15

Early Estimation of Software Reliability through Dynamic Analysis*

Anders Wesslén and Claes Wohlin
Dept. of Communication Systems, Lund Inst. of Tech., Lund University, Box 118, S-221 00 Lund, Sweden, Phone: +46-46-2223319, Fax: +46-46-145823, E-mail: (wesslen, claesw)@tts.lth.se

Abstract

Early estimations and predictions of software quality attributes are essential to be in control of software development and to allow for delivery of software products which fulfil the requirements put on them. This paper focuses on a method enabling estimation and prediction of software reliability from the specification and design documents. The method is based on dynamic analysis of a well-defined high level description technique, and by applying usage-oriented analysis, it is illustrated, through a case study, how the reliability can be controlled. Furthermore, it is described how the output from the analysis can be used as an acceptance criterion of the design, as support in the planning process for the test phases to come and finally as a method to enable estimation and prediction of the reliability in the testing phase and operational phase. The method is still being evaluated and improved, but it can be concluded that so far the results are inspiring for the future.

Keywords

Software reliability, statistical usage testing, dynamic analysis, usage modelling.

---

* This work is supported by National Board for Industrial and Technical Development (NUTEK), Sweden, Reference Dnr: 93-2850.

S. Bologna et al. (eds.), *Achieving Quality in Software*

1 INTRODUCTION

A key issue in achieving quality software is the ability to ensure the software quality attributes, for example reliability, throughout the software life cycle. In particular, this implies that quality attributes must be assessed early in the life cycle. Assessment during the early phases of software development is the only way to be able to influence the software process in the ongoing project, and hence also the final quality of the product. Therefore, this paper focuses on the ability to perform reliability estimation and prediction from software specifications and designs in a high-level description technique.

These early indications of software quality are essential for planning and controlling the further development as well as obtaining a quality check of the final software product. A method for a usage-oriented analysis approach, which enables software reliability estimation and prediction from specification and design documents, was originally proposed by Wohlin (1992). The method has since then been further elaborated and a number of problems have been solved. The method and its potential are here presented through a case study, where the actual implementation of the method is presented. The solutions to some technical problems that occur when implementing the method are highlighted to enable people to adopt the proposed method.

2 OBJECTIVES

A major problem to obtain quality control of software is the inability to obtain early and objective measures of quality. Thus, methods for early estimation and prediction of quality attributes are essential (Musa, 1990). The objective here is to provide such a method for early software reliability estimation and prediction.

Statistical usage testing (Mills, 1987) and (Runeson, 1995a) or operation profile testing (Musa, 1993) is an emerging technology. The objective with this test technique is to resemble the actual usage to allow for reliability certification. It is, however, not enough to improve the test phase, similar procedures are needed at earlier stages in the software life cycle.

The objective is to illustrate how a usage-oriented approach can be applied early, through:

- usage modelling;
- generation of usage cases;
- dynamic analysis from a usage perspective of a software specification or design;
- estimation of software reliability for dynamic failures identified by the available tool support;
- prediction of software reliability in general, which can be used to plan the forthcoming test phase and also determine when it is likely that the reliability requirement is fulfilled.

These issues are presented through a case study of a software design. It is shown how the data obtained from the dynamic analysis can be used both to control the subsequent development and testing phases and as an estimator and predictor of the final software reliability. Furthermore, it illustrates that achieving quality software must mean that high-level specification and design techniques are applied. These techniques do not only provide a better development environment, they provide actually also new opportunities to perform early analysis of different software quality attributes.

3 THE CASE STUDY

The case study used in this paper is a small telecommunication system. The system is called SPOTS and controls a small digital telephone exchange. This system is a part of an educational development system at the department of Communication Systems, see (Yeh, 1989). The basic functionality of SPOTS is to provide services for plain ordinary telephone calls. SPOTS is used in an undergraduate project course which is held to teach the students about system development for large and complex systems. In this project course the students modify and extend the basic SPOTS with the following new telephone services:

- Charging
  This service contains two parts. The first part is the actual charging of the calls and the second contains functions to read and to reset the charging.
- Take Call
  This service provides the user with the ability to take a telephone call from a different telephone than the ringing one.
- Call Forwarding
  This service moves the incoming telephone calls to another telephone.
- Maintenance functions
  These functions can only be ordered from the operator terminal. The maintenance functions are:
  - Installation of a new subscriber
  - Removal of a subscriber
  - Change the telephone number for a subscriber.

The design and implementation of the new services are made using SDL (ITU-T, 1988) and the development tool SDT* (SDL Development Tool). SDL (Specification and Description Language) is a standardized specification and design technique, and an introduction to the technique can be found in, for example, (Belina, 1991).

4 USAGE MODELLING

The system described briefly in the previous section is the basis for the case study. The services should be modelled from the user perspective, hence it is mostly necessary to model ordering and cancellation of services and not the actual behaviour as it is transparent for the user of the system. For example, it means that a subscriber phoning to another subscriber where the latter has forwarded his calls does not know that the call is forwarded, hence the actual behaviour is invisible and it should not be modelled in a usage model. This is the explanation of the mapping of the services in the previous section to the usage model in Figure 1.

The usage of the system is modelled with a hierarchical state model. This type of model is described by Runeson (1992, 1995a) and Wohlin (1994). The model is illustrated in Figure 1. The lowest level in the hierarchy is a description of the services. This level is described with a Markov chain, which is not shown in the figure.

---

* SDT is a registered trademark of Telelogic AB, Malmö, Sweden.

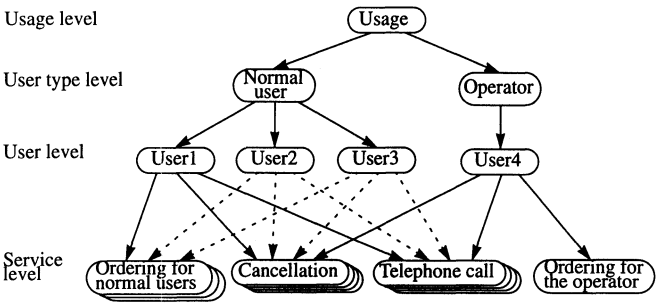

Figure 1 The hierarchical part of SPOTS's usage model.

The usage model is then complemented with a usage profile, which describes the usage frequency of different services and different events. As an example, probabilities are assigned to the possibility that the next event is generated by a normal user or by the operator. It is possible to make the probabilities dynamic, i.e. dependent on the actual state of different services in the system. It is, however, outside the scope of this presentation to elaborate on this issue any further. The details are presented in (Runeson, 1992 and 1995a) and (Wohlin, 1994).

The state hierarchy model is transformed into SDL, see (Runeson, 1995b). The model could have been transformed into other representations as long as they support state machines, but SDL was chosen due to the available tool support and that the system to be assessed was implemented in SDL. One advantage with transforming into a standardized technique is that it was not necessary to spend a lot of time developing a tool, which means that the focus could be on the quality issues.

5 USAGE GENERATION

The usage specification, i.e. usage model and usage profile, forms the basis for generating usage cases which are representative of the anticipated usage of the system.

The usage specification is now available in the tool environment and usage cases are generated by running through the hierarchy according to the assigned usage profile. The person generating the usage cases must act as an oracle at this stage and answer the usage specification with the responses expected of the system when it has been developed. The expected answers are obtained by using the requirements specification. This is illustrated in Figure 2.

The generation is made semi-automatically in the sense that the person generating the usage cases must act as an oracle, but the usage cases are logged automatically on a file. The log includes both the usage and the expected answers from the system, which are provided by the person generating the usage cases. Fault handling can also easily be incorporated, hence allowing for automatic execution of the generated usage cases and it is thus possible to log the failures that occur, i.e. deviations from the expected behaviour. The usage generation procedure is further described by Runeson (1995b) and Wesslén (1995).

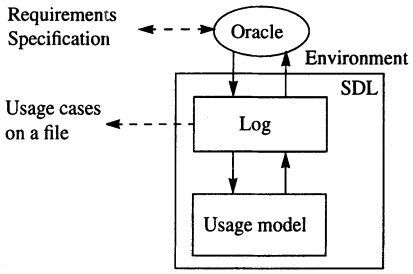

Figure 2 Generation of usage cases.

The usage cases are input to the dynamic analysis, which means that the analysis can be made from a usage perspective, hence supporting the objective of early estimation and prediction of software reliability.

6 DYNAMIC ANALYSIS FROM A USAGE PERSPECTIVE

6.1 Dynamic analysis

In the tool environment SDT there is a dynamic analyser which is called the Validator. The objective of the Validator is to support the specifier and the designer to avoid unwanted dynamic properties in the specified behaviour. The prototype of the Validator is described by Ek (1991). When the specified system interacts with the environment the analysis can be guided by using an SDL model of the interaction or a Message Sequence Chart, MSC (ITU-T, 1993), which describes the wanted exchange of signals between the system and the environment.

The Validator automatically detect some sorts of faults. The types of faults that the Validator can detect are, for example, MSC violation, deadlocks, more than one receiver of a signal, output faults. The analysis is made by using a tree expansion procedure and is halted when a fault is detected. When the analysis is halted a report is generated, which contains the type of fault, location of the fault and the number of passed states during the analysis. If the analysis of an MSC does not detect any faults an MSC Validated report is generated with the number of analysed states.

When using usage cases described as MSCs, the results can be used in reliability estimation and prediction, because the usage cases fulfil the assumptions of the reliability models and the Validator's results can be viewed as a measurement of the time between failures. If the Validator is guided by the SDL model of the usage specification the results can not be used in reliability models as the usage specification describes the whole usage and not a sample of it. Therefore, the usage cases are generated from the usage specification in SDL and stored as MSCs.

6.2 Implementation decisions

The Validator is guided by the usage cases described as Message Sequence Charts (MSCs). The result from the analysis of a usage case is a report containing the type of fault and the number of analysed states since the last report. The number of analysed states between failures are then a measure of the time between failures. The number of analysed states is then mapped into real time, see Section 6.3.

There are some different procedures that can be identified which have different effects of the results of the dynamic analysis. The following procedures have been identified and investigated:

- Same start point vs. continuation
  The same start point for all usage case leads to that a start period is included in every usage case. This leads to longer times between failures than during operation. If the starting point for the usage cases is redefined as the ending point of the previous case, the usage cases seem to be a very long usage case and the starting period is excluded. For the dynamic analyser which is used in this paper, the starting point is redefined by saving the system state when the previous usage case ends.
- Short vs. long usage cases
  If the usage case is long and erroneous, a long sequence is discharged or need to be regenerated. This problem is less if the usage cases are short. The result is not depending on how long the usage cases are because the cases follow on each other when the starting point is redefined.

Another problem which arises is what to do with the usage case when a fault is discovered. There are two possible solutions:

- Continue with the same usage case
  The result of this is that the usage case after the failure is depending on the usage case before and this contradicts the assumptions in statistical testing (or analysis). In statistical testing the times between failures are considered as independent, i.e. the usage cases are independent of each other. In these cases the reanalysis of the system with the same usage case serves as a regression analysis.
- Continue with a new usage case
  If the usage case which finds the failure is discharged and a new is used after the discovery the usage cases are independent and the result can be used in statistical testing. In this case there is no regression analysis made. It is obvious that in this case, it is better to have short usage cases, otherwise there is a risk that long sequences have to be thrown away.

When a fault is discovered there are two alternatives what to do with it. The alternatives depend on what kind of result the dynamic analysis aims at. The alternatives are:

- Correct the fault and proceed with the analysis
  If the faults are corrected as they are discovered the failure data can be used in reliability growth models, and in prediction of the reliability in the future.

- Leave the fault and proceed with the analysis
  If the faults are not corrected the failure data can only be used to estimate the current reliability, and the analysis must continue with a new usage case as the dynamic analyser can not continue after a discovered fault.

  The dynamic analysis in the case study has been performed as follows:
- Each new usage case is a continuation of the preceding.
- Short usage cases are analysed after each other.
- When a failure occurs, the fault is removed.
- The analysis continues with a new usage case after the fault is removed.

6.3 Results

The analysis from a usage perspective gives us a number of times between failures, where the time is measured in the number of executed process states in the SDL description until a failure occurs during analysis. The result from the analysis of the case study, SPOTS, is presented in Table 1.

Table 1 Failure data from dynamic analysis

Failure number	*1*	*2*	*3*	*4*
Number of states between failures	984	795	1802	>3394

The measure of number of states between failures are then mapped into real time with an expansion factor, see Wohlin (1992). For example, the expansion factor can be that 1000 states corresponds to 3 hours in operation. This is a hypothetical value for illustration purposes and a realistic value must be determined for each specific organisation and application. In Table 2, the times between failures are shown after mapping the failure data from dynamic failures into real time.

Table 2 Failure data in real time

Failure number	*1*	*2*	*3*	*4*
Time between failures (minutes)	177	143	324	>611

7 RELIABILITY ESTIMATION AND PREDICTION OF DYNAMIC FAILURES

The intention of the method in (Wohlin, 1992) was to use the dynamic failures in a reliability growth model that can estimate and predict the reliability of the software system for the dynamic failures that the tool environment could identify. The objective was to use the model described by Currit (1986). The number of failures which occurred in the dynamic analysis of the case study is however too small for using these types of models.

The failure data collected during the dynamic analysis can be used in a number of ways, for example:

- certify the reliability of the specification or design;
- estimate and predict the reliability for all failure types based on the expansion of dynamic failures to arbitrary failures;
- planning and controlling purposes, for example:
  - a decision basis whether or not to leave the specification or design phase;
  - planning of test resources based on the prediction of the reliability;
  - as an early prediction of the release time;
  - as a means to control the reliability of the system in a software project.

As it was impossible to use a software reliability growth model, a reliability demonstration chart, (Musa, 1987), is used instead. The purpose of this chart is to demonstrate that the software system meets the reliability objective with a given confidence. The assumptions made for this reliability demonstration chart are that the usage cases are derived from an operational profile and that no faults are removed. The first assumption is met, see Section 5, but the second is not met because the faults are corrected when they are discovered. If the faults are corrected the software's reliability will increase and the actual reliability is underestimated, i.e. the estimate is on the safe side. If corrections are made, it is proposed here to use the rejection line, see Figure 3, as a reset line, as after the correction it can actually be viewed as a new and improved software product and therefore it is not reasonable to reject it.

The objective of the reliability demonstration for the case study is that the mean time between dynamic failures are greater than 4.5 hours. The objective is used to normalize the failure times from the analysis. To calculate the rejection and acceptance lines in Figure 3 three other parameters are needed. The calculation of the lines is described by Musa (1987). The three parameters are as follows:

- The probability, α, to say that the objective is not met when it is, is 0.10, i.e. the probability to reject a product fulfilling the reliability requirement.
- The probability, β, to say that the objective is met when it is not, is 0.10, i.e. the probability to accept a product not fulfilling the reliability requirement.
- The discrimination ratio, γ, is 2. This value is recommended by Musa (1987).

The resulting reliability demonstration chart is shown in Figure 3. It can be seen that the objective is met during the analysis of the fourth usage case.

8 DYNAMIC FAILURES TO ARBITRARY FAILURES

The dynamic analyser can only find faults for which it is designed and these are only a subset of all failures that can occur during operation. To be able to estimate and predict the reliability during operation the dynamic failures must be mapped into arbitrary failures. The mapping of the dynamic failures into arbitrary failures are based on two assumptions:

- The set of failures found during dynamic analysis is a subset of all possible failures.
- The failures found during dynamic analysis are randomly spread among all failures, i.e. the ratio between the number of arbitrary failures and the number of dynamic failures during a certain period is an expansion factor called C.

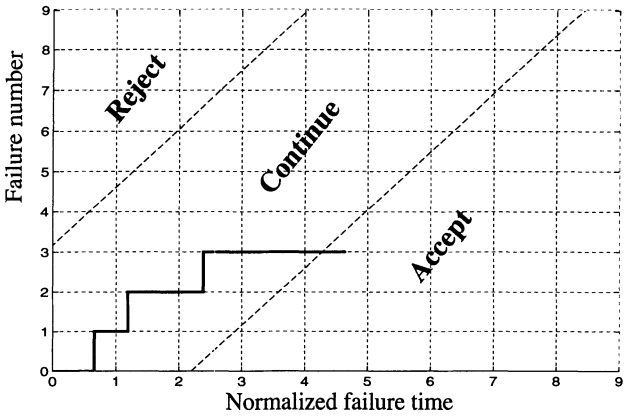

Figure 3 Reliability demonstration chart.

The expansion factor C is assumed to be known based on experience. The objective is to identify a realistic expansion factor between dynamic failures and arbitrary software failures. This is, however part of the future research and hence we use a hypothetical expansion factor here to illustrate how the information from the dynamic analysis can be used.

For the case study, the expansion factor is assigned the hypothetical value of 2.4, which means that for every dynamic failure there are 2.4 arbitrary failures in average during testing. The value of 2.4 is chosen by random to illustrate the method.

To map the dynamic failures into arbitrary failures, there are two things that must be done. First determine how many new failures that should occur in every interval from the dynamic analysis and the second is to place the new failures on the time scale.

The number of new failures in every interval is determined by the expansion factor C. If C is not an integer, the number of new failures to occur in every interval is determined from a two-point distribution with the possible values, trunc(C-1) and trunc(C), with the mean of C-1. If C is an integer, then C-1 new failures occur in each interval. If the last interval is open, i.e. the time between failures is not found, trunc(C/2) new failures are placed in this interval. A uniform distribution is used to place the new failures in the interval. This procedure is used to illustrate the method, but a realistic placement procedure must be determined as more experience is gained.

For the case study, the expansion factor of 2.4 implies that in every interval there should occur 2 new failure with the probability of 0.4 and 1 new failure with the probability of 0.6. In the last interval 1 failure is placed. The mapping to arbitrary failures for the case study is shown in Table 3.

Table 3 Arbitrary failures for the case study

Failure number	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
Time between failures (minutes)	100	22	55	87	56	22	302	237

9 RELIABILITY ESTIMATION AND PREDICTION

In Section 8, the dynamic failures were mapped into arbitrary failures. The arbitrary failures are failures that should have occurred if the system had been in test or operation. The arbitrary failure data can now be used in different ways, for example:

- estimate the reliability of the software when it is released for testing;
- estimate and predict the reliability when the software is in test and operation, which in particular includes the release time.

This information can be used to plan the test resources so that the software can be released at the right time with the required reliability.

From the failure data in Table 3, it is possible to estimate and predict the reliability in the case study at different points of time. The model used is presented in detail in (Currit, 1986), and it is a model to estimate the current reliability as well as to predict future reliability growth. The model is based on the following formula:

$$MTBF_k = A \times B^{k-1} \tag{1}$$

with $A = MTBF_1$ and k is the failure number. The variables A and B can be determined using linear regression to the log of the times between failures. The resulting graph is presented in Figure 4, although the number of data points is limited. From the graph it can be seen that the system in the case study has an estimated *MTBF* of 145 minutes at the release time. It is also possible to predict the *MTBF* for the system during the operational phase. The *MTBF* is predicted to be 175 minutes when the first failure have been discovered and corrected during operation. After the second failure is corrected the *MTBF* is predicted to be 210 minutes. These predictions can be made many steps ahead using the growth model when the variables A and B are determined.

The reliability requirement to release the software must be connected to the acceptance criterion of the design, i.e. the criterion based on the dynamic failures. Thus, the acceptance criterion can be derived from the overall reliability requirement and the two expansion factors described above.

10 CONCLUSIONS

The method presented can be used to estimate and predict software reliability from specification and design documents written in a well-defined high level description technique. It has been shown, through, a case study, that the usage-oriented approach to analysis is feasible and that valuable information can be extracted from the failure data obtained.

In particular, the data can be used for several purposes:

- acceptance of a particular specification or design;
- planning and controlling of the test phases, with particular emphasis on the system test when it is carried out as a statistical usage test;
- estimation and prediction of the reliability as the software enters the testing phase and also as a means for predicting when the reliability requirement is fulfilled.

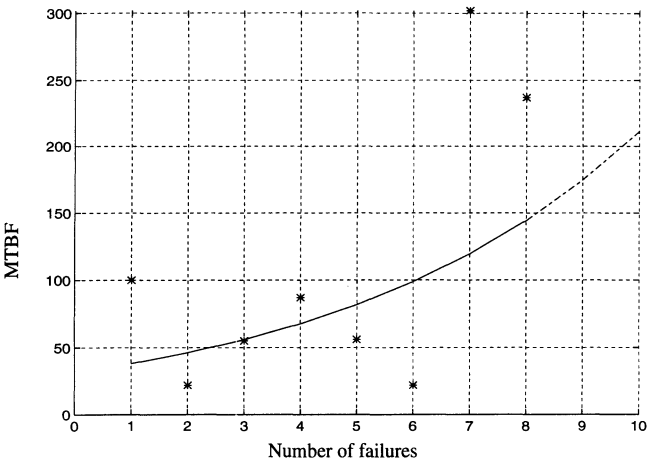

Figure 4 Reliability growth for the case study.

The method is based on two important expansion factors, which must be determined for each organization and application adopting this technique, namely time expansion from design states to real time and fault expansion from the dynamic failures that the tool can find to arbitrary software failures. The determination of these factors for the case study is part of the future work. The research objective is to take the case study through both the test phase and into operation to evaluate the proposed early software reliability estimation and prediction method.

The method ought to be a valuable tool in the future to stay in control of the software quality attributes, and to get early indications of the software reliability status. Thus, the method will help to achieve quality software.

11 REFERENCES

Belina, F., Hogrefe, D. and Sarma, A. (1991). *SDL with Applications from Protocol Specifications*. Prentice-Hall, London.

Currit, P.A., Dyer, M. and Mills, H.D. (1986) Certifying the Reliability of Software. *IEEE Transactions on Software Engineering*, **11(12)**, pp. 1411-23.

Ek, A. and Ellsberger, J., (1991) A Dynamic Analysis Tool for SDL, in *SDL '91: Evolving Methods* (ed. R. Reed and O. Færgemand), Elsevier Science Publisher B V (North Holland), pp. 119-34.

ITU-T Recommendation (1988) *Z.100: Specification and Description Language, SDL*, Blue book, Volume X.1.

ITU-T Recommendation (1993) *Z.120: Message Sequence Chart (MSC)*.

Mills, H. D., Dyer, M. and Linger, R. C. (1987) Cleanroom Software Engineering. *IEEE Software*, **September**, pp. 19–24.

Musa, J. D., Iannino, A. and Okumoto, K. (1987). *Software Reliability, Measurement, Prediction and Application*. McGraw-Hill Int.

Musa, J. D. and Everett, W. W. (1990) Software Reliability Engineering: Technology for the 1990s. *IEEE Software*, **November**, pp. 36-43.

Musa, J. D. (1993) Operational Profiles in Software Reliability Engineering. *IEEE Software*, **March**, pp. 14-32.

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

Runeson, P. and Wohlin, C. (1995a) Statistical Usage Testing for Software Reliability Control. *Informatica*, **19(2)**, pp. 195-207.

Runeson P., Wesslén A., Brantestam J. and Sjöstedt S. (1995b) Statistical Usage Testing Using SDL", Accepted for publication, to appear in Proceedings SDL Forum, Oslo, Norway, September 1995.

Wesslén, A. and Wohlin, C. (1995) Modelling and Generation of Software Usage", Accepted for publication, to appear in Proceedings International Conference on Software Quality, Austin, Texas, USA, October 1995.

Wohlin C. and Runeson P. (1992) A Method Proposal for Early Software Reliability Estimations", Proceedings 3rd International Symposium on Software Reliability Engineering, Raleigh, North Carolina, USA, pp. 156-163.

Wohlin, C., and Runeson, P. (1994) Certification of Software Components. *IEEE Transactions on Software Engineering*, **20(6)**, pp. 494-499.

Yeh C., Reneby L., Lennselius B. and Sixtensson A. (1989) An Educational Development System Employing SDL Design and Automatic Code Generation, Proceedings SDL Forum, Lisboa, Portugal.

12 BIOGRAPHY

Wesslén, Anders - Mr. Wesslén is Ph.d. student at the department of Communication Systems, Lund University, Lund, Sweden and he has an MSc in Computer Science and Engineering from the same university. His research is focused on requirements engineering, statistical usage testing and to quantify software quality attributes early in the development process.

Wohlin, Claes - Dr. Wohlin is associate professor at the department of Communication Systems, Lund University, Lund, Sweden. He has five years of industrial experience from software projects with object orientation techniques, quality assurance, simulation techniques, test methods, and prediction of system and organizational qualities. Claes Wohlin is currently responsible for the education and research in the area of software engineering in telecommunications at the department. His research interest includes methods and modelling techniques to achieve quality software, statistical usage testing and process improvement. Dr. Wohlin has published more than 35 papers in technical journals and at international conferences.