P. Runeson and C. Wohlin, "Usage Modelling: The Basis for Statistical Quality Control", Proceedings 10th Annual Software Reliability Symposium, pp. 77-84, Denver, Colorado, USA, 1992.

Usage Modelling: The Basis for Statistical Quality Control

Per Runeson and Claes Wohlin,
E-P Telecom Q-Labs, IDEON Research Park, S-223 70 Lund, Sweden,
Phone: +46-46-182980, Fax: +46-46-152880, e-mail: pr@q-labs.se.

Abstract

The testing method proposed in the Cleanroom methodology is Statistical Usage Testing. This testing technique is based on that the test cases represent the operational profile. The failure data from testing is then used in a software reliability model to certify the reliability of the software. This paper introduces a new model for modelling the usage of the system, i.e. the operational profile. It is shown to cope with the state explosion problem encountered when applying a simple Markov model to large software systems. The proposed model introduces a hierarchical Markov model, which is shown to solve the problems encountered in the ordinary Markov description. Users and services are represented so that the model can easily be extended when new users or services are added. It is concluded that the introduction of this model will make it possible to certify large software systems in the future. The model generates test cases providing failure data that can be used in software reliability models. The new model forms the basis for being able to apply statistical quality control to software systems.

Keywords

Cleanroom, Statistical usage testing, Statistical quality control, Markov model, Operational profile, Usage modelling, Software reliability

1. Introduction

The reliability problem in software systems of today is a well-known fact. No silver bullet will solve this problem, instead the solution will be the combination of several approaches. That is improvements throughout the whole life cycle. These improvements include for example specification and design, verification and validation, certification as well as maintenance. This is the approach taken in the Cleanroom methodology, [Mills87, Mills88, Dyer92], which includes methods for specification and design, verification and validation, as well as certification. In particular, Cleanroom supports the idea and philosophy that it is possible to develop zero-defect software.

The certification process is an important issue, since it is often one of the interfaces between the developer and the purchaser, in many cases the manager of the software. This process is the foundation for acceptance of a software product and a key issue in the quality control of software products. The objective is to certify during testing that the reliability requirements during operation are fulfilled. The basis for this is that the testing procedure resembles or models the operational profile. In Cleanroom this type of testing is referred to as Statistical Usage Testing, [Currit86, Cobb90].

The problem of certification involves two parts:

- modelling operation during testing, i.e. a statistical usage model.
- prediction of the reliability, i.e. software reliability models.

Numerous software reliability models can be found in the literature, some examples are presented in [Goel85, Jelinski72, Goel79]. The model proposed within the Cleanroom concept is presented in [Currit86]. Most of these models are based on the assumption of operational usage, but not much emphasis has been put into actually modelling and performing tests that fulfils this assumption.

It is often suggested that it is possible to describe the operation with a Markov model, see [Whittaker92]. But is this possible and can test cases be generated for all types of systems based on a Markov model?

In this paper the Markov model will be studied and we will indicate that it might not be useful for all types of systems. In particular, it will be shown that the ordinary Markov model is not possible to use for telecommunication systems. It is also believed that the conclusions from the telecommunication applications can be generalized to incorporate many types of distributed multi-user systems providing a wide range of services to the users. Based on the conclusion that the existing models are insufficient, a hierarchical Markov model is developed. It is shown how this model will cope with the problems encountered in for example telecommunication systems.

2. Certifying reliability

The certification of software within the Cleanroom methodology is discussed in [Currit86]. As stated above, the certification process consists of two equally important parts. The second part is the software reliability models. These shall model the behaviour of software failures and in particular predict the future behaviour and reliability of the

software. The models are based on several assumptions, where one of the most critical is the assumption that failures occur according to the operational usage. This means that the models can only be applied during operation or during testing where it is possible to generate test cases from an operational profile. The latter is the basis for Statistical Usage Testing.

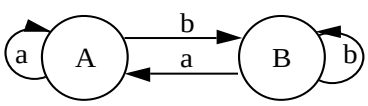
The first part, i.e. modelling the usage, has been much less studied than the software reliability models. The most frequently used approach is to model the behaviour and generate the test cases based on a Markov chain. This approach is not useful for all types of applications as will be shown below. The problem of modelling the usage for complex multi-user systems has been overlooked in the past, but if it shall be possible to certify the reliability under testing conditions this problem has to be solved. This has been one of the key issues in a project conducted for the Swedish Telecom. The objective with the project is to develop a method for certification of software products that can be used when purchasing software systems.

The usefulness of the development of software reliability models is debatable if the users´ needs can not be met, e.g. the assumptions made by the models have to be realistic or methods have to be developed that make the models realistic. The first approach has been found difficult, even if some attempts have been made for prediction of software reliability during different types of testing [Ehrlich87, Wohlin86, Ehrlich90]. The second approach is the one recommended in Cleanroom, i.e. to apply Statistical Usage Testing. This calls, however, for methods modelling the usage and the selection of suitable test cases. It has been shown that this type of testing is superior to other types of testing in finding the faults that influence the reliability during operation, [Adams84].

A model suitable for describing the usage of for example telecommunication systems will be described briefly below. A more comprehensive description can be found in [Runeson91]. This model will be one important part of a method for statistical quality control of software systems developed for the Swedish Telecom.

3. Usage modelling

The statistical usage profile is an external description of usage events and their probabilistic relationship. The usage is modelled as states in a state machine and transitions between states, drawn as arcs. An arc from state A to state B signifies an event associated with state A. The probabilities for transitions between the states are listed in a quadratic matrix.



from/to	A	B
A	0.2	0.8
B	0.9	0.

Figure 1. Statistical usage profile.

A simple statistical usage profile example is shown in figure 1. The start state is determined by the application.

From the statistical usage profile the test cases are generated. Starting in an initial user state a transition is chosen by e.g. the Monte Carlo method. The stimulus needed for this transition is recorded. From the new state a new transition is chosen etc. A test case can be made up of multiple state transitions. It can be of random length, or be finished by ending in a final user state, which is also determined by the application.

The test cases are randomly generated with respect to their probability of use and are then a representative subset of the use cases in operation. They are used to represent the operation and, like national polls, are the basis for the prediction of future results.

The basis for modelling the statistical usage profile as a Markov chain is that the usage exhibits the Markov property. When the usage model is in any state, the next state can be predicted without regard to previous states. The present state and the stimulus are enough to determine the next state. This is normally a reasonable description of the usage of most software systems.

3.1. Usage profile development

To develop the statistical usage profile three issues need to be handled:

- User states
- Stimuli
- Transitions

3.1.1. User states

The first task is to identify all the user states. A user state is the user´s view of the system at a special point of time. The user states must be independent from the implementation.

3.1.2. Stimuli

The stimuli are what the user can send to the software system. Each stimulus or set of stimuli will make a transition in the Markov chain, from one user state to another, or to the same state again.

In order to develop a usage profile we have to assign probabilities to the transitions between user states, according to the software usage in ordinary operation, i.e. to understand the probability of what the next user stimulus will be, given any user state.

3.1.3. Transitions

The probabilities for transitions between the states are represented in a quadratic matrix. The row index represent the state to leave and the column index the state to enter, see figure 1.

4. State explosion

When applying the previous description model on

telecommunication systems, we have experienced that the number of states grow very large, too large to be characterised feasibly, which is shown in [Runeson91]. It would probably happen for any complex multi-user system.

A general formula for calculating the total number of states can be found from using the combinatorial theory. The problem is equivalent to take *n* elements out of a set of *S* with respect to the order and with replacement.

$$\#\text{ states} = S^{n} = \#\text{ states per user}^{\#\text{ users}} \quad (1)$$

In a private branch exchange there is more than 10 states per subscriber and up to 1000 subscribers. Anyone can understand that there is no possibility to go further this way. Every state cannot be defined because they are too many.

Similar problems arise in the domains of protocol validation [West87] and specification analysis [Ek91].

4.1. Equivalence classes

It is proposed in [Musa87] to reduce the number of states by representing them by equivalence classes. An example considering the sum from throwing two dice, the states (1,6), (2,5), (3,4), (4,3), (5,2) and (6,1) can all be viewed as state 7. In telecommunication systems it is equivalent to skipping the identity of the subscribers.

We have tried using equivalence classes. States with the same number of subscribers in each user state are considered equal. Dependencies have for once been helpful in our ambition for decreasing the number of user states. These steps decreased the number of states rather a lot, but not enough to be satisfactory.

We have experienced that even in very simple examples, a state explosion occurs. The example systems studied did not contain any of the added services expected from a modern exchange, neither the operator´s role nor the net traffic were considered. These factors would make the systems even more complex.

4.2. Concluding remarks

Software usage is to be described with a description model. We have tried the plain Markov model in our application domain and concluded that the models are growing too large. There is a need for a new description model to master the state explosion problem.

5. State hierarchy

Our proposal is to describe the user states as a levelled State Hierarchy, abbreviated SHY. The general model can consist of several hierarchy levels. Here, however, it is presented in its simplest form with only two levels. Later, in section 5.4 an extended version is shown.

Examples are taken from the telecommunication domain, but the model is probably applicable in other domains as well.

5.1. Two-level state hierarchy

The upper level in the two-level SHY is a view of the usage model, called User Level (UL). It describes which user is generating the next stimulus. The lower level, which contains information about the user behaviour, is called Behaviour Level (BL). It describes which stimulus the selected user will generate. Both levels are described as Markov chains. When generating test cases, first a user is randomly chosen on the User Level, then a transition for the selected user on the Behaviour Level is chosen.

5.1.1. User Level

The User Level system view is a state machine, telling which user in the system is to give next stimulus. Figure 2 shows an UL description for a system with *N* users.

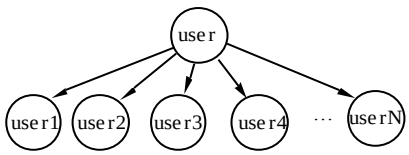


Figure 2. User Level with N users.

When a test case is generated, the user to generate next stimulus is selected by choosing a transition randomly on the UL. In every selection the main state, *user* in the figure, is the initial state.

5.1.2. Behaviour Level

The Behaviour Level in the SHY consists of state machines, showing the system from one user´s point of view. The behaviour of each user is described by a Markov chain. Figure 1 can be seen as a very simple example.

The only transition probabilities considered are those, caused by stimuli from the user itself. Transitions caused by stimuli from other users are marked with an asterisk in the transition matrix. These stimuli are generated in other Behaviour Level submachines, and shall not be represented twice.

The asterisk is a mark for the connection between the two BL machines, here called a *link*. The link means that a stimulus from one user causes a transition for another user.

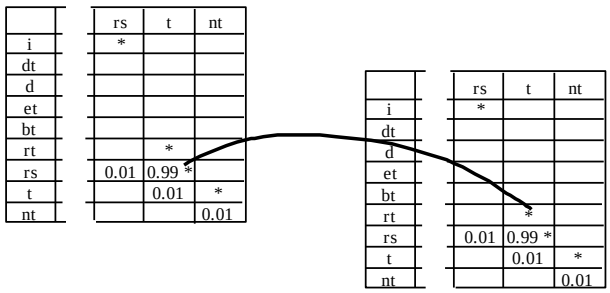


Figure 3. Connection link between two BL transition matrixes.

In figure 3 parts of two BL transition matrixes are shown. The user, corresponding to the leftmost matrix, makes a transition from state *rs (ringsignal)* to *t (talking)*, i.e. makes off hook when dialled. This transition causes another transition in the BL state machine, corresponding to the rightmost matrix, from state *rt (ringtone)* to *t (talking)*, i.e. is connected to the dialled.

The information on the linked state machine and the transition to be done has to be stored, connected to the transition matrix of the causing state machine. When a test case is generated and a link transition is chosen, both the causing and the caused transitions have to be added to the test script.

5.1.3. Usage model

The User Level and the Behaviour Level make together the SHY usage model in its simplest form. Each user has a corresponding state on the User Level as well as a state machine on the Behaviour Level.

The state of the whole usage model is described in terms of the BL machine states. A usage model state in the State Hierarchy model is denoted as an array of BL states, [$State_{user1}$, $State_{user2}$, ..., $State_{userN}$].

The total number of possible states is not less than in the plain Markov model, but here they are hidden in the hierarchical structure and must not be handled explicitly. The plain Markov description must be expanded to show every state. In the hierarchical model the description is divided into smaller parts, each of them easier to handle.

5.1.4. User Level probabilities

On the UL, the transition probabilities are depending on the BL machine states. The probabilities are functions of the BL machine states. Consider the following example from the telecommunication domain: When one subscriber is in a state within a sequence[1] , e.g. *dialling*, it is more probable that this subscriber gives the next stimulus than the other, being in the state *idle*. But when all subscribers are in the same state, e.g. *idle*, it is as probable for any of the subscribers to give the next stimulus, if the subscribers are equal.

To express this difference, every state in the BL machine is given a relative weight, denoted $w_{BL\text{-}state}$. This corresponds to the probability for selecting the user being in the very BL state.

The probability for choosing a user on the UL can now be expressed in terms of the state weights. The probability for choosing user *i* is denoted p_i:

$$p_i = \frac{w_{BL\text{-}state(i)}}{\sum_{k=1}^{N} w_{BL\text{-}state(k)}} \quad (2)$$

where *N* is the number of users.

5.2. Reality

In the plain Markov model, user behaviour is modelled as a Markov chain and this is supposed to be a reasonable model. Now our task is to show that the hierarchical model captures the same behaviour as the plain Markov model. This will be made credible by analysing three issues:

- State space
- Reachable states
- Transitions

5.2.1. State space

In a plain Markov model there are S^N states, according to (1), where *S* is the number of states from one user´s point of view and *N* is the number of users.

In the SHY model there are *N* Behaviour Level machines with *S* states each. These *S*N* states can however be combined in many ways. When one of the *N* users on the User Level is selected, each of the remaining *N–1* users can be in one of *S* states. They can be combined in S^{N-1} ways. The selected one can be in one of *S* states. The total number of combinations is then $S^{N-1} * S = S^N$, the same number as in the plain model.

5.2.2. Reachable states

In the theoretical Markov model, there is no limitation between which states transitions are possible. This possibility is however seldom used. In transition matrices representing real systems, most elements are equal to zero.

In the SHY model, only a subset of the states is reachable from each state in a fully connected BL machine. A transition is chosen in two steps, first one of *N* users is chosen, then one of *S* states is selected. This provides that there is only one transition between two BL states.

Some of the transitions however lead to the same system state. The number of unique states is hence:

$$\#\ states = N*S–N+1 = N*(S–1)+1. \quad (3)$$

where *N*S* is the number of transitions, non-unique calculated multiple. *N* of them result in the same state, they are subtracted and one state representing the *N* equal states is added.

The number of states in a fully connected Markov model is S^N. When applying the theoretical model on our practical problem, all states are not reachable. The stimuli are generated one at a time, then only the transitions caused by one stimulus are possible. Other transitions are made as

[1] States in a sequence means here states, often passed in almost non-interrupted succession, e.g. *dialtone — dialling — ringsignal — talking*.

sequences of transitions, caused by sequences of stimuli. This explains why most of the elements in the transition matrix are equal to zero.

In the plain model, only one of *N* users can give a stimulus in practice. This stimulus can cause a transition to one of *S* user states. That makes *N***S* reachable states. *N* of these result in the same final state, and hence the number of reachable states from one state is the same as in (3).

The general coupling between the plain Markov and the SHY model, is a mapping of the transition matrixes for each user in the SHY model onto the plain transition matrix. It is a transformation from *N* dimensions to two. The mapping function depends on our ordering of the states, and then it is difficult to find a general function.

The State Hierarchy model has not as many degrees of freedom as the fully connected Markov model. But the SHY model has enough degrees of freedom to represent telecommunication systems, which are not fully connected in the plain Markov model. Other systems are not investigated but it seems to be possible to use the SHY model in other domains as well.

5.2.3. Transitions

Consider an example with three users with three user states each, *A*, *B* and *C*, see figure 4a and 4b for the SHY and PM descriptions respectively.

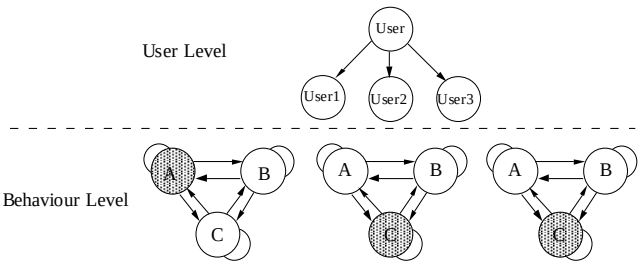


Figure 4a. SHY description.

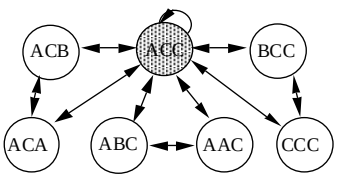


Figure 4b. Part of the corresponding plain Markov description.

Now consider the usage model state denoted [*A,C,C*] in the SHY model (figure 4a) and the corresponding state *ACC* in the plain Markov model (figure 4b). Next state can be one of the following seven: *ACC, BCC, CCC, AAC, ABC, ACA* and *ACB*. The probabilities for the transitions are denoted $p_{ACC–ACC}$, $p_{ACC–BCC}$ and so forth.

For the hierarchical model, some notations are needed: w_A, w_B and w_C are the weights for the three BL-states. p_{AA}, p_{AB}, p_{AC}, p_{BA} etc. are the transition probabilities between the Behaviour Level states. p_1, p_2 and p_3 are the probabilities for each user on the User Level to be the next to generate a stimulus.

The probability for a selected transition in the plain Markov model, can be expressed in terms of transition probabilities in the SHY model. It is a sequence of two choices. First a user is chosen, then a transition for this user. The probability for this transition is then the probabilities for the User Level transition, multiplied with the probability for the Behaviour Level transition. This is expressed as:

$$p_{PM} = p_{SHY\ UL} * p_{SHY\ BL} \quad (4)$$

where *PM* stands for Plain Markov, *SHY* for State Hierarchy, *UL* and *BL* for User Level and Behaviour Level.

In our example, the probabilities for the transitions from the state *ACC* are according to (4):

$$p_{ACC–ACC} = p_1 * p_{AA} + p_2 * p_{CC} + p_3 * p_{CC} \quad (5a)$$
$$p_{ACC–BCC} = p_1 * p_{AB} \quad (5b)$$
$$p_{ACC–CCC} = p_1 * p_{AC} \quad (5c)$$
$$p_{ACC–AAC} = p_2 * p_{CA} \quad (5d)$$
$$p_{ACC–ABC} = p_2 * p_{CB} \quad (5e)$$
$$p_{ACC–ACA} = p_3 * p_{CA} \quad (5f)$$
$$p_{ACC–ACB} = p_3 * p_{CB} \quad (5g)$$

where $p_1 = \frac{w_A}{(w_A+w_C+w_C)}$ and

$p_2 = p_3 = \frac{w_C}{(w_A+w_C+w_C)}$ according to (2)

It generally holds that the probabilities for leaving a selected state, always sum up to one. In the non-hierarchical model the sum of the probabilities for leaving *ACC* is (see figure 4b):

$$\sum p = p_{ACC–ACC}+p_{ACC–BCC}+p_{ACC–CCC}+p_{ACC–AAC}+ p_{ACC–ABC}+p_{ACC–ACA}+p_{ACC–ACB} \quad (6)$$

This can be expanded in terms of SHY model probabilities, according to (5a)–(5g):

$$\sum p = p_1*(p_{AA}+p_{AB}+p_{AC})+p_2*(p_{CA}+p_{CB}+p_{CC})+ p_3*(p_{CA}+p_{CB}+p_{CC}) \quad (7)$$

The probabilities for leaving a state in the Behaviour Level description in figure 4a sum up to one:

$$p_{AA} + p_{AB} + p_{AC} = 1 \quad (8a)$$
$$p_{BA} + p_{BB} + p_{BC} = 1 \quad (8b)$$
$$p_{CA} + p_{CB} + p_{CC} = 1 \quad (8c)$$

In the User Level description in figure 4a, the sum of the probabilities are:

$$p_1 + p_2 + p_3 = 1 \quad (9)$$

The equations (8a) – (8c) put into (7), and then with (9) our result is:

$$\sum p = p_1 + p_2 + p_3 = 1 \quad (10)$$

This indicates that transforming transition probabilities from the plain to the State Hierarchy model does not change the basic properties for the transition probabilities. To prove that the SHY captures the same behaviour as the plain Markov model, some Markov chain characteristics e.g. state occurrence probabilities, first passage probabilities and mean recurrence time could be investigated. The objective in the current project has however been to find a practical solution and not to prove the mathematical equivalence between the two different Markov representations.

5.3. Usability

Another important question is the usability of the model. Is it easy to model the transition probabilities? How to generate test cases? These questions are discussed in this section.

5.3.1. Modelling transition probabilities

The Behaviour Level transition probabilities are modelled to correspond to the user behaviour. From each user state, the possible transitions are considered and each of them, caused by stimuli from the user itself, is given a probability. The transitions caused by other users are marked with an asterisk, to indicate the connection to another user.

The state weights are chosen to reflect the probability that the next event originate from the BL-state, relative to other BL-states. Different values of the state weights cause different types of usage. If there is little difference between the probabilities for the different states, the state weights become equal. Then many users act in parallel, i.e. much function interaction is generated.

Other types of usage is generated when the probabilities are very different in different BL states. States, which often are passed in a fast sequence have higher state weights. Hence the probability for passing this sequence, without interrupt, is higher. The probability for many users acting in parallel is hence lower.

In table 1, two cases with different state weights and its consequences on the user probabilities are shown:

---

state = [A,C,C] gives

$$p_1 = \frac{w_A}{(w_A + w_C + w_C)}\ ;\ p_2 = p_3 = \frac{w_C}{(w_A + w_C + w_C)}\ ;$$

	Case 1	Case 2
	$w_A=1.1$, $w_B=1.2$, $w_C=1.3$	$w_A=1$, $w_B=2$, $w_C=3$
p_1	$1.1/(1.1+2*1.3) = 0.30$	$1/(1+2*3) = 0.14$
p_2	$1.3/(1.1+2*1.3) = 0.35$	$3/(1+2*3) = 0.43$
p_3	$1.3/(1.1+2*1.3) = 0.35$	$3/(1+2*3) = 0.43$

Table 1. UL probabilities with different state weights.

From a practical viewpoint can be concluded that this modelling of probabilities is easier than in the plain Markov model for multi-user systems. It is easier to associate a probability for a user performing a specific action than studying the probabilities for different events in a global state as is the case for the plain Markov model.

5.3.2. Test cases

To generate test cases from the SHY statistical usage profile, following four-step algorithm can be used:

1, Determine which user to generate next stimulus. This is done by choosing a user in the User Level state machine.
2, Determine which stimulus the selected user will generate. This is done by choosing a transition in the Behaviour Level state machine for the selected user.
3, Note the consequences of this stimulus for the selected user and for the others by following the links and append the generated stimuli to the test script.
4, Update the User Level transition matrix.

Random numbers for controlling the choices are taken from a uniform distribution, i.e. as a sample of a random variable uniformly distributed on (0,1).

5.4. Extensions

The reasoning in section 5.2 indicates that the hierarchical model can express the same as a plain Markov chain. But there is a possibility to express more with the State Hierarchy model. An extended SHY model is shown in figure 5. The levels in the figure can be described as follows:

- The state weights are representing the time scale in the usage profile. In the plain Markov model this aspect is implicit in the transition probabilities, but here it is more visible. It is no exact time scale, but a relative.

  By adding a stimulus intensity on a Time Level as an upper level, we can model an absolute time scale.

This gives a possibility to test time constraints and real time aspects in the system. It is also a way to model and control the system load.

- The Usage Level contains one state, which is the main state for selecting the underlying user types.
- The SHY model can be extended, by adding a User Type Level (UTL), above the User Level. On this level, a choice between different types of users can be done. This makes it easier to handle large systems.
- On the User Level (UL) the individuals of the user types are shown. They are instances of the user types.
- To support modularity and reuse of the usage model, a Service Level is introduced. This implies that the usage of each user is described as a set of different services, each of them describing a part of the usage. When adding new functionality it is easy to add new services to a user.
- This leads to that the Behaviour Level describes the behaviour of the services instead. Each service is described by a BL state machine, similar to the prior described (see figure 4a).
- A stimulus can be refined by using a Sub-Behaviour Level (SBL) state machine. E.g the stimulus ”digit” can be chosen on the BL and then an SBL choice selects the exact digit, 0 to 9.

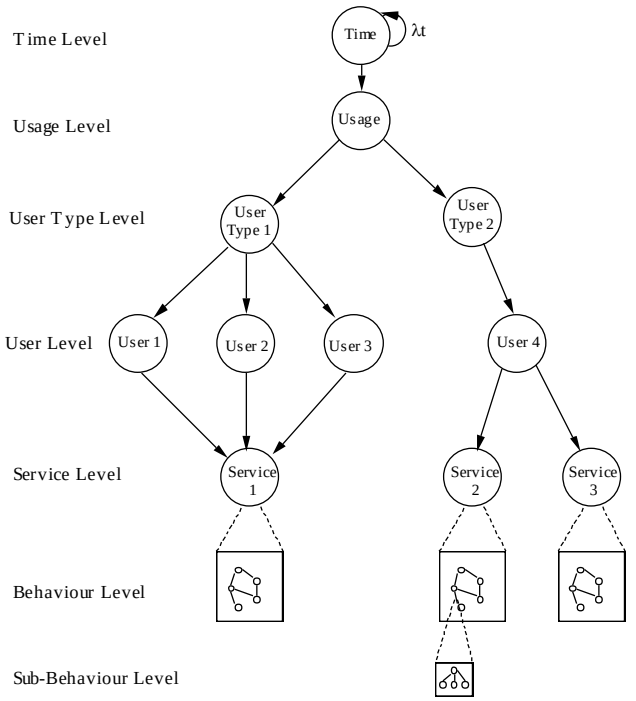


Figure 5. Extended SHY model.

With this extended SHY model it is rather easy to add new services or users. The model also supports easy reconfiguration of the system. Most parts of the system description can be reused. If the model once is constructed, changes are rather easy to cope with. Only the sub-parts of the system, directly describing the changes must be exchanged.

Test cases are generated by traversing the SHY model. First a time to next event is drawn on the Time Level. Then the main state, Usage, is entered from which a selection of a User Type is done. If e.g. User Type 2 is selected, there is only one user and this will hence be drawn on the User Level. One of the services connected to User 4 is drawn, e.g. Service 2, and then a transition in its Behaviour Level state machine. The selected stimulus and its possible influence on other BL-state machines are added to the test script, or if there is a Sub-Behaviour Level connected to the stimulus, a refinement of the stimulus is drawn and it is added to the test script. Then the probabilities are updated and the model can be traversed again.

6. Experience

We have performed a case study in usage modelling for an advanced telephone device. The case study has given experience about specifying a statistical usage profile. The greatest positive experience is that it is possible to describe a SHY usage profile for a real product of the tele-communication domain. Even though the studied product is rather small, we have seen needs for alternatives to a plain Markov model.

We have seen as well that the SHY model gives a description, which is of reasonable size. We have concluded that there is space for specifying a usage profile for a much larger product, using the SHY model.

Problems encountered concern service interaction, i.e. which services may act together and which services affect each other. We have not found any simple trick to solve the service interaction problems. They are however common in the telecommunication domain and is a large current research area.

7. Conclusions

It can be concluded that the statistical quality control of software products is an important issue. The certification process is central in this effort. This process is highly dependent on relevant software reliability models and a sound basis for prediction. The basis includes relevant failure data, i.e. data that is obtained under circumstances fulfilling the assumptions of the reliability models. In particular, this means that the failure data during testing has to be similar to the failure data encountered during operation. This type of testing is often referred to as Statistical Usage Testing. To be able to use this type of testing, a model for the usage is needed. No suitable model was found in the literature, i.e. the existing ones were not suitable for describing the application studied (primarily telecommunication systems).

A model for describing the usage of software systems has been proposed. The model is based on describing the usage as a hierarchical Markov model. This modelling approach provides some essential benefits: for example, it is based on well-known theories, it is simple to understand, easy to add new parts as well as remove old parts. The model can easily be used to generate test cases that are representative of the operational usage. This means that the model provides the necessary basis for performing Statistical Usage Testing. It can be concluded that the proposed model fills a gap in the process of certifying software.

Acknowledgement

The project is being conducted for the Swedish Telecom, to whom we are grateful for specifying this project and in particular for letting us publish the results.

Thanks to Professor Lars Reneby, Department of Communication Systems, Lund Institute of Technology for valuable comments in the work with the master thesis of Mr Runeson.

Many thanks to Erik Johansson and Bo Lennselius, E-P Telecom Q-Labs for interesting and fruitful discussions and comments throughout the project.

We also would like to acknowledge Dr James A. Whittaker, University of Tennessee, Knoxville and Dr Even-Andre Karlsson, E-P Telecom Q-Labs for valuable improvement suggestions concerning the paper.

References

[Adams84] E. N. Adams, ”Optimizing Preventive Service of Software Products”, IBM Journal of Research and Development, January 1984.

[Cobb90] Richard H. Cobb and Harlan D. Mills, ”Engineering Software Under Statistical Quality Control”, IEEE Software, November 1990, pp. 44-54.

[Currit86] P. Allen Currit, Michael Dyer and Harlan D. Mills, ”Certifying the Reliability of Software”, IEEE Transactions on Software Engineering, vol SE-12, no 1, January 1986, pp. 3-11.

[Dyer92] Michael Dyer, ”The Cleanroom Approach to Quality Software Development”, John Wiley & Sons, 1992.

[Ehrlich87] Willa K. Ehrlich and Thomas J. Emerson, ”Modeling Software Failures and Reliability Growth during System Testing”, In Proceedings 9th Int. Conf. on Software Engineering, 1987, pp. 72-82.

[Ehrlich90] Willa K. Ehrlich, S. Keith Lee and Rex H.Molisani, ”Applying Reliability Measurement: A Case Study”, IEEE Software, March 1990, pp. 56-64.

[Ek91] Anders Ek and Jan Ellsberger, ”A Dynamic Analysis Tool for SDL”, SDL ’91: Evolving Methods, Elsevier Science Publisher B V (North Holland) 1991.

[Goel79] Amrit L. Goel and Kazuhira Okumoto, ”Time-dependent Error-detection Rate Model for Software Reliability and Other Performance Measures”, IEEE Transactions on Reliability, Vol. R-28, No. 3, 1979, pp. 206-211.

[Goel85] Amrit L. Goel, ”Software Reliability Models: The State of the Art”, IEEE Transactions on Software Engineering, Vol. SE-11, No. 12, 1985, pp. 1411-1423.

[Jelinski72] Z. Jelinski and P. Moranda, ”Software Reliability Research”, Statistical Computer Performance Evaluation, 1972, pp. 465-484.

[Mills87] Harlan D. Mills, Michael Dyer and Richard C. Linger, ”Cleanroom Software Engineering”, IEEE Software, September 1987, pp. 19-24.

[Mills88] Harlan D. Mills and J. H. Poore, ”Bringing Software Under Statistical Quality Control”, Quality Progress, November 1988, pp. 52-55.

[Musa87] John D. Musa, Anthony Iannino and Kazuhira Okumoto, ”Software Reliability, Measurement, Prediction, Application”, McGraw-Hill Int. 1987.

[Runeson91] Per Runeson, ”Statistical Usage Testing for Telecommunication Systems”, Dept. of Communication Systems, Lund, Sweden, Report no. CODEN: LUTEDX (TETS-5134)/1-49)/(1991)&Local 9, 1991, Master thesis.

[West87] Colin H. West. ”Protocol Validation by Random State Exploration”, Protocol Specification, Testing and Verification VI, Elsevier Science Publisher B V (North Holland) 1987.

[Whittaker92] James A. Whittaker, ”Markov Chain Techniques for Software Testing and Reliability Analysis”, Dept. of Computer Science, University of Tennessee, Knoxville, USA, 1992, Ph.D. Dissertation.

[Wohlin86] Claes Wohlin, ”Software Testing and Reliability for Telecommunication Systems”, In Software Engineering ´86, Peter Peregrinus Ltd., United Kingdom, 1986, pp. 27-42.