 10 hr when employed for the average user. This system, when executed for 10 hr, would operate without failure for 95 of these periods out of 100. The reliability of a software-based product depends on how the computer and other external elements will use it [5].

It is well known of major factors driving in any production discipline is quality of software product has three dimension in which product operation deals with the key factor reliability [6]. Software reliability is an important attribute of software quality, together with functionality, usability, efficiency, maintainability and portability. Reliability is the property referring to how well software meets its requirements and also the probability of failure free operations for the specified period of time in a specified environment. It is a probabilistic phenomenon. This randomness means the failure may not be predicted accurately. Software reliability is hard to achieve, because the complexity of software tend to be high.

3. Operational Profile

A software system can fail due to the inputs it receives from the external environment [7]. The environment in which software is executing affects the failure of the software. Perfectly working software may also breakdown or fail if the running environment is changes. After success of Ariane-4 rocket the maiden flight of Ariane-5 ended up in the flames while designing defects in the control software were unveiled by the faster horizontal drifting speed of the new rocket. It seems that the greatest problem in the field of software reliability estimation is the accuracy of operational profile. The operational profile is defiled as the probability density function (over the entire input space) the best represents how the inputs would be selected during the life time of the software. The operational profile of the software reflects how it will be used in practice. It consists of a specification of classes of input and the probability of their occurrence. When an existing manual or automated system is being replaced by new system, it would be easy to assess the probable pattern of usage of new software. The reliability of a software-based product depends on how the computer and other external elements will use it.

The operational profile is a quantitative characterization of how the software will be used, assuming occurrence probability to software operations. Therefore operational profile is essential in any Software Reliability Engineering (SRE) application.

The operational profile is defined as the probability density function (over the entire input space) that best represent how the input would be selected during the life time of the software. The operational profile of the software reflects how it will be used in practice [8]. It consists of a specification of class of input and the probability of their occurrence. When an existing manual or automated system is being replaced by new software system, it would be easy to assess the problem pattern of usage of new software. It may roughly correspond to the existing usage some allowance made for the new functionality, which is included in the new software.

The operational profile is a quantitative characterization of how a software system is used, assuming occurrence probability to software operations. It is used in the test environment to ensure that the most used operation receives the most testing [9].

The operational profile is a fundamental concept which must be understood in order to apply SRE effectively and

with any degree of validity. This section provides a detailed description of the Operational Profile.

A profile is a set of independent possibilities called elements, and their associated probability of occurrence. If operation *A* occurs 60 percent of the time, *B* occurs 30 percent, and *C* occurs 10 percent, for example, the profile is [*A*, 0.6...*B*, 0.3...*C*, 0.1].

4. Developing an Operational Profiles

Operational profile is a complete set of operations with their probabilities of occurrence. Probability of occurrence refers to probability among all invocations of all operations. Developing an operational profile for a system involves one or more of the following steps [10].

1. Find customer profile
2. Establish the user profile
3. Define the system-mode profile
4. Determine the functional profile
5. Determine the operational profile itself

4.1 Customer profile

It is an array of independent customer types. A customer type is one or more customers in a group that intend to use the system in a relatively similar manner, and in a substantially different manner from other customer types.

An example of a software system with different customer types would be a database package. (Table 1)

Various customers could be Industry, Railway, educational institutions, businesses, and individual home users. Each of these types of customers may be expected to utilize the database in a substantially different way. For instance, Industrial organization might use them for inventory control and payroll system. Railway might use them for reservation system. Schools might use them for tabulating and updating student grades. Businesses might use them mainly for financial and operations controls. Home users could keep track of their monthly income and expenses, as well as investments and savings plans. The customer profile is the list of customer types and the associated probabilities. These probabilities are simply the proportions of time that each type of customer would be using the system.

Table 1 : Customer profile

Customer type	Occurrence Probability
Industrial Organization	0.55
Railway Organization	0.20
Educational Institution	0.13
Business Organization	0.11
Individual Home User	0.01

4.2 User Profile.

The user profile is the set of user types and their associated probabilities of using the system. A user type is a set of users that will operate the system similarly. Identification of different user types allows the task of operational profile development to be divided among analysts.

4.3 System Mode Profile.

A system mode is a way that a system can operate. The system includes both hardware and software. Most systems have more than one mode of operation. For example, system testing may take place in batch mode or user-interactive mode. An airplane flight consists of takeoff and ascent mode, level flight mode and descent and land mode. A system can switch among modes sequentially, or

it can permit several modes to operate concurrently, sharing the same system resources. For each system mode, if there are more than one or two, an operational profile (and sometimes functional profile) should be developed. There are no technical limits on how many system modes may be established. The system mode profile is represented by the list of system modes and their corresponding occurrence probabilities. (Table 2)

Table 2 : System mode profile

System Mode	Occurrence Probability
Batch Mode	0.65
User-Interactive Mode	0.35

4.4 Functional Profile.

Functions are essentially tasks that an external entity such as a user can perform with the system. For instance, the payroll system has following functions open an employee account, enter the monthly attendance, and print out of the pay slip etc. Functions are established during requirements based on what activities the customer wants the system to be able to perform.

The functional profile can be either explicit or implicit, depending on the key input variables.

A key input variable is an external parameter which affects the execution path a software system traverses based on the different values the parameter takes on. These key parameter variables in many cases consist of ranges of variables that cause different operations to be performed. These various ranges are referred to as levels. A profile is explicit if each element is designated by simultaneously specifying the levels of all key input variables needed for its identification. A profile is implicit if it is expressed by sub profiles of each key variable. That is, each key environmental parameter is assigned probabilities associated with the ranges it can legally use.

Suppose there are two key independent parameters, X and Y, each taking on three discrete values. Nine operations can be defined based on the combinations of the variables. (Table 3). The main advantage of using the implicit profile is that a significantly smaller number of elements need to be specified, as few as the sum of the number of levels of key input parameters. (Table 4) The explicit profile can have as many as the product of the number of levels for each variable. In most cases it is not necessary to generate the explicit profile, because it exists by default from the implicit profile.

Table 3 : Implicit Operational Profile

Key input variable Values	Occurrence Probabilities
X1	0.5
X2	0.3
X3	0.2
Y1	0.7
Y2	0.2
Y3	0.1

Table 4 : Explicit Operational Profile

Key input variable Values	Occurrence Probabilities
X1Y1	0.35
X2Y1	0.21
X3Y1	0.14
X1Y2	0.01
X2Y2	0.06
X3Y2	0.05
X2Y3	0.04
X2Y3	0.03
X3Y3	0.02

Development of the functional profile generally involves the following four steps:

1. Generate an initial function list

2. Determine environmental variables

3. Create final function list

4. Assign occurrence probabilities

The initial function list should be comprised of features and capabilities needed by the users. This list can be organized by functions relevant to each key input variable if an implicit profile is used. Features should be obtained from the customer and/or users, and may be stored in a system requirements specification. The functions should be identified rather easily if a good job of requirements evaluation has been done. If functions are difficult to identify, the requirements may be incomplete or unclear. A Request for Proposal is often issued for government programs which contains a list of capabilities or features that a system must have.

The next step is to define the environmental input variables and their value ranges that segregate development. Functions will many times be broken up and allocated to different modules for development based on environmental variables. These environmental variables characterize the conditions that influence the paths traversed by a program, but do not correspond directly to features. Examples of environmental variables include hardware configuration and traffic load.

The system design team should work together to brainstorm a list of environmental variables that would cause different behaviors of the software program, and pare this list down.

Prior to creating the final function list the key environmental and feature variables should be examined for dependencies. The list should be as orthogonal as possible. If one variable is significantly dependent on another, it can be eliminated from the final function list. Partial dependencies can cause difficulties because all possible combinations of levels of both variables may need to be listed. Those interactions which cannot occur and which have insignificant likelihood of ever occurring should be removed from the list. The final number of functions in the list is then calculated as the product of the number of functions in the initial list and the number of environmental variable levels, minus the combinations of initial functions and environmental variable values that do not occur. The final function list consists of the functions and environmental variables for each function.

The final step in functional profile development is the assignment of occurrence probabilities. The ideal data source for these values consists of usage measurements taken on the latest release or a similar system. These measurements may be obtained from system logs or data storage devices. Occurrence probabilities computed with the historical data should be updated to account for new functions, users, or environments. In the event that a system is completely new the functional profile might be very inaccurate. It should still be developed, however, and updated as more is known about how the system will be operated. The process of predicting usage forces interaction with the customer, which can be very important. The required dialogue may highlight the relative importance of the various functions, indicating that some functions may not be necessary while others are most significant. Reducing the number of functions should increase reliability [11].

4.5 Operational Profile.

Figure1 shows the elements involved in determining operational profiles from functions. A function may comprise several operations. In proper order, operations

are made up of many run types. Grouping run types into operations partitions the input space into domains. A domain can be partitioned into sub domains, or run categories [12]. To use the operational profile to drive testing, first choose the domain that characterizes the operation, then the sub domain that characterizes the run category, and finally the input

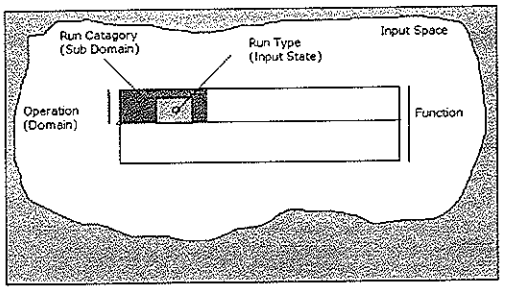

Figure 1 Operational Elements

state that characterizes the run. The functional profile is a user-oriented view of system capabilities. From the developers' perspective, it is operations that actually implement the functions. Operations are usually the focus of testing. An operation represents a task being accomplished by the system from the viewpoint of the people who will test the system. To allocate testing effort and develop a test description, the operational profile must be available for the purposes of test planning.

The operational architecture describes how the user will likely employ operations to accomplish functions. In general there are more operations than functions, and, as Figure 1 shows, operations tend to be more refined. The primary objectives in determining the operational profile include listing the operations and determining the occurrence probabilities [13].

The list of operations can be extracted from the functional profile, mapping functions to operations using the operational architecture of the system. If possible, functions should be defined so that there is a one-to-many correspondence between functions and operations. This greatly simplifies the derivation of operational profile from functional profile.

The three steps in determining the operations list are to divide the execution into runs, identify the input space, and partition the input space into operations.

4.5.1 Divide execution into runs.

A run is essentially a discrete task performed by a system in response to external variable(s). Runs from the same class form a run type. To be in the same class the runs should take the same execution path through the software. Any variation in the level of any input variable results in a separate run type. Each run type has an associated input state, a set of input variables and their associated levels that generates the run. The input state from an input space uniquely determines the path of control of an execution of a run. An input parameter is allowed only one value for each run. Externally initiated interrupts would be considered input variables because they are parameters which exist external to the program. Intermediate data items, obtained within a run, are not input variables.

4.5.2. Identify input space.

A software system's input space, combinations of inputs and the number of different values they can take on, is prohibitively large for a program of even low complexity. An input state profile is the set of input states and their associated occurrence probabilities. The complete input state profile is never really measured in practice because any continuous variable can potentially take on an infinite number of values. It is important to understand because it can help in identifying approximations that cost-effectively approach the ideal scenario. An input space can be identified by simply listing the set of input variables involved.

Partition input space.

In practice, the profiles should be limited to several hundred elements due to cost restrictions. The input space can be dissected by selecting ranges for input variables that belong to the same run type. By grouping run types into operations, the input space can be partitioned to significantly reduce the number of elements. The larger the ranges (less levels), the smaller the input space. The ranges should be selected so that each range represents a different execution path. Each combination of ranges then would represent a run type. Partitioning in this way provides the framework for sampling non-uniformly across the input space. If operations are selected randomly according to the operational profile and input states drawn randomly from the input space, non-uniform random tests matching operational profile will be selected. We judge the reliability of a program by the output states (sets of values of output variables created) of its runs. As run represents a transformation between an input state and an output state [14]. Multiple input states may map to the same output state, but a given input state can have only one output state. The input state uniquely determines the particular instructions that it will be executed and values of their operands. Thus, it establishes the path of control taken through the program. It also uniquely establishes the value of all intermediate variables. Whether a particular fault will cause a failure for a specific run type is predictable in theory.

4.5.3. Reducing the number of operations.

The operational profile may create an unrealistic set of tests because the list of operations is too long. There are at least three ways to restructure it:

1. Reduce the number of run types.
2. Increase the number of run types grouped per operation.
3. Ignore the remaining set of run types expected to have total occurrence probability appreciably less than the failure intensity objective.

The number of run types can be reduced by reducing either the size of the input variable list or the number of levels of the input variables. To potentially reduce the number of input variables, one of the following can be performed:

1. Reduce functionality.
2. Reduce the number of possible hardware configurations.
3. Restrict the environment the program must operate in.
4. Reduce the number of fault types.
5. Reduce unnecessary interactions between successive runs.

The first four of these alternatives change the system's features, thus impacting the customer and reducing flexibility, robustness, and possibly reliability. The fifth option, however, is a desirable one. Design changes can be implemented which should not impact functionality for the customer.

4.5.6. Methods for reducing run interactions are as follows:

1. Minimize the input variables that application programs can access at any one time.
2. Reinitialize variables between runs.
3. Use synchronous, as opposed to asynchronous, design.

Although reducing interactions can effectively reduce input space, it adds another level of complexity. It is much more risky than the other approaches to reducing input space.

4.5.7. Occurrence Probabilities.

After each input variable is partitioned into ranges, probabilities associated with each range for each variable must be identified. In some instances field data may already exist on the frequency of key input variable ranges. It is highly recommended that an attempt to find this data or

data on a similar system be undertaken. In the long run, this may be more cost effective than attempting to estimate probabilities with no usage background. Validity of reliability estimates is directly related to the proximity of the probability estimates to the actual occurrences in the field. To minimize the risk of obtaining inaccurate reliability estimates, start by taking measurements of input variable ranges on an initial release of a system, updating the occurrence probabilities used for testing associated with later releases if it is determined that the estimates made during testing differ from those observed in a target environment in the field. Beta testing may be useful. The initial estimation effort should be performed by an experienced systems engineer or smeone who has a thorough understanding of both the system and user needs. Experienced users should be interviewed to verify that estimates are within reason.

It may also be helpful to create an interaction matrix of input variables plotted against other key input variables. The matrix should reveal combinations of variables that do not occur or contain dependencies. The remainders of the matrix contain independent combinations where the estimates of occurrence probabilities are the product of individual input variable probabilities.

4.6. Example of Data-driven system:

Financial and billing systems are commonly data driven. Suppose a cable television billing system was designed as an account processing system. This system processes the charge entries for each account for the current billing period and generates bills. The reliability to evaluate is the probability of generating a correct bill. This involves determining the reliability over the time required to process the bill and its entries.

Assume that the design was not anticipated when the functional profile was developed, so the relationship between the functional profile and operational profile is complex. For instance, typical functions might have been bill processing, bill correction, and delinquency identification.

The account-processing system has an operational profile that relates to account attributes. Its operations are classified by customer type (business or residential), service type (basic, expanded basic, premium package), and payment status (paid, delinquent).

Assume that 90 percent of the customers are residential and 10 percent are businesses. Forty percent of the customers subscribe to the basic cable service. Half of all customers receive expanded basic, and the remaining 10 percent pay for the full premium package. History shows that 2 percent of the accounts are delinquent, on average.

Table shows the set of operations and the associated probabilities.

Table 5 : Operational Profile for Account-Processing Billing System

Operation	Occurrence Probability
Residential, Expanded Basic, Paid	0.4410
Residential, Basic, Paid	0.3528
Residential, Premium, Paid	0.0882
Business, Expanded, Paid	0 0490
Business, Basic, Paid	0.0392
Business, Premium, Paid	0.0098
Residential, Expanded, Delinquent	0.0090
Residential, Basic, Delinquent	0.0072
Residential, Premium, Delinquent	0.0018
Business, Expanded, Delinquent	0.0010
Business, Basic, Delinquent	0.0008
Business, Premium, Delinquent	0.0002

This concludes the general discussion on operational profiles.

5. Conclusion :

The operational profile plays a central role in commonly used reliability prediction models. However the exact operational profile of a system is usually not known during the testing and certification process before actual system usage. In best circumstances where extensive empirical and/or theoretical information on operational profile is available, only a good approximation of operational profile can be determined. In practice, there will always be uncertainty; hence the applicability of reliability prediction method is in doubt when system operational profile cannot be accurately determined.

References :

1. Jiantao Pan: *"Software reliability. Carnegie Mellon University,"* 18849b Dependable Embedded System, 1999.
2. Pankaj Jalote: *"An integrated Approach to Software engineering, Second Eddition,"* Nrosa Publishing House 2002.
3. Dr.K.K.Agrawal, *"Dr.Yogesh Singh Software Engineering, Third Edition chapter 9"*, page 309.
4. Musa J.D., A.Iannino, K.Okumoto, *"Software Reliability Measurement, Prediction and Applications"*, McGrtaw-Hill Book Company, NY., PP. 183-185, 1987.
5. Musa, John D;Iannino, A.; Okumoto, K; *"Software Reliability Measurement"*, Prediction, Application, McGraw-Hill, 1987
6. J.P. Cavano and J.A. McCall. : *"A Framework for the measurement of software quality. Proceeding ACM software quality assurance workshop"*, page 133-139 ACM Nov 1978.
7. Dr.R.A.Khan Software Quality : *"Concept and Practices, Nrosa Publishing House"*, (Software Reliability Estimation.).
8. Software confidence for the Digital Age :Research Resources, Cigital Labs.
9. J.D.Musa , G.Fuoco, N.Irving, et, al.: *"The operational profile. Handbook of Software Reliability Engineering"*, 1996, pp, 167-216 McGraw Hill
10. The Operational Profile in Software Reliability Engineering: An Overview, Dr. John D. Musa.
11. Lyu, Michael R. *"Handbook of Software Reliability Engineering"*, IEEE Computer Society Press, 1996.
12. Musa, J.D., *"Operational Profiles in Software Reliability Engineering,"* IEEE Software Magazine, March 1993, p.14-32.
13. Musa, J.D., *"Operational Profiles in Software Reliability Engineering,"* IEEE Software Magazine, March 1993, ã 1993 IEEE, reprinted with permission
14. Dr J.D.Musa, *"Software Reliability Engineering: more reliable software faster and cheaper"*, Chapter 2: Implementing operational profiles.
15. D. Dyer, :Probability Models: *"Operational profile testing, J.Marciniak, ed. Encyclopedia of software engineering"*, John Wiley & Sons pp 824-837, 1994.

Author's Biograpy

Wali Ullah, is working as Lecturer, Department of Computer Science, Mumtaz PG College Lucknow, University of Lucknow, since Aug-2003 onwards. He received his Master (MCA) degree (1994) from Institute of Engineering and Technology Lucknow. He is doing Ph.D. ("Software Reliability Estimation: An Operational Profile Perspective") from Integral University, Lucknow. He has About ten years teaching experience.At present he is working in Mumtaz P.G. College Lucknow as Lecturer in Department of Computer Science since Aug-2003 onwards. Worked with Baba Saheb Bhimrao Ambedkar University Lucknow as a "Guest Faculty ", Department of Information Technology from January 2008 to December 2009. Worked with Islamia Degree College Lucknow as Lecturer in Department of Computer Science, from Sep-2001 to Aug-2003.

He has also About seven years Industrial experience.Worked with M/s. Diviya Chemicals Ltd. Roha Raigad (MS) as "EDP Executive", from Sep-95 to Sep-2001. Worked with Infosis Computer System from Aug-94 to Sep-95 as Instructor / Programmer.

Developments: Payroll System, Inventory Control System.

Favorite Papers : Microprocessor, Visual Basic, Data Structure, C, C++, Operating System , Computer Architecture & Organization, Software Engineering, Cryptography and Network Security, Database Management System.

Dr. Surya Prakash Tripathi is an Assistant Professor and head in the Department of Computer Science & Engineering, Institute of Engineering & Technology Lucknow.

He received his M.Sc. (Matamatics) degree from Allahabad University, M.Tech. (Computer Science) degree from I.I.T. Delhi, and Ph.D. (Computer Science & Engineering) from University of Lucknow. He has about 23 years teaching experience in the Department of Computer Science & Engineering. Seven Research scholars are working under his supervision. He has published about 25 papers in the reputated journals. Dr.S.P.Tripathi has extensively worked in various fields of electronics and Computer Engineering. Besides, contribution to the academic field Dr.S.P.Tripathi has been associated with several organizations. He established many workshops. He is member of International Association of Computer Science and Information Technology, Singapore, life member of Computer Society of India, life member of Indian Science Congress Association, Member Editorial Team, Anjaneya Lucknow Revue, Sardar Bhagat Singh College of Technology and Management Lucknow, started January 2009 a quarterly Journal.

Dr. Riyaz Ahmad Khan is an Associate Professor and head in the Department of Mathematics, Integral University

Lucknow. Presently he is working as a lecturer, Nizwa College of Technology Nizwa Oman. He received his M.Sc. (Mathematics) and Ph.D. (Mathematics) degree from Avadh

University Faizabad. He also served as Chief Proctor. Presently Supervising Six Ph.D. students in the area of "***Fluid Dynamics and Differential Geometry***", one has been awarded for Ph.D. degree and one is to be likely awarded for the same He is also supervising two Ph.D. students of Computer Science in the field of Software Engineering. He has about 15 years teaching experience in the Department of Mathematics. He has published two papers, four papers are accepted, while four papers are communicated in the reputed journals. He attended many conferences. He is also member of International Egyptian Engineering Mathematical Society, Egypt. Dr. Riyaz Ahmad Khan is a prolific writer, he wrote nine mathematics book for B.Tech., Biotech., and Pharmacy courses. He also attended many short term courses. He was also Member of executive council of Integral University, member of Academic council of Integral University, Lucknow. He is Convener of Departmental Research Committee from 2006 onwards.