4

Automating software quality modelling, measurement and assessment

Barbara Kitchenham
Department of Computer Science, University of Keele
Keele, Staffordshire, ST5 5BG, UK, Phone: +44 1782 583075
e-mail: barbara@cs.keele.ac.uk

Alberto Pasquini
ENEA
Via Anguillarese 301,I-00060 Roma, Italy, Phone: +39-6-3048 6038,
e-mail: pasquini_a@casaccia.enea.it

Uwe Anders
TÜV-Nord
Grosse Bahnstasse 31, Postfach 54 02 20, D-22502
Hamburg,Germany, Phone: +49 -40-8557 2429, e-mail:
100632.430@compuserve.com

Jørgen Bøegh
DELTA Dansk Elektronik, Light & Acoustics
Venlighedsvej 4, DK-2970 Hoersholm,Denmark, Phone: +45-
427722, e-mail: jb@delta.dk

Stephano de Panfilis
Engineering, Ingegneria Informatica Spa
Via Dei Mille 56, I-00185 Roma, Italy
Phone: +39-6-3048 6038, e-mail: depa@mail.eng.it

Stephen Linkman
Department of Computer Science, University of Keele
Phone: +44 1782 583075, e-mail: steve@cs.keele.ac.uk

D. Gritzalis (ed.), *Reliability, Quality and Safety of Software-Intensive Systems*

Abstract

This paper describes the SQUID approach to modelling, measuring, and assessing software quality. SQUID is an ESPRIT III project that has developed a method and an experimental toolset to support and automate these activities. The toolset assists in quality specification, quality planning, quality control and quality evaluation. More specifically, it provides the means to establish targets for the product quality requirements and evaluate their feasibility (quality specification). Then, the toolset supports the identification of the internal software product and process attributes that must be controlled during the development process to fulfil the project quality requirements (quality planning and control). Finally, the toolset helps to assess the fulfilment of the project quality requirements (quality evaluation).

Keywords
Software Quality, Software Measurement, Software Progress Monitoring, Software Quality Prediction, Software Quality Specification

1 INTRODUCTION

This paper introduces the SQUID (Software QUality In the Development process) approach to modelling, measuring and assessing software quality and describes its method of automating its approach. The SQUID approach is derived from a number of different research results in the area of quality and measurement.

Gilb suggested defining quality requirements for software products in terms of the operational conditions required by product users (Gilb, 1987). According to Gilb, quality requirements must be expressed in measurable terms identifying target values for each of the specified operational requirements. This approach is now well established and it has been accepted in the International Standard ISO 9126 (1992). To establish quantifiable target values more easily, the standard identifies six quality characteristics for which a quality requirement must be fixed. The standard proposes decomposing the quality characteristics into sub-characteristics and to continue this decomposition process until it is possible to identify sub-characteristics for which a quantifiable target value can be fixed.

The REQUEST project focused on development process rather than on final software product (Petersen and Kitchenham, 1988, Kitchenham and Walker, 1989). REQUEST suggested and implemented ways to model and measure software quality by monitoring the effectiveness of the development process. This was done setting targets for internal software product and process attributes and comparing targets with actuals. The TASQUE project investigated the problem of specifying quality attributes from the user's point of view (Anders et al., 1995). The TASQUE tool allows user requirements to be translated and refined to reflect the programmer's and the quality engineer's viewpoint. Metrics and target values are suggested. Once actual values are identified they can be compared with the targets. TASQUE, however, uses one of several *fixed* quality models. MERMAID emphasised the importance of environmental factors when collecting measures (Kok et al., 1990). The results of the MERMAID project suggested that a process measure has no meaning out of the context of the development model, the company standards, the policies, and the procedures that are used during the development process. For this reason, a quality model must be integrated with the

specific development model that is used within an organisation. In addition, MERMAID defined a set of statistically driven estimation procedures.

SQUID combined the results of these projects to develop and automate a quantitative method for modelling, measuring and assessing software quality. The SQUID conceptual model of software quality is the basis for the SQUID toolset architecture as shown in Figure 1. It comprises:

- The SQUID Conceptual Data Model which provides the relationships between development model (deliverables, milestones etc.), quality model (quality characteristics, sub-characteristics and attributes), and measures. The SQUID Conceptual data model is the basis of the SQUID toolset database.
- The project quality activities involved in specifying, assuring and controlling software quality which also identifies the high level user-visible functionality of the SQUID toolset.

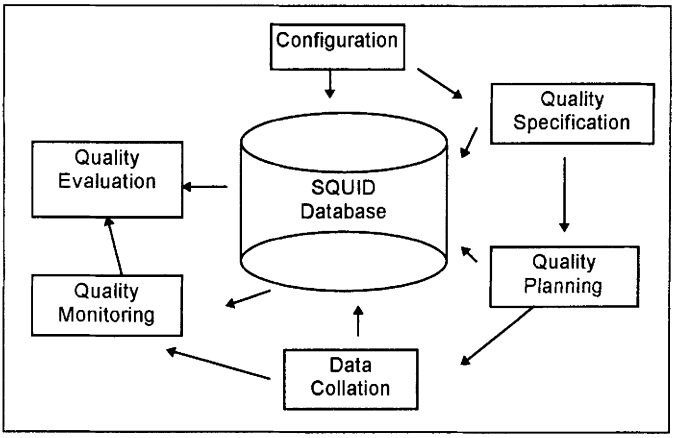

Figure 1 SQUID Toolset Architecture.

This paper describes the SQUID Conceptual Data Model and the associated software quality activities and explains how these concepts are automated in the experimental toolset.

2 THE SQUID QUALITY MODEL

The SQUID Quality Model is a conceptual model with three major elements:

- a development process structure;
- a quality model structure
- measures.

Measures provide the link between the concept of quality requirements and the development process. Quality requirements are specified in terms of measurable properties derived from the intended operational properties of the product. These are external measures. Thus, a quality specification can be viewed as a set of target values for external measures. The

actual values of external measures observed on the operational product can be compared with the targets to evaluate whether or not a quality requirement has been achieved.

The integrated SQUID conceptual quality model is rather complex, but it is possible to understand the principles from a simplified view of the major components. The SQUID development model (Figure 2) comprises projects objects that belong to three different project object types: deliverables, deliverable activities and review points. The SQUID quality model structure is shown in Figure 3. This shows the relationship between quality requirements, and quality characteristics.

The quality model and the development model are linked by the concept of measurable attributes. Our view of measures is shown in Figure 4. It expresses the view that an actual value measures an attribute of a software object in a particular unit and that the value is obtained using a defined counting rule. This follows the structure suggested by Kitchenham et al. (1995). In addition, SQUID emphasises that a value can be obtained by measurement (i.e. can be an actual value) or can be a target or an estimate. Following Gilb, we believe quality characteristics must be decomposed until they can be expressed as the target values of measurable attributes. Quality models identify internal properties that are believed to influence final characteristic levels. Again these can be represented as measurable attributes. Using the REQUEST approach, we set target values of internal measurable attributes with the intention of controlling software development. Quality control and quality assessment then become processes of comparing target values with actuals, to assess progress during development and achievements after completion, and with estimates to assess, during development, whether furture targets are feasible.

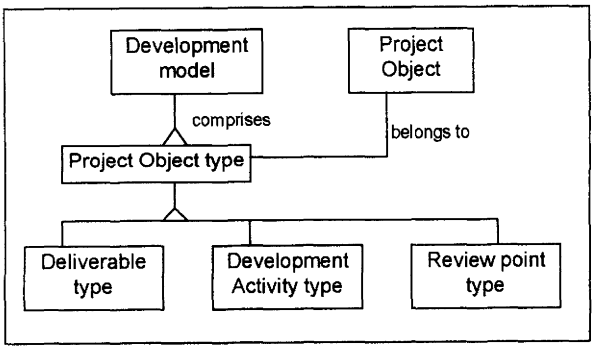

Figure 2 The SQUID view of development.

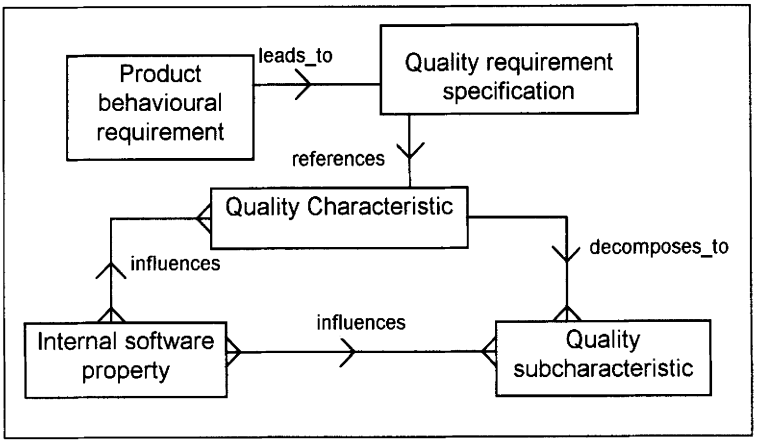

Figure 3 SQUID view of a product quality model.

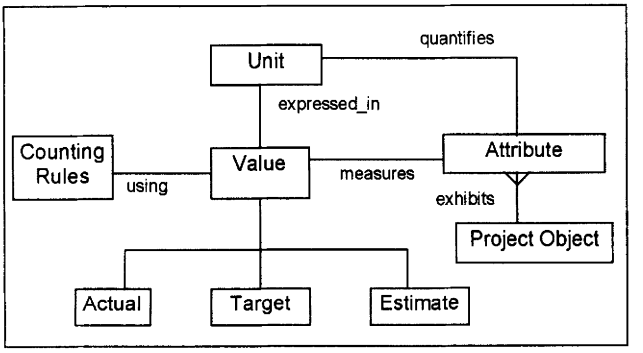

Figure 4 The SQUID view of Measurement.

3 SOFTWARE QUALITY ACTIVITIES

We have identified four major software quality activities that SQUID needs to support:

- Quality specification
- Quality planning
- Quality monitoring and control
- Quality Evaluation.

In order for SQUID to support flexible, organisation specific quality and development models along the lines suggested by MERMAID, SQUID also needs to support the specification of organisational lifecycles and quality models. SQUID refers to this as ***configuration***. In addition, since SQUID is concerned with analysis of actual and target values

of measurable attributes, SQUID also support data collation and storage. Note, SQUID does not provide software data extraction facilities.

3.1 Quality Specification

Quality specification involves defining the operational characteristics required of a new product in terms of target values for a set of quality characteristics. It also involves deciding whether or not the quality requirements are feasible given the capability of the development organisation. Quality specification may involve a first level design decomposition to assess whether there are product subsystems with substantially different requirements.

The SQUID toolset supports quality specification by providing the following facilities:

- a means of defining quality requirements in terms of the operational behaviour of the product;
- a means of specifying quality requirements quantitatively as quantitative targets of external measures;
- a means of predicting the likely quality outcome for a proposed product based on the achievement on past similar products;
- help and advice in the event of quality requirements appearing to be infeasible.

3.2 Quality planning and quality monitoring

Quality planning and quality monitoring are closely linked activities, so they are considered together. SQUID takes the view that quality planning involves setting quantitative targets for attributes associated with intermediate products and processes against which progress will be monitored. This differs from the view of quality planning taken in some standards (for example, the IEEE Standard for Quality Assurance Plans, 1984 and ISO 9001-3, 1991) that concentrate on the definition of quality activities for each lifecycle stage.

SQUID views quality monitoring as a means of assessing product development against the targets set during quality planning and deciding appropriate reaction to any deviations.

In order to support these activities, the SQUID approach and toolset aims to assist a user to:

- assign target values to measurable characteristics of project objects as required by company standards.
- identify deviations by comparing target values with actual values;
- identify other unusual conditions (e.g. by using anomaly detection);.
- determine the reason for deviation or unusual condition;
- evaluate the impact of the deviation and the consequences of any corrective action;
- determine an appropriate response to each deviation or unusual condition.

3.3 Quality Evaluation

Quality evaluation is responsible for assuring that a product has been produced in accordance to the standards and procedures defined in its quality plan. In addition, quality evaluation may also be concerned with assuring that the product is fit for purpose (i.e. satisfies its specified requirements).

Although quality evaluation could be viewed conceptually as a single activity, it is usually necessary to distinguish first, second or third party evaluation. We discuss each of these separately below.

First Party Evaluation

First party evaluation is assessment undertaken by the product development group (usually as part of project management). It involves monitoring conformance to the quality plan and maintaining the records needed to support second and third party evaluation.

Quality evaluation is similar to normal progress assessment, so first party evaluation is supported by the facilities used for specification, planning and monitoring. However, the emphasis is on the measures relating directly to user quality requirements.

Second Party Evaluation

Second party evaluation is usually performed by an independent evaluation or quality assurance role within the product development company. It has two main functions: assurance that a project is organised according to its quality plan, and responsibility for the decision to deliver the product.

Throughout product development this activity involves auditing the project for conformance with the standards and procedures defined in the quality plan. In particular, it must assure that:

- tasks and activities are performed in accordance with process standards;
- deliverables are produced to the specified format and quality standards;
- all required project records are produced and available for inspection.

At the end of product development, second party quality evaluation is responsible for signing-off a product as suitable for delivery. This involves review of product records including system test results and/or performance of specific quality assurance tests. The quality evaluation activity must collate various information into an overall assessment of whether the product is fit for delivery. Collation of quality information requires an assessment process and an assessment model.

Using the SQUID toolset second party evaluators can have access to information about all tasks and deliverables and the final quality achievements. However, the current version of the SQUID experimental toolset does not support the definition of an assessment model.

Third Party Evaluation

Third party evaluation is undertaken by the client (i.e. the purchasing organisation) to obtain an evaluation of product quality. A purchasing organisation may delegate third party evaluation to an independent organisation such as a testing laboratory. Third party evaluation has the same functions as second party evaluation but concentrates on the end-user's view of quality. This view concentrates on issues such as whether the product functions provided actually meet the purchasing organisation's needs, whether the product can be used easily and effectively by its user population, whether the product is reliable enough to allow required tasks to be performed, whether it is simple for the client to migrate from current working practices to new working practices supported by the new product.

Third party and second party evaluation are very similar with respect to the activities involved. The main difference is who performs them. Again the SQUID toolset supports these roles by providing a record of the development process and the measured behaviour of the final product.

An important issue for a toolset that aims to support third-party assessment, is that the toolset itself must support related international standards. Thus, the SQUID method and toolset have been designed to support relevant standard and guidelines (Bøegh and de Panfilis, 1996).

3.4 Configuration

Configuration is an essential element of the SQUID toolset because it allows an organisation to create its own lifecycle and quality models. Configuration involves definition of the lifecycles that project managers may use, including specification of the permitted development processes, and the required documentation and reporting standards.

In order to support this activity SQUID provides:

- a means of creating quality models comprising user-defined quality characteristics, subcharacteristics, internal measures, external measures and their relationships;
- the ability to specify lifecycle models, in terms of permitted project object types (deliverables, activities and review points);
- the ability to associate internal measures with project object types.

One of SQUID's major contribution is a framework for collating the quality view with the development process view via measures. The process of collation ensures that the internal measures identified as relevant in a quality model are associated with appropriate project object types. For example, if a quality model includes structural complexity as an internal software measure, it must be linked to an appropriate project object type (e.g. module) in a development model so that modules in a specific product can be assigned an appropriate structural complexity target.

3.5 Implications of the SQUID toolset

The activities described above imply that an organisation wanting to use the SQUID toolset effectively must have well-defined quality assurance and process definition roles and an extensive data collection capability. This implies that organisations capable of benefiting from the SQUID toolset are most likely to be confined to those that have achieved a fairly high standard of process maturity.

We would not expect organisations at level 1 of the SEI Capability Maturity Model (Paulk et al., 1993) to have the organisational infrastructure necessary to use the SQUID toolset. We would expect organisations at level 3 or above to be able to find the SQUID toolset easy to incorporate into their working environment, but only some of the level 2 organisations would be able to use the SQUID approach. For level 2 organisations, we hope that the SQUID concepts will help process engineers identify improvement options and assist their efforts to achieve level 3.

4 CONCLUSIONS

The SQUID project has developed and automated an approach to quantitative quality modelling. The SQUID experimental toolset is based on two areas of work:

1. The SQUID conceptual quality model that identifies the entities used in the SQUID approach and their relationships. The toolset data model has been derived from this model.
2. An analysis of the quality activities involved in software development and the functions they need to perform. This analysis is the basis of the functionality of the SQUID toolset.

Although the SQUID toolset is still only experimental, we hope our ideas on quality measurement, modelling and evaluation are of interest to quality engineers and our approach to automation is of interest to other tool developers.

5 REFERENCES

Anders, U. et al. (1995) Abschlußbericht (Final Report) TASQUE (Tool for Assisting Software Quality Evaluation), Hamburg.

Bøegh, J. and de Panfilis, S. (1996) SQUID: A method for managing software quality during the development. ESA Product Assurance Symposium and Software Product Assurance Workshop. ESTEC, Netherlands.

Gilb, T. (1087) *Principals of software engineering management.* Addison-Wesley.

IEEE (1984). Standard for Software Quality Assurance Plans. IEEE std.730.

ISO 9126 (1992) Information technology - Software product evaluation - Quality characteristics and guidelines for their use, International Organisation for Standardization, Geneva.

ISO 9001-3. (1991) Quality management and quality assurance standards - Part 3: Guidelines for the application of ISO 9001 to the development, supply and maintenance of software. International Organisation for Standardization, Geneva.

Kitchenham, B.A. and Walker, J.G. (1989) A quantitative approach to monitoring software development. Software Engineering Journal, **4**,1, pp 2-13.

Kitchenham, B.A., Pfleeger, S.L. and Fenton, N.E. (1995) Towards a framework for software measurement validation. IEEE Transaction on Software Engineering, **21**(12), December.

Kok P.A.M, Kitchenham B.A. and Kirakowski, J. (1990) The MERMAID Approach to Software Cost Estimation. Proceedings of the ESPRIT '90, Kluwer Academic Press.

Paulk, M.C., Curtis, W, Chrissis, M.B. and Weber, C.V.(1993) Capability Maturity Model, Version 1.1. IEEE Software, **10** (4), pp 18-27.

Petersen, P.G. and Kitchenham, B.A. (1988).The development of a software quality model. Proceedings 1st European Conference on Software Quality, Brussels.

6 ACKNOWLEDGEMENT

The SQUID project P8436 was partly funded by the European Commission as part of the ESPRIT III Programme.

7 BIOGRAPHY

Barbara Kitchenham is a Principal Researcher in Software Engineering at the University of Keele. Her research interests are centred on software metrics and their application to project management, quality control, and evaluation of software technologies. She has worked both in industry and academia and has been involved with several joint national and international software engineering research projects. She has written numerous articles and published a book on the topic of software measurement. Kitchenham received a PhD from the University of Leeds in 1976. She is an associate fellow of the Institute of Mathematics and Its Applications and is a fellow of the Royal Statistical Society.

Alberto Pasquini received the laurea degree in Electronic Engineering from the University "La Sapienza" of Rome in 1978. He works for the Italian research agency for the new technology, energy and the environment (ENEA). His research interest are in software engineering and, in particular, software reliability, modelling, software testing and software quality.

Uwe Anders received a certificate in Physics in 1969. From 1969-1982 he was active in the fields of semiconductor technology (solar cells for satellite applications), and in data logging and process control in the plastic and rubber industry. Subsequently he worked within TÜV Nord on the Computer-based Systems Group. Currently he is deputy head of the Software & Electronics Laboratory. He has managed several national and European R&D projects related to the assessment of safety related computer systems and the design and development of software assessment tools. He is a member of the European Workshop on Industrial Computer Systems.

Jørgen Bøegh has a degree in mathematics and computer science from Aarhus University. He has been with DELTA Danish Electronics, Light and Acoustics since 1985. His main interests are in the area of software development methods and software quality. He has been involved in the development of software product evaluation technology and is working as an editor of two international standards in this area. He has participated in international projects in the area of software quality, including the ESPRIT projects SCOPE and SQUID. He is the author of several papers and a book about object oriented software development.

Stefano De Panfilis joined Engineering in 1984 working on many aspects of Software Engineering. In 1988 he moved to the R &D Department to be the responsible of the Formal Methods research area. During this time he spent 6 months at Transform Logic headquarters (Scottsdale, Arizona) defining and implementing a case tool integrated with the Transform code generator. In 1991 he moved to the Methodology and Quality Assurance Department where he was responsible for the introduction of the Object-Oriented approach into the Analysis and the Design phases of Engineering development process. In 1993 he participated in the working group which resulted in Engineering being certified under ISO 9001 series as one of the first software houses in Italy. At present he is the Project Manager of the

INNOVATION VALSE project. He has several national and international publications and has participated in several International Conference Program Committees.

Stephen Linkman is Principal Researcher in Software Engineering in the Computer Science Department at the University of Keele. He has interests in Metrics, Risk, Estimation ,Quality and Evaluation and Assessment and has published papers in all these areas. Prior to joining the University of Keele, Linkman was a Principal Consultant with the United Kingdom National Computing Centre. Prior to this he had worked in various development and research roles for International Computers Ltd and its parent company Standard Telephone and Cable. Linkman received a BSc in Engineering from the University of Leicester.