C. Wohlin and P. Runeson, "Certification of Software Components", IEEE Transactions on Software Engineering, Vol. 20, No. 6, pp. 494-499, 1994.

Certification of Software Components[1]

Claes Wohlin[2] (e-mail: claesw@tts.lth.se)

Per Runeson (e-mail: pr@q-labs.se)

Q-Labs, IDEON Research Park

S-223 70 LUND, SWEDEN

Phone: +46–46–182980, Fax: +46–46–152880

Abstract

Reuse is becoming one of the key areas in dealing with the cost and quality of software systems. An important issue is the reliability of the components, hence making certification of software components a critical area. The objective of this article is to try to describe methods that can be used to certify and measure the ability of software components to fulfil the reliability requirements placed on them.

A usage modelling technique is presented, which can be used to formulate usage models for components. This technique will make it possible not only to certify the components, but also to certify the system containing the components. The usage model describes the usage from a structural point of view, which is complemented with a profile describing the expected usage in figures. The failure statistics from the usage test form the input of a hypothesis certification model, which makes it possible to certify a specific reliability level with a given degree of confidence. The certification model is the basis for deciding whether the component can be accepted, either for storage as a reusable component or for reuse.

It is concluded that the proposed method makes it possible to certify software components, both when developing for and with reuse.

---

1. This work has been carried out within the ESPRIT-2 project #7808 REBOOT (Reuse Based on Object-Oriented Techniques).

2. From 1 July 1993 at the Department of Communication Systems, Lund Institute of Technology, Lund University, Box 118, S - 221 00 Lund, Sweden.

Keywords:

Reuse, Software components, Certification, Usage testing, Operational profile testing, Usage modelling, Usage profile, Hypothesis certification

1. Introduction

Interest in software reliability engineering is growing rapidly with particular focus being set on usage testing [1] or operational profile testing [2]. An important issue in software engineering is reuse, which seems to be one of the key factors in coping with the cost and quality of software systems [3]. To fulfil the objective of reuse, it is necessary to have reliable reusable components. The reliability of the components can be certified with usage testing by developing usage models, applying usage profiles and finally applying a certification model to show the level of confidence in them. This article focuses on combining experience from usage testing with the need for reliability engineering in the reuse community.

The basis for reuse is the reliability of the components intended for reuse and the gains achieved through its application. This means that the components developed for reuse must have a quality stamp concerning, for example, reliability and performance. This is achieved by continuous evaluation, in parallel with the development process for reuse. The process must include methods and models for quantifying the characteristics of the reusable components. In particular, one important issue is to establish methods of certification, i.e. both to certify a component being developed, as well as being able to rely on a reusable component. The development of new software components to be reused is referred to as development *for reuse*, whilst the development reusing components stored in the repository is referred to as development *with reuse*. The term component is used in a generic sense. A component may be, for example, a system, sub-system or system service.

The focus in this article is on usage modelling, usage profiles and the certification of components by applying a hypothesis certification model. A more detailed version of this paper is available from the authors, [4].

2. Certification and reuse

2.1 Introduction

Object-oriented techniques make it possible to develop components in general, and to develop reusable components in particular. These components must be certified regarding their properties, for example their reliability.

A component developed for reuse must have reliability measures attached to it, based on one or several usage profiles. The objective of the certification method discussed below is to provide a basis for obtaining a reliability measure of components. The reliability measure may either be the actual reliability or an indirect measure of reliability such as MTBF (Mean Time Between Failures).

During development for reuse, a usage model must be constructed in parallel with the development of the component. The usage model is a structural model of the external view of the component. The probabilities of different events are added to the model, creating a usage profile, which describes the actual probabilities of these events. The objective is that the components developed will be certified before being put into the repository. The component is stored together with its characteristics, usage model and usage profile. The reliability measure stored should be connected to the usage profile, since another profile will probably give a different perceived reliability of the component altogether.

Development with reuse involves retrieving components from the repository, and at the retrieval stage it is necessary to examine the reliability of the components being reused. The components have been certified using the specific profiles stored, and if they are to be reused in a different environment with another usage profile, they must then be certified with this new usage profile.

A method of certification can be described in the following steps: 1) *Modelling of software usage*, 2) *Derivation of usage profile*, 3) *Generation of test cases*, 4) *Execution of test cases and collection of failure data* and 5) *Certification of reliability and prediction of future reliability.* The method can be applied to certification of components as well as to system certification.

2.2 Component certification

The components must be certified from an external view, i.e. the actual implementation of the component must not influence the certification process. The estimation of usage probabilities must be as accurate as possible. It may in many cases, be impossible to exactly determine the usage profile for a component. This will be problematic, especially when the individual component is indirectly influenced by the external users of the system. It must, however, be emphasized that the most important issue is to find probabilities that are reasonable relative to each other, instead of aiming at the “true” probabilities, i.e. as they will be during operation.

The reuse of components also means that the usage model of the component can be reused, since the usage model describes the possible usage of a component (without probability estimates of events being assigned). This implies that the structural description of the usage can be reused, even if the actual usage profile can not. The problems of component reuse and model reuse for different cases are further discussed in section 3.5. Component certification is also discussed by Poore et al. [5].

3. Usage modelling

3.1 Introduction

This section discusses usage models and usage profiles for software systems as a whole, as well as for individual components, and an illustration is given in section 5 by means of a simple telecommunication example. A system may be seen as consisting of a number of components. A component is an arbitrary element which handles coherent functionality.

Usage models are intended to model the external view of the usage of the component. The user behaviour should be described and not the component behaviour. The users may be humans or other components. Modelling usage of software components includes problems which do not arise when modelling the usage for systems as a whole. The primary users of a component are those in the immediate vicinity, for example other components. But in most cases there are other users involved, for example end-users, which indirectly affect the use of actual components. Therefore, the usage of a component may have to be derived from an external user of the system, even if the user does not communicate directly with the component.

It is assumed that the usage models are created in accordance with the system structure to support reuse. The view is still external, but the objective is to create usage model parts which conform to the structure of the system. The reuse of components also means that it will be possible to reuse the usage model describing the external usage of that particular component. The usage models of the components may combine in a way similar to the components´ combination within a system, providing services to an external user.

Different usage profiles may be attached to one usage model.

3.2 Usage model

Markov chains as a means of modelling usage are discussed by Whittaker and Poore [6]. The use of Markov chains has several advantages, including well-known theories. A main disadvantage is, however, that the chain grows very large when applying it to large multi-user software systems, [7]. The objective of the usage model is to determine the next event, based on the probabilities in the Markov chain. The chain is used to generate the next event without taking the time between events into consideration, which means that the times between events are handled separately and an arbitrary time distribution can be used.

A hierarchical Markov model is introduced, the State Hierarchy model (SHY), to cope with this disadvantage, known as the state explosion problem [7]. The SHY model can describe different types of users, i.e. human users, other systems, system parts, and multiple combinations and instances of the user types. During the development of the usage model the user types are handled and constructed separately, and they are composed into a usage model for the system as a whole. The model, being modular, is therefore suitable for reuse, since the objective is to ensure a conformity between the usage model and the system structure, see section 3.1.

System configurations, for example, in different markets may differ in terms of user types and services available. Therefore, usage models for different system configurations, may be constructed, by combining the SHY models of the reusable components, and SHY models of the configuration-specific parts, hence obtaining SHY models for different system configurations. In particular, different services are one of the key component types to be reused. This implies that the certification is often related to the services potentially provided by the system.

The general principles behind the SHY model are shown in figure 1.

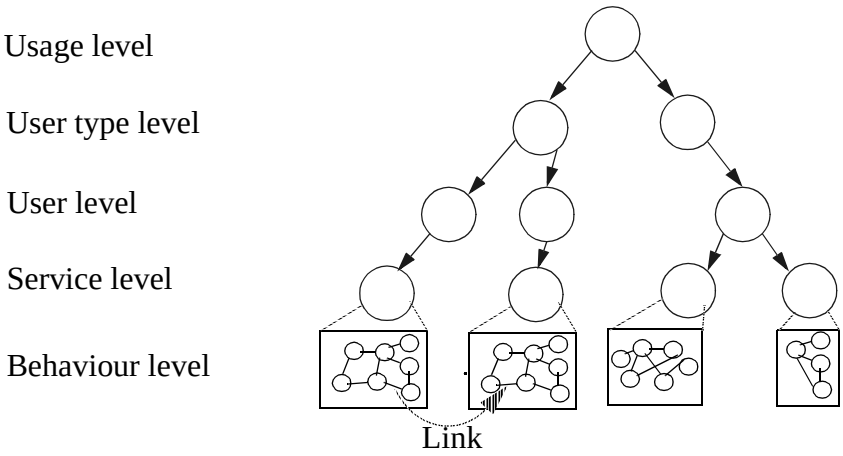

Figure 1. The State Hierarchy model.

The usage is divided into a hierarchy, where each part represents an aspect of the usage.

- The usage level is a state which represents the complete usage.
- The user type level contains user types or categories.
- The user level represents the individual users.
- The service level represents the usage of the services which are available to the user. The service usage description is instantiated for different users.
- The behaviour level describes the detailed usage of a single service as a normal Markov chain.

The interaction between different services is modelled as “links”, meaning a transition in one Markov chain on the behaviour level, causing a transition in another chain on the behaviour level. An example, A dials B, is shown in figure 2, where the transition from “idle” to “dial” for user A leads to a transition from “idle” to “ring” for user B.

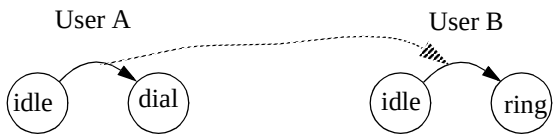

Figure 2. Link example, A dials B.

The model is discussed in more detail by Wohlin and Runeson [4, 7] and it is also used in the example in section 5, see in particular section 5.2.

3.3 Usage profile

The usage model is complemented with a usage profile, which assigns probabilities to every branch in the hierarchy, together with every transition on the behaviour level. The probabilities must be derived based on experience from earlier systems and expected changes concerning the actual system or expected usage of the system as it is marketed.

The probabilities in the hierarchy can be assigned static values, as in the example in section 5, or be dynamically changed depending on the states of the users in the model. The latter approach is needed to be able to model the fact that some events are more probable under special conditions. It is, for example, more probable that a user who has recently lifted a receiver will dial a digit, than that a specific user will lift the receiver. The use of dynamic probabilities in the hierarchy are further discussed by Runeson and Wohlin [7].

Test cases are selected by running through the SHY model. First, a user type is selected by random controlled selection, then a specific user is chosen, after which a service, available to the selected user, is drawn. Finally, a transition in the Markov chain on the behaviour level for the actual service is selected. This transition corresponds to a specific stimulus which is appended to the test script, and the model is run through again, beginning from the usage level, see figure 1 and section 5.4. The generation of a specific stimulus also means generating the data being put into the system as parameters to the stimulus, hence data is taken into account.

3.4 Usage profile and reuse

A component developed for reuse is certified with a particular usage profile for its initial usage and is stored in a repository for future reuse. The component is stored together with its characteristics, usage model and usage profile.

The reliability measure is attached to the actual usage profile used during certification, and since it is based on this particular profile it is not valid for arbitrary usage. The parts of the components, most frequently used in operation, are those tested most frequently, which is the key objective of usage testing. These parts are less erroneous, since failures found during the certification are assumed to be corrected. Another usage profile relating to other parts of the component, will probably give a lower reliability measure.

When developing with reuse of software components, it is necessary to compare the certified usage profiles with the environment in which the component is to be reused. If a similar profile is found, the next step is to assess whether the reliability measure stored with the component is good enough for the system being developed. If the component has not been certified for the usage profile of the new system, a new certification must be performed. The usage model stored with the component is used for the certification. After certification the new profile and the new certified reliability are stored in the repository with the component.

Objective measures of reliability for an arbitrary usage profile would be of interest in development with reuse. It is, however, impossible to record such measures, since the definition of reliability is: the probability of a device performing its purpose adequately for the period of time intended, under the operating conditions encountered. The component has therefore to be re-certified if it is reused under other operational conditions than initially profiled.

3.5 Reuse of the usage model

The proposed usage model itself can easily be reused. The extent of model reuse and how the model is reused depends on how the system or components of the system are reused. Some different reuse scenarios are presented for component certification, together with those in a system context.

Component certification:

- Reuse with the same usage model and usage profile

  The usage model and usage profile for a component can be reused without modification, if the component has been certified before being stored in the repository.

- Reuse of the usage model with an adjusted usage profile

  The usage model for a component can be reused, if the component is to be certified individually with a new usage profile. The certification can be obtained with the same model by applying the new usage profile. This gives a new reliability measure of the component, based upon the new expected usage.

- Reuse with adjustments in both the usage model and the usage profile

Adjustments in the structural usage of a component are made if the component is changed in order to be reused. The usage model must therefore be updated accordingly and a new certification must be made.

Reuse of components in a system context:

- Reuse of a component without modification

  The objective is to derive the usage model of the system, from the usage models of the components, when a system is composed of a set of components. This can, in particular, be achieved when the structural usage of the component is unchanged, but the probabilities in the usage profile are changed. It should be further investigated whether it is possible to derive system reliability measures from the reliability measures of the components, when the usage profiles for the components are unchanged. The main problem is the probable interdependence between components, which has not been assessed during component certification. This is an area for further research.

- Reuse of a component with modifications

  The change in the usage model is a result of the change or adaptation of the component. Therefore, the usage model of the individual component must be changed, thereby changing the usage model of the system. The system has either to be certified with the expected usage profile of the system, or the reliability of the system must be derived from the components. This problem is further addressed in [5].

- Changes to the existing system

  If an element of the system is changed, for example a component is replaced with another which has different functionality, the usage of the affected component is changed in the existing usage model. A new certification must then be obtained based on the new usage model.

Two factors concerning the SHY usage model make it suitable for reuse. First, the distinction between the usage model and the usage profile is important, since it facilitates the use of the same model, with different profiles, without changes. Secondly, the modularity of the usage model, and the traceability between the constituents in the usage model and components in the system, are essential from the reuse point of view.

3.6 Evaluation of the SHY model

Important aspects of the SHY model:

- Intuitive: The conformity between model parts and system constituents makes it natural to develop the model. This is particularly the case when a system constituent provides a specific service for the external user.
- Size: The model size increases only linearly with the increasing numbers of users, [7].
- Degree of detail: The model supports different levels of detail. The actual degree of detail is determined based on the system, the size of the components and the application domain.
- Dependencies: Functional dependencies are included in the model through the “link” concept.
- Reuse support: The model supports reuse, as stated in section 3.5, through its modularity and conformity to system constituents.
- Assignment of probabilities: The structure of the model helps to partition the problem into smaller parts, hence making it easier to derive transition probabilities.
- Calculations: It is theoretically possible to make calculations on the hierarchical model, by transforming it into a normal Markov model. However this is impossible practically due to the size of the normal model. Work is in progress to allow calculations to be performed directly on the SHY model.
- Generation of test cases: Test cases can be generated automatically from the model.

4. Certification of components

4.1 Theoretical basis

In the book by Musa et al. [8] a model for reliability demonstration testing is described. The model is a form of hypothesis certification, which determines if a specific MTBF requirement is met with a given degree of confidence or not. A hypothesis is proposed and the testing is aimed at providing a basis for acceptance or rejection of the hypothesis. The procedure is

based on the use of a correct operational profile during testing and faults not being corrected. The primary objective is, however, to certify that the MTBF requirement is fulfilled at the end of the certification, hence corrections during the certification process ought to be allowed. A relaxation of the assumption in the model of no correction after failure is discussed below. The hypothesis certification model is based on an adaptation of a sampling technique used for acceptance or rejection of products in general.

The hypothesis is that the MTBF is greater than a predetermined requirement. The hypothesis is rejected if the objective is not met with the required confidence and accepted if it is. If the hypothesis is neither accepted nor rejected, the testing must continue until the required confidence in the decision is achieved.

The hypothesis certification is performed by plotting the failure data in a control chart. Figure 3 shows failure number (r) against normalized failure time (t_{norm}). The failure time is normalized by dividing the failure time by the required MTBF.

The testing continues whilst the measured points fall in the continue region. The testing is terminated when the measure points fall in the rejection or the acceptance region, and the software is then rejected or accepted accordingly.

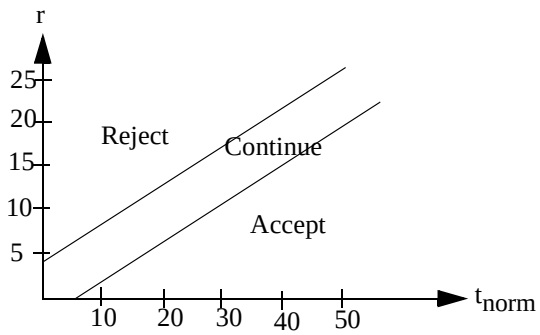

Figure 3. Control chart for hypothesis certification of the reliability.

The control chart is constructed by drawing the acceptance and rejection lines. They are based on the accepted risks taken for acceptance of a bad product and rejection of a good product. The calculations are described by Musa et al. [8].

Correction of software faults can be introduced by resetting the control chart at the correction times. It is not practical to reset the control chart after every failure, so the chart is reset after a number of failures has occurred. The reason for resetting the chart is mainly that after the correction, the software can be viewed as a new product.

It can be concluded that the hypothesis certification model is easy to understand and use. The hypothesis certification model provides support for decisions of acceptance or rejection of software products at specified levels of confidence.

If different failure types are monitored, for example with different severities, the failure data for each type can be plotted in a diagram and related to a required MTBF for specific types. The overall criterion for acceptance should be that the software is accepted after being accepted for all failure types.

The hypothesis certification model does not give any predictions of the future reliability growth. The certification can, however, be complemented with a software reliability growth model for that purpose.

4.2 Practical application

The hypothesis certification model is very suitable for use for certification of both newly developed and reusable software components. A major advantage with the model is that it does not require a certain number of failures to occur before obtaining results from the model. It works even if no failure occurs at all, since a certain failure-free execution time makes it possible to state that the MTBF, with a given degree of confidence, is greater than a predetermined value. Therefore, the MTBF is a realistic measure, even if a particular software component may be fault free. The model does not assume any particular failure distribution. Most available software reliability growth models require the occurrence of many failures before predictions can be made about component reliability. A normal figure would be in the region of 20-40 failures. Hopefully, this is not a realistic failure expectation figure for a software component.

It is also anticipated that as the popularity of reuse develops, the quality of software systems will improve, since the reusable components will have been tested more thoroughly and certified to a specific reliability level for different usage profiles. Therefore implying, that when performing systems development with reuse, the number of faults in a component will be exceptionally low. The proposed hypothesis certification model can still be applied.

5. A simple example

5.1 General description

The objective in this section is to summarize and explain the models presented in the sections above in a thorough but simple manner using an example from the telecommunication domain. The example follows the steps outlined in section 2.1 and the basic concept of the usage model presented in figure 1.

A module M is to be certified. The module is composed of two components, C1 and C2. The module itself can be seen as a component. C1 is a reused component found in a repository and C2 is reused with extensive adjustments. C1 offers service S1, to the user, whereas C2 originally offered S2. In the reused component version, S2 is changed to S2', and a further service is added, S3. The module is shown in figure 4.

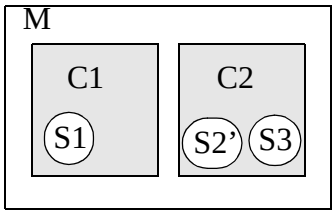

Figure 4. Module to be certified.

The reused component, C1, has been certified before but with a usage profile other than the profile expected when using C1 in module M. It must therefore be re-certified.

5.2 Modelling of software usage

The SHY usage model for module M is developed according to the structure in figure 1. Two different user types use the module, namely UT1 and UT2. There are two users of the first type, i.e. U11 and U12, and one user of the second type, U21. The three upper levels of the SHY model – usage level, user type level and user level – are illustrated in figure 5.

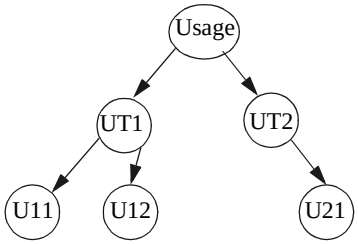

Figure 5. Upper levels of the usage model for the module.

Three services are available for the users of the module. The first service, S1 in C1, is reused without adjustment. Therefore, the behaviour level usage model for the service can also be reused without adjustment. The usage profile for S1 has, however, changed and a new profile must be derived. Service S2 in C2 has changed to S2', which results in the addition of two new states to the original usage model of S2. For service S3 in C2, a new behaviour level usage model must be developed, since the service is new. The behaviour level usage models for the services are illustrated in figure 6. The shaded areas are changed or new.

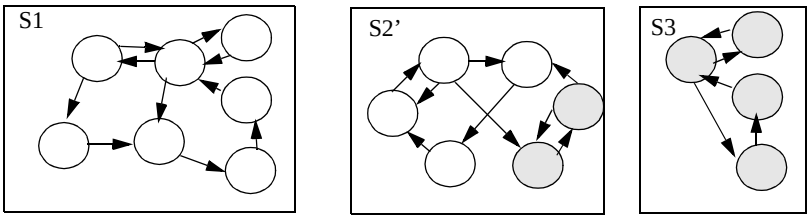

Figure 6. Usage models for S1 (reused), S2' (modified) and S3 (new).

Now the entire usage model can be composed using the structure in figure 5, together with the usage models for each service in figure 6. The users of type 1 have access to service S1. For each user of user type 1, an instance of the usage model of service S1 is connected. The user of type 2 has access to services S2' and S3, and hence one instance of each usage model is connected to the user of type 2. If the services are interdependent, links between services would be added, hence modelling the dependence between services used by the external users.

The entire usage model for the module is shown in figure 7.

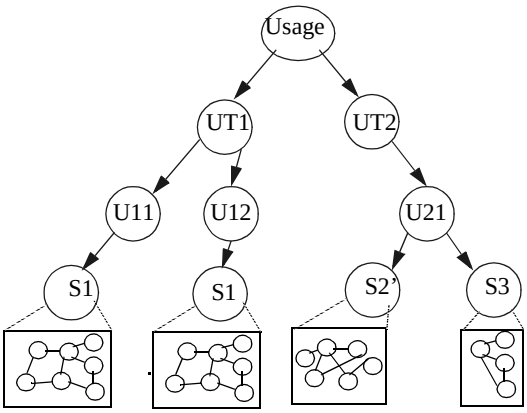

Figure 7. Usage model for the module.

5.3 Derivation of usage profile

The usage profile is derived, i.e. probabilities are assigned to the different transitions in the usage model. Static probabilities are used in this example, see section 3.3. First the probabilities for the different user types are determined, followed by the assignment of probabilities for the users of each type. The probabilities for selection of the services available for each user are determined. Finally, the transition probabilities within the behaviour level model for the services are assigned.

The usage profile can be derived bottom-up, if it is more suitable for the application in question. The strength of the modelling concept is that only one level need be dealt with at a time.

A possible usage profile for the usage model is shown in figure 8, though without probabilities for S2´ and S3.

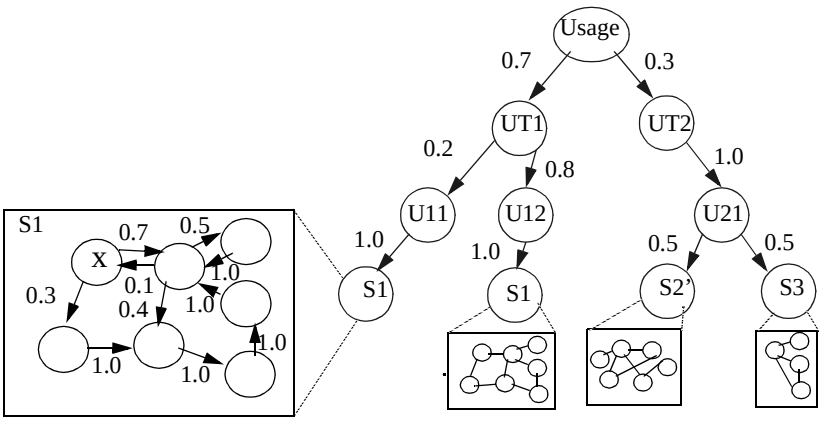

Figure 8. Usage profile for the module.

Unfortunately, it is impossible to go through the example in detail in this article, and in particular to discuss the links in more depth. For a more detailed presentation see [4, 7].

5.4 Generation of test cases

Initially, the usage model starts in a well-defined initial state, i.e. each of the behaviour level models are in their defined initial states. The selection starts in the usage state, where a random number is drawn, for example 0.534 selecting UT1, since 0.534 < 0.7, see figure 8. Another random number is drawn, for example 0.107 selecting U11, and for selection of its service no random number is needed. The actual state of the service usage model is denoted “X”. A transition is then selected and performed. The stimulus connected to that transition is

appended to the test script and the SHY model is run through again beginning from the usage state.

Several test cases or one long test case can be generated, depending on the application.

5.5 Execution of test cases and collection of failure data

The test cases generated are executed as in other types of tests and the failure times are recorded. Failure times can be measured in terms of execution time or calendar time, depending on the application. An example of failure data is shown in table 1. Times between failures are given in an undefined time unit. The failure data are constructed to illustrate the example.

Table 1: Failure data.

Failure number	1	2	3	4	5	6	7	8	9	10
Time between failures	320	241	847	732	138	475	851	923	1664	1160

5.6 Certification of reliability

The reliability is certified by applying the hypothesis certification model to the collected failure data. In fact, an MTBF requirement is certified. The MTBF objective in this example is 800 time units, which means that the first normalized failure time (t_{norm}) is equal to 320/800, see table 1 and figure 9. The data points are plotted in a control chart, with the module under certification being accepted when the points fall in the acceptance region. The module is accepted between the eighth and the ninth failure and the time for acceptance is about 5400 (6.8 * 800) time units.

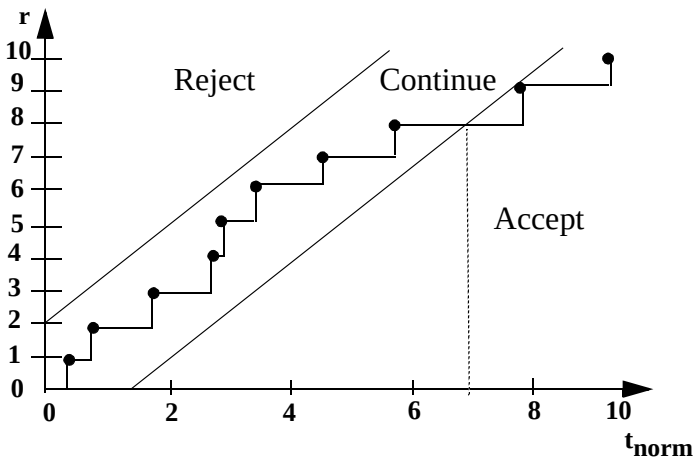

Figure 9. Control chart for certification.

6. Conclusions

Reuse will be one of the key developmental issues in software systems in the future, in particular reliable systems. Therefore, the reused components must be reliable. Component reliability can be achieved by applying sound development methods, implementing fault tolerance and finally, by adapting methods to ensure and certify the reliability and other quality attributes of the components both when developing for and with reuse.

Usage testing or operational profile testing has already shown its superiority over traditional testing techniques from a reliability perspective. A higher perceived reliability during operation can be obtained by usage testing, than with coverage testing, with less effort and cost. The gains in applying usage testing have been presented by, for example, Musa [2].

This article has concentrated on the certification of software components. The other issues (development of reliable software and fault tolerance) are equally important, although not discussed here.

The proposed method of usage modelling, i.e. the State Hierarchy (SHY) model, is shown to be a valuable abstraction of the fundamental problem concerning components and their reuse. The model in itself is divided into levels and the services are modelled as independently as possible, therefore supporting the reuse objective. The area of certification of software components is quite new. Some ideas and capabilities have been presented, but more research and, of course, application are needed.

The reliability certification model discussed is well established in other disciplines and it can also be adapted and used in the software community. The model is simple to understand and apply.

It has been emphasized that the usage models developed and the reliability measures with a given usage profile can be reused together with the components. The division into a structural usage model and different usage profiles makes it possible to reuse the usage model, and to apply new profiles, as a new environment has a behaviour different to that considered earlier.

The models and methods have been applied to a simple example to illustrate the opportunities and the benefits of the proposed scheme. It can therefore be concluded that the proposed method, or one similar to it, should be applied in a reuse environment, to obtain the necessary reliability in the components both when developing for and with reuse.

7. Acknowledgement

We wish to thank Johan Brantestam, Q-Labs for valuable technical comments, Helen Sheppard, Word by Word and Graeme Richardson for helping us with the English, as well as the whole personnel at Q-Labs. We also acknowledge suggestions made by anonymous *IEEE Transaction on Software Engineering* referees.

8. References

[1] Mills, Harlan D., Dyer, Michael and Linger, Richard C., “Cleanroom Software Engineering”, IEEE Software, September 1987, pp. 19–24.

[2] Musa, John D., “Operational Profiles in Software Reliability Engineering”, IEEE Software, March 1993, pp. 14-32, 1993.

[3] Tracz, W. (ed.), “Software Reuse: Emerging Technology”, IEEE Computer Society Press, Washington DC, USA.

[4] Wohlin, Claes and Runeson, Per, “Certification of Software Components” (long version), REBOOT – ESPRIT project 7808, Technical Report, REBOOT – 8213.1, 1993.

[5] Poore, J. H., Mills, Harlan D., and Mutchler, David, “Planning and Certifying Software System Reliability”, IEEE Software, January 1993, pp. 88-99, 1993.

[6] Whittaker, James A., and Poore, J. H., “Markov Analysis of Software Specifications”, ACM Transactions on Software Engineering Methodology, Vol. 2, No. 1, January 1993, pp. 93–106, 1993.

[7] Runeson, Per, and Wohlin, Claes, “Usage Modelling: The Basis for Statistical Quality Control”, Proceedings 10th Annual Software Reliability Symposium, Denver, Colorado, pp. 77–84, 1992.

[8] Musa, John D., Iannino, Anthony, and Okumoto, Kazuhira, “Software Reliability: Measurement, Prediction, Application”, McGraw-Hill, New York, 1987, pp. 201–203.