Re-Certification of Software Reliability without Re-Testing

C. Wohlin

Dept. of Communication Systems, Lund Institute of Technology, Box 118, S-221 00 Lund, Sweden, e-mail: claesw@tts.lth.se

Abstract

Usage testing or operational profile testing is depicted as an important test technique to remove the most critical faults from an operational perspective. The technique also allows for determination of the software reliability through application of software reliability models. This is, of course, beneficial, but a problem arises as the usage changes and this can be expected as new services are added to the existing system or the behaviour of the users change due to some reason. Therefore, a method to re-certify software reliability without re-testing the software for all potential changes that may occur is needed. This paper outlines such a procedure based on fault content estimations and by recording a number of measures during usage testing. The procedure gives a basis for determining whether more testing or other means to remove faults is needed prior to the usage changes or if the reliability requirements are fulfilled even after a change in usage. It is concluded that the proposed procedure can be a valuable tool to cope with changes in usage, but more research as well as practical experience from the proposed procedure is needed.

Keyword codes: D.2.0, D.2.5, D.2.8
Keywords: Software Engineering, General; Software reliability, usage testing, operational profile testing, reliability certification, software metrics, fault content estimations, statistical quality control.

1. INTRODUCTION

A reliability estimate based on intended usage gives a good figure of the perceived reliability during operation. The usage profile is used to get an estimate of the reliability experienced during operation. Several profiles may be used during usage testing if different usages are expected based on, for example, the location of the installation or different customers, [1]. The estimate of the reliability based on usage is still the only way to get a realistic estimate, but the change in usage compared to the usage profile applied when estimating the reliability must be handled in some way.

This paper tries to describe one possibility to obtain an estimate of the reliability when the usage is changing. Some different reasons to experience a change in usage compared to the usage profile applied during certification can be identified:

- An erroneous profile was applied during the certification. The change will be experienced as the software is being released.
- A change will occur with the time, either slowly or perhaps quickly due to for example marketing of some specific services

This paper mainly discusses the second item.

Unfortunately, the estimate from usage testing will probably give an optimistic estimation, since if the operational usage is different from the one applied during testing, the reliability will probably be overestimated. The reason is that the certification will be more thorough in the areas where the main usage is supposed to be, but if this is wrong the software will be used more in the less tested regions. Thus it is reasonable to believe that the reliability will drop.

The objective with certification must be remembered, i.e. to certify a specific reliability level, in particular the level that will be perceived during operation. This is done by performing usage testing. The certification objective is not to find faults in general, but to indicate their absence in the frequently executed parts or to locate the faults influencing the reliability the most. Testing, as discussed here, is not primarily supposed to be seen as a fault removal procedure.

2. RELIABILITY ESTIMATE FROM USAGE

Some different opportunities of specifying usage exist, see for example [1, 2, 3]. These methods resemble each other, but have some different properties. The approach discussed in [3] leads to a state explosion for large systems, and the solution proposed in [2] is a further development of the ideas discussed in [3], while the solution proposed in [1] is very similar to the one proposed in [2]. One major difference is the way of modelling the external states. These are modelled explicitly in [2] and more implicitly in [1]. We will here primarily discuss the method presented in [2], which is based on a hierarchical Markov chain.

A usage specification consists of a usage model and a usage profile. The usage model is a structural model of the anticipated usage, while the profile quantifies the actual usage.

In the hierarchical Markov chain model, it is proposed that the probabilities for choosing a particular path in the hierarchy are dynamic, i.e. the probabilities in the hierarchy are changed based on the state of the Markov chain on the lowest level. This level describes the actual state of an external user. The dynamic change of probabilities is a way to capture the difference in load, primarily in terms of capturing the functional interaction that will occur depending on how many users that are in different states. The dynamic behaviour will, however, change around some equilibrium probabilities. A simpler solution in the hierarchical model will therefore be to assign static probabilities in the hierarchy, i.e. the probabilities are independent of the actual states of the external users.

This means that the usage model can either be used with static or dynamic probabilities. If a possible future change in usage shall be modelled it seems unnecessary to model the probabilities dynamically, because of the uncertainty in the future usage. Dynamic probabilities are, however, recommended during usage testing with the expected usage profile, as it, for example, allows that it is more probable that a user who has started a specific sequence continues than that a new specific user enters the system. One way of estimating the reliability based on change in the usage profile is to test the software with a new profile. Thus try to estimate the reliability based on a future possible change in the usage. This approach is feasible but very costly, since it would mean re-testing the whole system as new services are introduced or the usage is expected to change. In particular for large systems which have a long life time and a continuously change of available services, an example of this type of system is a telephony exchange. Another solution would be to make a re-calculation based on the expected future change in usage. The re-calculation is based on the failure data when applying the original usage profile, estimation of fault content and the new profile. This approach is much more tractable and it is the method studied in this paper.

But, before presenting the method, it is essential to realize that even if the usage profile may be slightly incorrect or slowly changing over time, it is better to apply usage testing than to continue performing coverage testing. This is for example emphasized with the results presented in [1, 4].

3. RE-CERTIFICATION OF SOFTWARE RELIABILITY

3.1 Introduction

It is advantageous to make an estimation based on a re-calculation instead of applying new usage profiles in testing. The re-calculation shall be based on the failure data obtained from the usage profile applied during testing, fault content estimations and the new usage profile. A re-calculation procedure would make it possible to calculate the reliability for different profiles without performing new tests. The re-calculation gives an estimate of the reliability after the change in usage has occurred. Hence, providing a valuable input to decide whether the software fulfils the reliability requirements after the change in usage. If it is found that the requirement is not fulfilled then a decision to perform more testing before the change actually occurs can be taken. This means that decisions can be taken based on actual information and not by guessing.

The main problem in reliability estimation is that some faults may be hidden in the software, which were not found in the original usage test but are critical if the usage changes. This is particular problematic if the usage changes dramatically, i.e. a use case which has been very little tested due to little usage in the original case becomes a use case that is used extensively. This will probably make the reliability drop considerably. Therefore, the re-calculation procedure must be based on fault content estimations from either complexity metrics, as discussed in for example [5], or by application of capture-recapture methods as discussed in [6, 7].

3.2 Re-calculation procedure

Several models and measures are needed for the re-calculation procedure. Some of these are normally recorded when performing usage testing, they are:

- Usage model
- Usage profile
- MTBF (Mean Time Between Failures) values

These have to be complemented to allow for a re-calculation, which requires the following models and measures:

Unchanged model:

- The usage model is the same as in the original application. It is assumed that the usage model has not changed as the usage profile changes.

New measures to record:

- Fault content estimations from either complexity metrics or capture-recapture methods must be performed, which primarily must be applied prior to the testing phase to "know" the fault content in different system parts when entering the test phase.
- The locations of the faults found during usage testing must be recorded.
- The probability for failure if the part containing a fault is used has to be determined. It may be the case that a failure does not occur even if the faulty part of the usage model is exercised. This can be determined based on the observed failures and the test cases run in usage testing.
- The mean execution time for a arbitrary transition within a specific service or component as seen from the usage model has to be recorded. This parameter can also be recorded from the test cases executed. The mean execution time shall be used to re-calculate the MTBF values. It is assumed that, if a failure occurs in a service it will take half the execution time to reach the fault.

New input:

- The new usage profile is the reason for performing the re-calculation and it is, of course, an essential input.

During usage testing, it is essential that the usage model is correct and that the profile is as good as possible. The original profile applied results probably in a number of failures. These failure times are recorded, i.e. the MTBF values are noted. This is the normal procedure when applying statistical usage testing.

The re-calculation procedure requires that the fault content prior to usage testing can be estimated and that the actual location of the faults resulting in failures shall be noted. The time to a failure is based on a particular execution. Thus meaning that new MTBF values can be calculated based on the new usage profile and the actual times to reach the parts in the usage model containing the unobserved faults. This also includes the probability for fault detection if the part containing a fault is executed. The probability shall be used in determining whether a fault is found or not, when a part containing a fault is executed. The probability is a measure of fault exposure.

Due to this, it is necessary to record the execution times for different usage parts, e.g. a transition within a service provided to the users. All times to execute a specific abstraction level in the usage model must be recorded, e.g. the services provided to the users or some suitable component level. This figure must be noted, since it has to be used when re-calculating MTBF values based on a new usage profile.

The re-calculation shall be applied, as stated above, when the usage changes or is expected to change. Hence, it is assumed that certification has been performed for one usage profile and a reliability measure has been calculated. The application of re-calculation means that it is possible to prepare for different outcomes in the future. Re-calculation is needed as the change in usage will mean that the perceived reliability changes. The objective is to estimate the new reliability from the experience stored during prior usage tests and the new usage profile.

The re-calculation procedure can be summarized in the following steps:

1. The software reliability requirement to certify is known from the requirements specification.
2. The fault content for different system parts when entering the test phase is estimated applying complexity metrics or capture-recapture methods on, for example, the code.
3. Fault location in the usage model is recorded during testing, hence being able to derive the number of remaining faults in the different system parts.
4. Mean execution time for a suitable level in the usage model is recorded at the testing time.
5. Probability for fault exposure is determined based on experience from usage testing.
6. MTBF values are generated by randomly running through the usage model and adding successful execution time to the current MTBF value, until a fault is reached and executed.

An example is presented in the next section to illustrate the steps in the re-calculation procedure.

3.3 An example

The six step procedure in the previous section will be gone through for a minor example to show how the procedure is intended to work.

Step 1: The software reliability requirement in terms of MTBF is known to be 50 t.u. (time units).

Step 2: The fault content is supposed to have been estimated from phases prior to the test phase. The parts in which the failures reside are assumed to be services in for example a telecommunication system. The following estimates are assumed, based on applying capture-recapture on the code inspections, Service 1: 4 faults, Service 2: 2 faults, Service 3: 5 faults and Service 4: 4 faults.

Step 3: During the ordinary certification, it is assumed that a total of 7 failures were found and assumed corrected, hence leaving an estimate of 8 remaining faults in the software. The usage testing revealing these 7 faults is assumed to result in certification of the required reliability level, i.e 50 t.u., based on the expected usage profile. The usage model and the location of the remaining faults are depicted in figure 1, as well as the new usage profile to be used in the re-certification. The new profile describes an expected change.

The usage modelling technique depicted in figure 1 is further discussed in [2] and the correspondence between system parts, for example services, and model parts is further discussed in [8].

The new usage profile for which the reliability shall be calculated has to be known in terms of the probability for selecting user type 1 from the usage state etc. These probabilities are shown in figure 1. Thus it is possible to run through the usage model by starting in the usage state and then running down in the structure until a specific service is executed.

Step 4: The execution times for each service are supposed to be known from measurements on the system during testing. The times are: Service 1: 2 time units (t.u.); Service 2: 1 t.u.; Service 3: 2.5 t.u. and finally Service 4: 4 t.u. It is assumed that it is not certain that a failure occurs just because the service is executed. Probabilities for fault exposures are given in step 4. It is also assumed that if a failure is executed it will in average happen after half the execution time of the service.

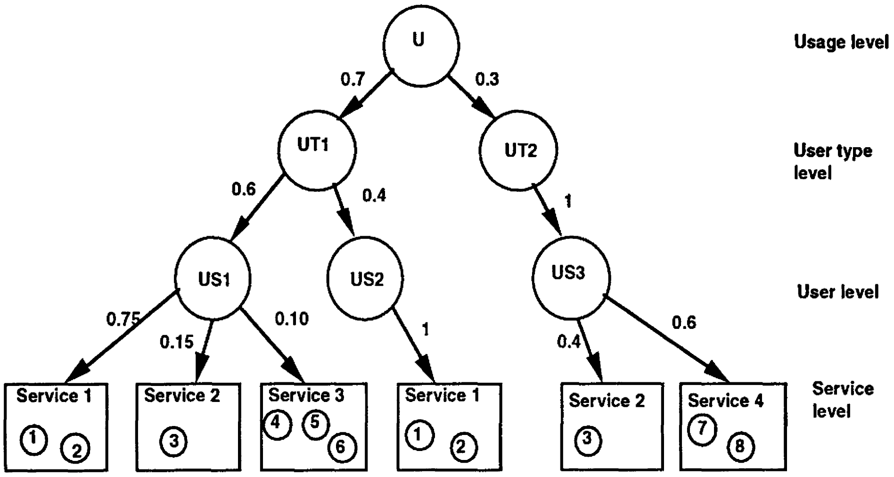

Figure 1. Usage model and new profile with estimated faults remaining.

Step 5: The following fault exposure probabilities are assumed based on the outcome of the usage test performed: Fault 1, 0.05; Fault 2, 0.20; Fault 3, 0.50; Fault 4, 1; Fault 5, 0.30; Fault 6, 0.01; Fault 7, 0.75 and Fault 8, 0.25. The fault exposure probabilities are determined based on the number of test cases run and the faults found in comparison with the estimated number of faults present in the different services.

Step 6: Random numbers are generated to run through the usage model and also to determine whether an actual execution of a service result in a failure or not. The MTBF values are recorded. The outcome became:
Fault no. 4: MTBF(1) = 10.25 t.u.,
Fault no. 2: MTBF(2) = 1 t.u.,
Fault no. 3: MTBF(3) = 6.5 t.u.,
Fault no. 7: MTBF(4) = 20 t.u.,
Fault no. 8: MTBF(5) = 30 t.u.,
Fault no. 1: MTBF(6) = 27 t.u.,
Fault no. 5: MTBF(7) = 75.75 t.u.

Finally, the certification time without failure is found to be 161.5 t.u., i.e. fault no. 6 was not found. The latter indicating that a longer random number sequence has to be generated to find the last known fault. An example of how the outcome was generated is presented in appendix A.

It must be noted that based on the probabilities, the expected order to find the faults are: 7, 2, 3, 8, 4, 1, 5 and 6. The rank correlation between the expected order and the actual outcome in this particular case is 0.69.

These six steps have given new MTBF values. It is seen from these values that if the requirement of MTBF is 50 t.u., then the requirement is not fulfilled and further testing has to be conducted to locate the faults remaining in the software before the usage changes as expected.

3.4 Discussion

A major question remains to be answered: Does this procedure generate the true reliability perceived by the users after the change in usage? The answer to this question is that by applying the procedure it is possible to gain experience from the re-calculation procedure. It is not until it has been tested that the question can be fully answered. Further research is, however, needed to improve both fault content estimations as well as the proposed re-calculation procedure.

The main advantage with this method is that no re-testing has to be performed. The procedure forms a basis for deriving new MTBF values and hence gives an input to taking an informed decision. It can hence be determined whether more testing is needed to remove more faults or if the performed usage test if sufficient to fulfil the software reliability requirement after the usage changes.

It can be concluded that the re-calculation procedure is advantageous compared to applying a new profile directly, since it may not be necessary to re-test. The result from the re-calculation procedure is either that the reliability requirement is still fulfilled or new efforts to locate the faults have to initiated

The re-calculation procedure is a tractable method, hence indicating that it ought to be used because of its simplicity and to gain experience. More research has however to be conducted in the area to make it really useful.

4. CONCLUSION

The change in usage may cause a large fall in the reliability and change in usage is expected over time. It is therefore essential to be able to predict how the change in usage will affect the perceived reliability before it occurs in the operational phase.

The first step is of course to understand that the reliability will change as the usage changes. Then the method presented in this paper can be used to gain experience and then it can be improved. Thus giving a better method of predicting the reliability as the usage changes.

The method presented in this paper is not the optimum method, but possibly the best method available. Further research is needed in the area based on the ideas presented. The method shows, however, that it ought to be possible to predict the reliability based on change in usage, instead of waiting until a fall in reliability is experienced in the field.

5. FUTURE RESEARCH

5.1 Introduction

One disadvantage with usage testing is that correct test cases do not improve the reliability. They only contribute to show that the software has a certain reliability. This is a clear waste of resources, since if it is correct there is no use testing it. To completely eliminate correct test cases seems impossible, but they at least ought to be minimized without losing the opportunity to certify a specific reliability level of the software. It is also well-known that it is impossible to certify the reliability levels required within the available test period, in particular this is the case for safety-critical systems.

5.2 Requirements certification method

What are the characteristics of a good test method?

- From the developer´s point of view, as many as possible of the faults must be found.
- From the procurer´s viewpoint, the software product must be reliable (no faults at all in the software would be preferable)
- The number of erroneous test cases under a given time interval ought to be maximized.
- The length of the test interval must be minimized.
- It would be favourable if the total number of faults and their location could be estimated.
- The software ought to be released with a given reliability (within confidence bounds).
- Future failure occurrences ought to be predicted.

- A certain degree of coverage of the code ought to be obtained.
- Automatic control if the test cases go right or wrong would be beneficial.

Does this test methodology exist? The answer is clearly no.

The objective of the future research is to improve the existing test methods so they better fulfil the characteristics outlined above.

ACKNOWLEDGEMENT

I am grateful to Jan-Eric Johansson, Telia (Swedish Telecommunication Administration), Sweden for granting permission to base this research on a prestudy conducted by Telia.

REFERENCES

1. Musa, J. D.: 'Operational Profiles in Software Reliability Engineering', IEEE Software, 1993, Vol. 11, No. 2, pp. 14-32.
2. Runeson, P., and Wohlin, C.: 'Usage Modelling: The Basis for Statistical Quality Control', Proceedings 10th Annual Software Reliability Symposium, Denver, Colorado, USA, June 1992, pp. 77-84,.
3. Whittaker, J. A. and Poore, J. H.: 'Markov Analysis of Software Specifications', ACM Transactions on Software Engineering and Methodology, 1993, Vol. 2, No. 1, pp. 93-106.
4. Adams, E. N.: 'Optimizing Preventive Service of Software Products', IBM Journal of Research and Development, January 1984.
5. Munson, J. C., and Ravenel, R. H.: 'Designing Reliable Software', Proceedings Fourth International Symposium on Software Reliability Engineering, Denver, Colorado, USA, November 1993, pp. 45-54,.
6. Eick, S. G., Loader, C.R., Long, M.D:, Votta, L.G and Vander Wiel, S. A.: 'Estimating Software Fault Content Before Coding', Proceedings 14th International Conference on Software Engineering, Melbourne, Australia, 1992. pp. 59-65.
7. Vander Wiel, S.A., and Votta, L.G.: 'Assessing Software Designs Using Capture-Recapture Methods', IEEE Transactions on Software Engineering, 1993, Vol. 19, No. 11, pp. 1045-1054.
8. Wohlin, C., and Runeson, P.: 'Certification of Software Components', IEEE Transactions on Software Engineering, 1994, Vol. 20, No. 6, pp. 494-499.

APPENDIX A: EXAMPLE OF RE-CALCULATION

The objective of this appendix is to illustrate the derivation of MTBF values for a new usage profile, see section 3.3, step 6.

1. Random numbers are generated. For example: 0.4077, 0.0433, 0.3966, 0.6597 and 0.7538. A total of 450 random numbers were generated to get the MTBF values presented in section 3.3.
2. The five random numbers give a specific sequence in the usage model, see figure 1. The following is obtained:
   0.4077 < 0.7 => transition U -> UT1
   0.0433 < 0.6 => transition UT1 -> US1
   0.3966 < 0.75 => transition US1 -> Service 1 (i.e. service 1 is executed)
   Two faults are located in service 1. It is now necessary to find out if these are found or not. The probabilities for fault exposure are given in section 3.3, step 5.
   0.6597 > 0.05 => Fault 1 is not found.
   0.7538 > 0.20 => Fault 2 is not found.

3. Thus the total execution time so far is equal to the execution time of Service 1, let the execution time be denoted S1.
4. New random numbers are generated and the usage model is run through from state U over and over again until a fault is found.
5. The first MTBF value is determined based on the sum of the execution times of the services until the first failure occurs. The MTBF becomes:
   MTBF = S1 + S2 + S1 + S1+ S1 + S3/2.
   It was assumed that half the execution time occurred before the fault was encountered, hence explaining S3/2. It was fault number 4 that was found, which is located in service 3.
6. The actual value of the MTBF is determined from the execution times of the services, see section 3.3, step 4.
   MTBF = 2 + 1 + 2 + 2 + 2 + 2.5 / 2 = 10.25 t.u.
7. The usage model is run through until 7 failures have been found and a long execution time has passed without encountering the last known fault. The MTBF values are given in section 3.3, step 6.