C. Wohlin, "Improving through an Incremental Approach", Proceedings 2nd European Industrial Symposium on Cleanroom Software Engineering, Berlin, Germany, 1995.

Improving through an Incremental Approach[1]

Claes Wohlin
Department of Communication Systems,
Lund Institute of Technology, Lund University,
Box 118, S-221 00 Lund, Sweden
Phone: +46-46-103329, Fax: +46-46-145823, E-mail: claesw@tts.lth.se

Abstract

A major potential problem in software development is the contradiction between on-time delivery and fulfilment of reliability requirements of the software. Therefore, improvements are always needed to support on-time delivery without sacrificing the reliability requirement. Incremental development and certification allows for a better combination of on-time delivery and reliability fulfilment than other development approaches. Therefore, our research is very much focused on the incremental approach, which includes both improving the specification technique for increments and certification of increments. The main reason for focusing on Cleanroom from an incremental viewpoint is the conviction that an incremental approach is superior when it comes to achieving the right reliability in the shortest possible time. To illustrate this conviction a method to determine a suitable order of development and certification of increments is presented before giving an introduction to research areas pursued by the software engineering research group at the department of Communication Systems.

1. Introduction

The basis for managing software quality is control throughout the development. This can be achieved by incremental development, since the basic idea with incremental development is to implement an executable part of the system. The incremental approach allows for reliability certification of each increment and in particular the software system may grow and thus it is possible to certify the cumulative growing system. This approach is advocated in Cleanroom Software Engineering, [Mills87, Linger94].

The real benefit from the incremental approach can only be enjoyed if the increments are developed in an appropriate order. The order must support managers so that the software system meets the reliability requirements in as short time as possible. It is possible to find such an order and the objective of the paper is to present a method, which gives the best order to fulfil both reliability and time constraints. The method is based on mapping the requirements into increments, where the mapping must be made so that a suitable development and certification order can be determined. The order of increments is based on usage of the increments. It is discussed why the proposed method is superior to other approaches in choosing the development and certification order of the increments. It is concluded that incremental development

1. This work is supported by the National Board for Industrial and Technical Development, Sweden, Dnr: 93-2850.

and certification do help the software manager to fulfil the reliability requirements in shortest possible time.

The paper starts with presenting the incremental approach from a reliability and on time delivery point of view. The ability to certify increments is an important ingredient in determining a suitable order of increments, hence reliability certification is discussed briefly. Thereafter, the method which recommends a certain order of the increments is presented. Furthermore, the method is illustrated by reasoning through an example in order to show that the proposed incremental approach is superior to other possible orders of the increments. The proposed order of increments, hence also gives guidelines of how to actually divide the requirements into suitable increments to improve the overall cycle time.

After having shown the superiority with an incremental approach, a brief survey of the Cleanroom related research carried out at the department of Communication Systems is presented.

2. Incremental approach

The discussion is mainly concerned with two quality attributes: time to delivery and reliability. These two are always potentially contradictory. The procurer of a software product may get the product on time, but the software has not been thoroughly verified before the release due to delays. The delays often means taking short-cuts in the verification and validation of the system before releasing the software. In other words, poor software, in terms of reliability, may be delivered due to time constraints. The testing is not carried out as it ought to; it may even happen that untested software hits the field.

The requirement on a specific release date often may overrule other quality objectives as for example reliability. This is of course unacceptable.

The design and coding of one increment are followed by testing of that increment, which makes it possible for the developers to start implementing the next increment while the testers validate, verify or certify the first developed increment. Here, it is assumed that development and certification are performed by different teams, which also is one of the principles in Cleanroom.

The incremental approach hence allows for a good deal of parallelism between development and certification. The benefit from this parallelism is not only the possibility to work in parallel, but also that the testers really start certifying the software to be delivered at an early stage. This property is essential in the discussion below, since it supports early reliability control through certification of increments.

Two major questions arise when discussing incremental development:

- How are increments identified from a requirements specification?
- In which order ought the increments to be developed?

The first question is very hard to answer from a general point of view. It is important that the persons specifying the system have a good knowledge and experience of the application

domain. The formulation of suitable increments is still a creative process. Some guidelines can be found when answering the second question, which is further discussed below.

3. Certification

Certification is the control of the quality fulfilment, for example to certify that a specific reliability has been achieved. The basis for reliability certification is usage testing. Reliability certification consists of two main constituents:

- Usage specification, which includes both a usage model and a usage profile. The model must describe the usage as it is expected in the operational phase. This means describing a user of the system in terms of states of the user and actions a user of the software can perform. The profile describes the relative frequencies to perform certain actions, i.e. the actions are assigned probabilities. The usage specification is used to select test cases which resemble the anticipated usage of the software when in operation. Some different techniques to model usage are discussed in [Whittaker93, Runeson92, Musa93].
- Reliability estimation and prediction procedure, which includes different techniques to perform estimation and prediction. Several different methods and models are presented in the literature. The methods range from sampling techniques, [Parnas90], to software reliability growth models, [Currit86], via hypothesis testing methods, [Musa87]. The choice of the best method and model may differ between different development environments.

The objective with this presentation is not to present reliability certification techniques in any depth, but to show that the techniques are available and to introduce the constituents briefly. A detailed description of the reliability certification method advocated in our research is presented in [Runeson95a]. The description includes both usage specification and reliability estimation technique. Therefore, it can be stated that it is possible to estimate and predict the reliability or the mean time between failures (MTBF) applying usage testing and reliability certification. Two case studies when applying statistical usage testing are presented in [Runeson95b].

It will hereafter be assumed that it is possible to model the usage of a system or an increment, generate test cases according to the expected usage, collect failure data, estimate and predict the MTBF. This assumption is essential in the presentation below. Usage testing can not only be applied to increments and systems, but also to components to ensure that components with a certified reliability level can be reused. The certification of components is further presented in [Wohlin94a].

In the incremental model, the reliability certification can be applied on each increment, since the definition of an increment is that it is an executable part of the system. The first certification is performed on the first increment, while the second certification is done on increment one and two together and so on. The system grows increment by increment until the complete system finally can be certified. A usage specification must be formulated on an incremental level to allow for the certification. The ability to certify the system incrementally is one of the major advantages with incremental development.

4. Order of increments

The introduction of incremental development will probably cause some initial problems, since it requires a cultural change in an organization. This may mean that special considerations must be taken when choosing the order of increments in a first project. Some special considerations may be necessary in other projects as well, but the objective here is to present the best order without considering the special cases which may be due to, for example identified risks, political reasons, staffing problems or available test equipment.

The method presented below gives an optimum order from a reliability point of view, without taking any special conditions into consideration. This is the only way to determine a suitable order, it is then up to the user of the proposed method to adopt it and apply it with any special needs taken into account.

A full mathematical treatment of optimizing the order of increments is not presented. The major problem with a mathematical treatment is that it requires a number of assumptions concerning parallelism between development and certification. An example: Do we continue the certification process if the development of the next increment is not ready as we have certified the required reliability level? This is just one of many such questions where assumptions have to be made and instead of making all these assumptions general arguments are given here. The main arguments of why an optimal order exists and why it has to be based on usage are as follows:

- The most used increments must be in the certification process the longest as it is necessary to ensure the highest reliability level for these to obtain a reliable system. This motivates why the most used increment must go into the certification process first. After certification of the first increment, this increment is certified together with the second increment and so forth.
- The last increment, in some sense, determines the time when the overall reliability requirement is fulfilled. The whole system is now in the certification process and it is ready for delivery when the system is certified. At this point of time, no development is carried out in parallel and hence it is up to the certification team to ensure that the software system can be released in accordance with the contract. It is from this reasoning clear that a high usage increment would cause problems as it has to have a high reliability to give a high system reliability. Therefore, it is quite clear that it is advantageous to let a low usage increment be the last increment. It may even be the case that the system reliability requirement is fulfilled without certifying the last increment, if it is used seldom.

The method for determining the order of increments is hence based on the usage. Thus, the increments should be developed and certified in a descending usage order. It may however be difficult to rank them exactly and therefore it is suggested to start by dividing the increments into different usage classes, for example:

1. High usage increments

   These are the most important increments from a reliability point of view, since high usage means that the reliability of the increments must be high to fulfil the overall reliability requirement.

2. Average usage increments

These increments represent the next class. Increments in this class must have quite high reliability to ensure a high system reliability.

3. Low usage increments

   The low usage increments are not that important from a reliability point of view since they are seldom used and their contribution to the overall system reliability is quite small.

These classes also support the specifiers in their task. In their work they must try to collect as homogeneous increments as possible, hence distributing the requirements on suitable increments to obtain a structure which supports the classes defined above. The definitions of these classes must act as guidelines for the specifiers when identifying increments.

The above order is based on the following:

- Other considerations must be taken into account when applying the proposed order, for example risks, availability of test equipment and so forth. The order above should be viewed as general guidelines which must be combined with common sense and the actual situation at hand.
- The order above is also based on the assumption that the usage profile is not changed as new increments are added to the system. The probabilities must be rescaled to add up to one, but it is assumed that the initial overall profile is not changed. The changing of the profile is one way of trying to accelerate the certification to resemble the order proposed above, when other considerations mean that the order has to be abandoned, due to example risk considerations.

5. Time to delivery

5.1 Introduction

The actual time to delivery depends on a large number of factors, for example priority given to the project, project management, complexity of the problem and reliability requirements. These factors are here assumed to be possible to describe with development time and certification time. The development time is assumed to take a certain amount of time (in terms of calendar time), while the certification time primarily is a function of the initial reliability of the software as it is delivered to the certification team, the reliability growth during certification and the reliability requirement.

5.2 Incremental: any order vs. usage order

The major difference between the any order approach and the usage order is in the certification process as it is assumed that the development time is indifferent based on the order of the increments. The order, however, is essential for the certification process as the objective is to fulfil an overall reliability requirement. The overall reliability is a weighting between the reliability of the increments of the system, where the weighting is based on the usage frequencies.

The need to develop and certify the increments in usage order is based on that the first increments will achieve most testing, when assuming that the overall profile is not changed during

the development and certification, see above. This is due to that the recommended certification procedure in Cleanroom, i.e. first to certify increment 1 and then as increment 2 is developed to certify increment 1 and 2 together. This procedure is repeated until the complete system has been put together. Hence, if the system consists of N increments then the first increment will go into the certification procedure N times.

The reasoning about which increments that will achieve the most testing is the key to why there is a best order. The increment which contains the functions which are anticipated to be used the most must therefore go first into the certification, as the first increment will get tested more than the later ones. Only in this way will the weighting of the reliability of the different increments reach the overall requirement in the shortest possible time.

The above reasoning is also illustrated in a simple example in figure 1, where it is assumed that the order of the increments means that they have the same reliability in the two alternatives after the certification. This assumption is impossible to justify as the usage probabilities will influence the reliability growth of the specific increments, but the objective is only to illustrate that we will indeed fulfil the overall reliability requirement in a shorter period of time if determining a sensible order. There is actually no better assumptions that can be made unless taking a larger example and making a large number of assumptions. In the example, it is also assumed that the total certification time is the same for the two alternatives presented in figure 1. This is comparable to showing that it takes a longer time to certify the overall reliability requirement for the first alternative.

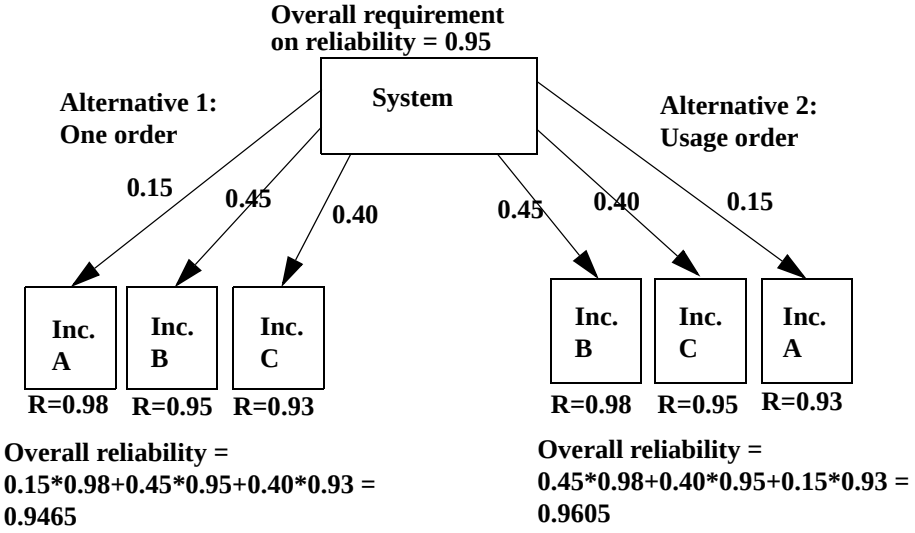

FIGURE 1. Two alternative incremental orders.

From the example, it can be seen that in alternative 1 the overall reliability after a certain total certification time becomes 0.9465, while for alternative 2 the overall reliability becomes 0.9605 in the same time. Thus, in alternative 2 the overall reliability requirement, which is 0.95, is actually more than fulfilled and for alternative 1 the requirement is not fulfilled, although it is close. Therefore, it is concluded that there is a best way from a usage perspective and that the objective must be to try to identify increments which corresponds to the order when trying to identify suitable increments as the incremental construction plan is written.

To really evaluate the difference a more extensive example or even better a practical case study must be conducted. A somewhat larger example is presented in [Wohlin94b], where the order is discussed in more detail.

6. Conclusions

Two important principles proposed in Cleanroom, incremental development and certification, have been discussed. The benefits with both incremental development and certification have been emphasized and in particular the advantages with combining incremental development with certification techniques have been presented. Incremental development and certification are methods which support managers in their work to stay in quality control in terms of reliability and software release time.

The proposed usage based order of increments is superior to other orders of incremental certification, hence providing a higher reliability in a shorter time period. The usage order also provides some guidelines when trying to identify a suitable division into different increments.

The possibility to develop as high reliability software in as short time period as possible is a major challenge and thus incremental development in general ought to be applied, and the usage order of increments will improve the situation even further. Some studies indicate that high reliability is highly correlated with high maintainability, [Stålhane93], which means that the result presented becomes even more important.

Managing software quality means adopting incremental development and certification, which provide higher reliability (and maintainability) in a shorter time period, hence allowing for on-time delivery with reliability control and fulfilment of the quality requirements.

7. Research areas

7.1 Usage model development based on formal use case specifications

Object-Oriented Software Engineering [Jacobson92] and Usage Testing have each of them a concept for the usage of the system. Use cases are discussed in Object-Oriented Software Engineering while a usage model is depicted as important in Usage Testing. These two concepts are similar but are used in different ways. If the development of the system and the testing shall have the same basis, only one concept is preferred to make it easy for the procurer to understand the process.

If the use case concept is formalized, it can form the basis for both object-oriented development and statistical usage testing. To use the formal use case concept in statistical usage testing there has to be methods for building usage models from use cases.

An issue closely related to the above research is the comparison between Object-Oriented Software Engineering, for example as presented in [Jacobson92], and Cleanroom. A comparison between the approach in [Jacobson92] and an object-based Cleanroom approach is presented in [Cosmo95a].

7.2 Object-oriented development and usage testing of components

Object orientation is depicted as a suitable development strategy to develop reusable components, but a system can not be composed of components from a repository without having a certified reliability. Therefore, it is important to develop methods to certify reliability of components and also to formulate methods to derive system reliability from the reliability of the system components. The research includes:

- Usage testing and reliability certification of software components
- Approaches to derive system reliability from the components of the system

Some results have been presented within this problem domain [Poore93, Wohlin94a], but many problems remain to be solved.

7.3 Usage evaluation from a life cycle perspective

Above, usage testing has been described as a suitable technique to allow for a better evaluation of the software prior to the operational phase, but the concept can be applied throughout the life cycle. A method proposal for usage analysis of a software design has been proposed in [Wohlin92] and the objective of the research is to develop this approach further. The aim is also to develop a simulation procedure based on the usage testing concept. Some preliminary results are presented in [Wohlin94c].

The research includes:

- Formulation of a usage evaluation process, which enlarges the usage testing approach. Some findings are presented in [Wesslén95].
- Further development of the method for usage analysis, which allows for a first reliability evaluation during analysis of the software design. A first enlargement and improvement of the method is presented in [Wohlin95a].
- Formulation of a usage simulation method, which can either be a functional simulation or it may include real time aspects which means that capacity issues can be evaluated as well.

These research areas must be combined with usage testing results to obtain a comprehensive usage evaluation method which covers most of the life cycle.

7.4 System re-certification without re-testing

A major problem in reliability certification is that the certified value is only valid for the applied usage profile. The reliability estimate may be considered trustworthy for minor deviations from the applied profile, but there is no guarantee, and it is not relevant for major changes in the profile. It is, however, not cost-effective to re-test the software as the usage profile changes or a new service is included in the system. Instead it is favourable if it is possible to re-calculate the system reliability based on a prior certification and the known changes.

Some preliminary findings have been presented in [Wohlin95b], but more research is needed, as for example:

- Development of a method for sensitivity analysis of the reliability estimate as the usage profile changes
- Improvement of the method, [Wohlin95b], to re-certify the system reliability based on a calculation procedure instead of re-testing the software

7.5 Incremental development

Incremental development as defined in [Mills87] requires that a specification of the total system and one specification for each increment is done. When using this approach, the specification and verification of the increments are very time consuming since each increment has to be specified and verified separately. In addition to this the specifications of each increment must when being appended together be equivalent to the specification of the total system.

Incremental development is very appealing when requirements change a lot, which is almost a rule in software development. When implementing a changed requirement, developers are supported by good traceability from requirement to specification down to design and implementation.

Some results are presented in [Cosmo95b], but further research is needed, for example:

- A Use Case Specification that can act both as a specification of the total system and as a specification of each increment must be defined
- Identification of a closer relationship between requirements and its specification is needed
- Identification of a closer relationship between specification and its components is needed
- Feature interaction and incremental development must be investigated. A first attempt to combine ideas in feature interaction with a statistical analysis approach is presented in [Kimbler95].

Acknowledgement

I would like to thank the people in the software engineering research group working with Cleanroom related research for valuable comments on an earlier version of this paper. The Cleanroom related research is primarily conducted by Henrik Cosmo, Per Runeson, Anders Wesslén, Björn Regnell, Kristofer Kimbler and myself. I would also like to acknowledge the comments made by Professor Jesse Poore, which have increased the quality of the paper considerably.

References

[Cosmo95a] Cosmo, H., Wohlin, C., and Johansson, A., “An Empirical Study: Object-Based Cleanroom vs. Object-Oriented Software Engineering”, Submitted to 5th European Software Engineering Conference, Sitges, Barcelona, Spain, 1995.

[Cosmo95b] Cosmo, H., “The Black Box Specification Language for Software Systems”, Proc. 2nd Annual European Industrial Symposium on Cleanroom Software Engineering, Berlin, Germany, 1995.

[Currit86] Currit, P. A., Dyer, M., and Mills, H. D., “Certifying the Reliability of Software”, IEEE Transactions on Software Engineering, Vol. SE-12, No. 1, pp. 3-11, January 1986.

[Jacobson92] Jacobson, I., Christerson, M., Jonsson, P., and Övergaard, G., “Object-Oriented Software Engineering – A Use Case Driven Approach”, Addison-Wesley Publishing Company and ACM Press, 1992.

[Kimbler95] Kimbler, K., and Wohlin, C., “Statistical Usage Analysis to Detect Feature Interaction”, Proc. Telecommunications Information Networking Architecture, pp. 219-230, Melbourne, Australia, 1995.

[Linger94] Linger, R. C., “Cleanroom Process Model”, IEEE Software, pp. 50–58, March 1994.

[Mills87] Mills, H. D., Dyer, M. and Linger, R. C., “Cleanroom Software Engineering” IEEE Software, pp. 19-24, September 1987.

[Musa87] Musa, J. D., Iannino, A. and Okumoto, K., “Software Reliability: Measurement, Prediction, Application”, McGraw-Hill, New York, 1987.

[Musa93] Musa, J. D., “Operational Profiles in Software Reliability Engineering”, IEEE Software, pp. 14-32, March 1993.

[Parnas90] Parnas, D.L., van Schouwen, A. J. and Kwan, S. P.,”‘Evaluation of Safety-Critical Software”, Communications of the ACM, Vol. 33, No. 6, pp. 636-648, June 1990.

[Poore93] Poore, J. H., Mills, H. D., and Mutchler, D., “Planning and Certifying Software System Reliability”, IEEE Software, pp. 88-99, January 1993.

[Runeson92] Runeson, P. and Wohlin, C., “Usage Modelling: The Basis for Statistical Quality Control”, Proc. 10th Annual Software Reliability Symposium, pp. 77-84, Denver, Colorado, USA, 1992.

[Runeson95a] Runeson, P. and Wohlin, C., “Statistical Usage Testing for Software Reliability Control”, Accepted for publication, to appear in Informatica, Vol. 19, 1995.

[Runeson95b] Runeson, P., Wesslén, A., Brantestam, J., and Sjöstedt, S., “Statistical Usage Testing using SDL”, Accepted for publication, to appear in Proc. 7th SDL Forum, Oslo, Norway, 1995.

[Stålhane93] Stålhane, T. and Wedde, K. J., “The Quest for Reliability – A Case Study”, Proc. 2nd Int. Conf. on Achieving Quality in Software, pp. 309-320, Venice, Italy, 1993.

[Wesslén95] Wesslén, A. and Wohlin, C., “Usage Modelling and Generation of Validation and Verification Cases”, Submitted to International Conference on Software Quality, Austin, Texas, USA, 1995.

[Whittaker93] Whittaker, J. A., and Poore, J. H., “Markov Analysis of Software Specifications”, ACM Trans. on Software Engineering Methodology, Vol. 2, No. 1, pp. 93–106, 1993.

[Wohlin92] Wohlin, C., and Runeson, P., “A Method Proposal for Early Software Reliability Estimations”, Proc. 3rd Int. Symposium on Software Reliability Engineering, pp. 156-163, Raleigh, North Carolina, USA, 1992.

[Wohlin94a] Wohlin, C. and Runeson, P., “Certification of Software Components”, IEEE Transactions on Software Engineering, Vol. 20, No. 6, pp. 494-499, 1994.

[Wohlin94b] Wohlin, C., “Managing Software Quality through Incremental Development and Certification”, In Building Quality into Software, pp. 187-202, edited by: M. Ross, C. A. Brebbia, G. Staples and J. Stapleton, Computational Mechanics Publications, Southampton, United Kingdom, 1994.

[Wohlin94c] Wohlin, C., “Evaluation of Software Quality Attributes during Software Design”, Informatica, Vol. 18, No. 1, pp. 55-70, 1994.

[Wohlin95a] Wohlin, C., Runeson, P., and Wesslén, A., “Software Reliability Estimations through Usage Analysis of Specifications and Designs”, Submitted to International Journal of Reliability, Quality and Safety Engineering, 1995.

[Wohlin95b] Wohlin, C., “Re-Certification of Software Reliability without Re-Testing”, In Software Quality and Productivity: Theory, Practice, Education and Training, pp. 219-226, edited by: M. Lee, B-Z Barta and P. Juliff, Chapman & Hall, London, UK, 1995.