20th IMEKO TC4 International Symposium and
18th International Workshop on ADC Modelling and Testing
Research on Electric and Electronic Measurement for the Economic Upturn
Benevento, Italy, September 15-17, 2014

Software Product Quality: Some Thoughts about its Evolution and Perspectives

Luigi Buglione[1]

[1] *GUFPI-ISMA (Gruppo Utenti Function Point Italia – Italian Software Metrics Association),*
luigi.buglione@gufpi-isma.org, Luigi.buglione@eng.it

Abstract – **From the mid '70s on, a plenty of attention was devoted to evaluate and measure software quality. As Tom Demarco said: "you cannot control what you cannot measure". But coming back, it's also true that "you cannot measure what you cannot define" and again "you cannot define what you don't know". Thus, moving from definition is the priority for any activity and creates also measures from a common, shared definition. The FCM (Factor-Criteria-Model) was the first 'quality model' in 1977 by the US Air Force trying to state a "three-tier" model for defining what quality could be for a software product. Later, Boehm (1978) and ISO (1991) with its first version of the 9126 model (now evolved into the 25010:2011 one, in the SQuARE standard series, did the same exercise. Again, maybe less known, other models and taxonomies have been created and proposed in the technical literature (e.g. FURPS+, ECSS-E-10A, ISO 21351:2005, etc.) for the same purpose.**
What a very few do is to understand (and deal with, accordingly) that 'quality' means 'non-functional' (or at least, a large part of the ISO definition of NFR – Non-functional Requirements). From a measurement perspective it means to deal with a very (relative) unexplored area, with a plenty of possible developments. In fact, a FUR (Functional User Requirement) is something about the 'what' a software can do by its functionalities and FPA (Function Point Analysis) - whatever the variant adopted – in a single number tries to relate a sizing unit expressing such 'functional dimension'.
Dealing with NFR and Software Quality is a very complex work, because of the large number of attributes composing 'quality'. Each category in one of the aforementioned quality models could be a separate issue as well as now is the 'functionality' one. Taking into account several attributes at the same time and determining a 'quality profile' for a certain type of software will be one of the next decade challenges. Estimators will need to understand better and better which NFR-related (quality) measures to include (at least 2+ ones) as independent proxies in estimation models, allowing estimators to reduce MRE (Mean Relative Error) figures as much as possible, saving project resources and improving the overall project value for its stakeholders.
In order to do that, this paper will try to discuss from an evolutionary perspective what software quality has been, is and should/could be perceived and defined during next years, by a measurement perspective.

Keywords – Software Quality, Quality Models, Non-functional Requirements, FPA, SNAP, ISO 25010, GQM.

I. INTRODUCTION

'Quality' is a risky and misleading term because including so many meanings and attributes – even if often seen simply as '*defectability*' - within a single word that often in assessments and evaluations it besides in the 'qualitative' side more than be extended also in the 'quantitative' one, finding proper measures for quantifying it. Thus, questions such as 'which is the value for quality? How to measure quality?' are typical also in the Software Engineering community. It can be quite easy to count something but less to evaluate its quality side, because difficult to express the core question ("what does it mean quality"?).

Tom Demarco said that "*you cannot control what you cannot measure*". But coming one step back, it's also true that "*you cannot measure what you cannot define*". Coming one step back again, "*you cannot define what you don't know*". Thus, it's a knowledge problem and the priority is to move from a common, shared definition. Reading these three statements in the opposite order, (1) if you know something, you're able to properly describe it and share such definition with others; (2) if you're able to share definitions, it'll be easier to quantify such 'thing' in the same way (looking at metrology, two measurers should vary very few counting/evaluating the same 'thing' → repeatability); (3) if you're able to measure something in a proper way, understanding what attribute(s) you're measuring, you can have information and should be sufficiently aware for taking decisions. Just a short example for better expressing the need and value when having (or not) a clear and not ambiguous definition: asking what is a LOC (Line of Code), possible answers could be: (a) a physical statement; (b) a logical statement; and both could be complemented (c) with or (d) without commented lines. Thus, counting LOCs for a

ISBN-14: 978-92-990073-2-7

software system, numbers could vary a lot just applying slightly different definitions[1]. Another short example with Function Points (FP): the IFPUG method till v4.2 formally included the so-called VAF (Value Adjustment Factor), expressing 14 non-functional attributes 'adjusting' the initial functional size value. Thus, AFP (Adjusted FP) formula included also VAF, while UFP (Unadjusted FP) not. But what should it mean the solely FP acronym? Which should be the right number of Function Points to count and declare for such activity? As in Figure 1, since any 'thing' to be evaluated is a mix of quantity and quality and each side has different parameters for being evaluated (in terms of productivity, costs and so on), it's fundamental to deeply analyze the 'quality' side – that has been right now the less explored (also because more complex) part of the 'yin-yang' representation.

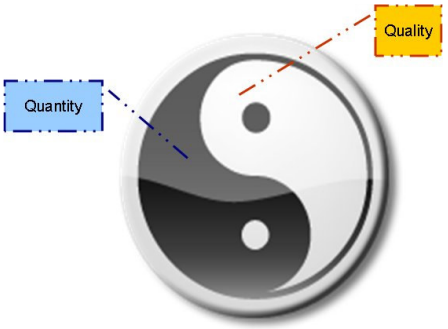

Fig. 1. Quantity and Quality – a 'Yin-Yang' representation[2]

The paper is organized as follows: Section 2 will propose a short history of quality models (QM) from mid '70s on. Section 3 will discuss the stakeholders' issue: the inclusion (or not) for an attribute in a QM could be also due to the viewpoint faced and the stakeholders included (or not) in the analysis. Moving from the historical perspective shown, Section 4 will propose perspectives about how QM are evolving and should still evolve for properly catching the value for software quality during next years.

II. A SHORT HISTORY OF QUALITY MODELS (QM)

Our core question is: what is quality? 'Quality' is a multi-facet term because it's an aggregator for multiple attributes. If you should express why you've appreciated a certain food, you would start to list a series of 'attributes' such as: flavour, taste, way to be presented, freshness of ingredients, the quality/price ratio, etc. Next step would be their quantification, trying to find a shared way to 'count' them. That's the application of the well-known Goal-Question-Metric (GQM) paradigm [20]. The same happened (and still happens) in Software Engineering with Quality Models (QM). If the 'quantity' side expresses the functionalities (what the software product – not the software project! - is asked to do), the 'quality' side should express the non-functionalities (how those functions should work for satisfying its users-clients). Thus a QM can be defined as a shared list of attributes/characteristics that an entity of interest (EoI) can own, expressing its non-functional side ('how'). A QM can be articulated in one or more tiers: in the second case, there will be a hierarchy of attributes with high-level and low-level attributes. For 'completing' a QM, typically a further tier is added with measures that help in quantifying a certain attribute. Now a list of more known QM will be presented, trying to stress their peculiarities for catching useful elements for improving the next generation of QMs.

A. FCM (Factor-Criteria-Model)

This is the first QM, produced in the mid '70s within the Air Navy [1].

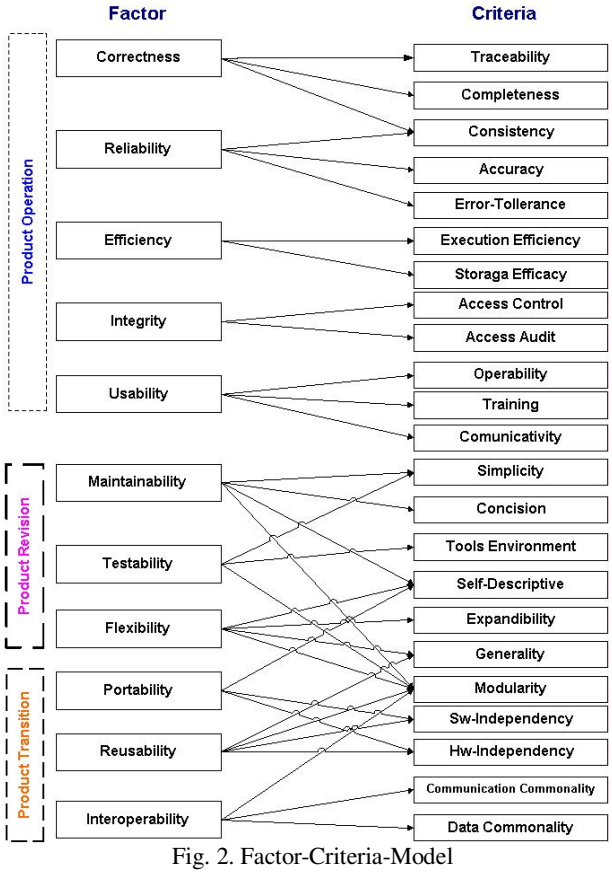

Fig. 2. Factor-Criteria-Model

It contained 11 factors (the first layer-tier) and 23 criteria (the second layer). Each factor was linked to 2+ criteria. Of course, as in any QM, each element needs to have a clear definition with unambiguous statements. Factors

[1] According to Jones [13] , there could be variability till 500% between extremes.

[2] Another way to express the same concept is using a coin: quality and quantity are the two faces of a coin. It's not possible to obtain a comprehensive evaluation not dealing with both faces. But each one has its own properties (attributes) and measures.

were classified into three moments in time along the software life cycle (SLC): product operation, product revision, product transition.

B. Boehm Quality Model.

One year later, Boehm proposed his own QM, with 7 high-level characteristics (1st level) and 12 primitive characteristics (2nd level) [2]. Also here a high-level char could be linked to 2+ primitive characteristics. Introduced the 'utility' concept, splitting the 'as-is utility' and the 'maintainability' for software products.

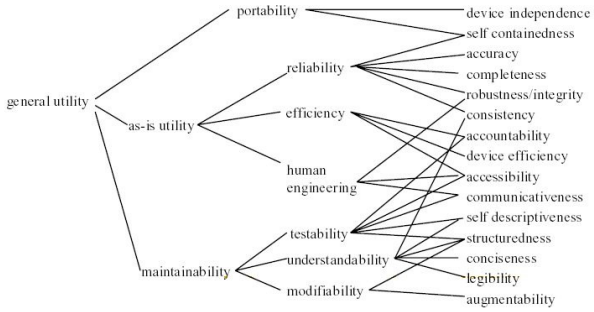

Fig. 3. Boehm's Quality Model

C. ISO 9126:1991

Moving from such early QMs, ISO decided – after the realising of the first 9001 version in 1986 – to release its own QM [3]. The model included 6 characteristics and 18 sub-characteristics. Here each high-level characteristic is subdivided in a more refined list, with no-crossed links.

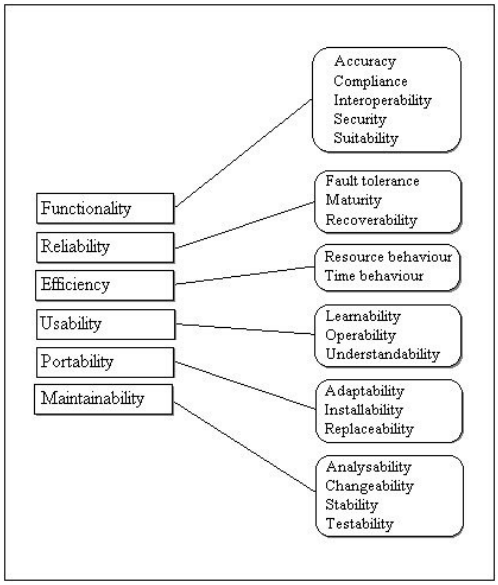

Fig. 4. ISO 9126:1991

IEEE 1061-1992 replied the content of ISO 9126:1991, including such list of 'attributes' in the Appendix A. In 1998, IEEE 1061-1998 deleted such list, considering an open list of values and not a closed list of attributes.

D. ISO 9126-1:2001

After 10 years, ISO refined its view on quality and proposed the new version for the 9126 QM [4]. Introduced the concept of different viewpoints by different stakeholders: internal, external and quality in use viewpoints. Here the first two ones, with 6 characteristics and 26 sub-characteristics. Each low-level characteristic was linked with 1+ process(es) from the ISO/IEC 12207 process model for any related process improvement activity.

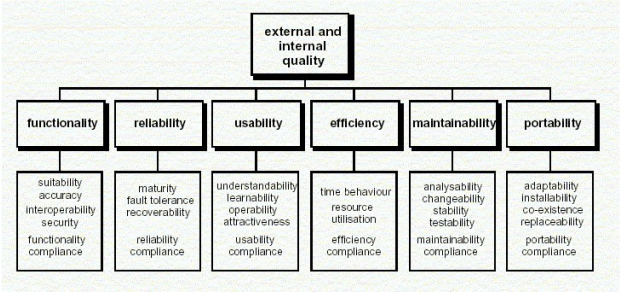

Fig. 5. ISO 9126-1:2001 – External/Internal Quality views

And here the quality in-use view, with the four additional characteristics.

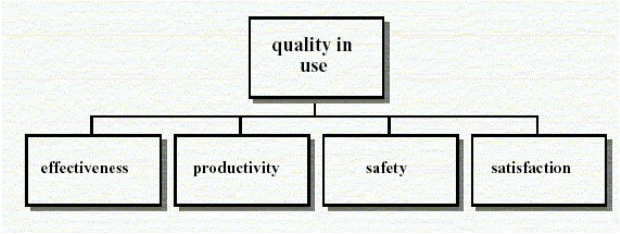

Fig. 6. ISO 9126-1:2001 –Quality in-use view

E. ISO 25010:2011

After 10 year more, ISO revised again its view on quality and evolved 9126 into the SQuaRE (Software product Quality Requirements and Evaluation) 25000 series with the new 2501x block of standards [5].

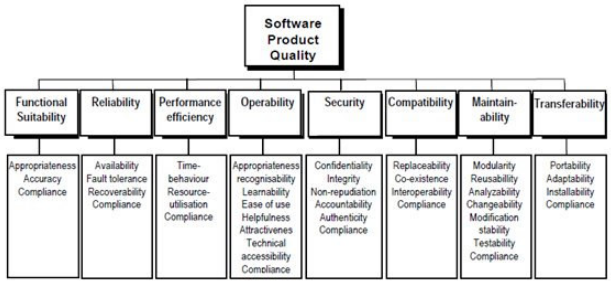

Fig. 7. ISO 25010:2011

ISO 25010:2011 now includes 8 characteristics and 38 sub-characteristics. Refined some characteristics (e.g. Usability evolved into Operability, stressing more the Accessibility issue than before) and introduced others as Security.

F. Other QMs and NFR-related approaches

Other possible QM are:

- FURPS(+): FURPS is the acronym for a software product quality taxonomy – as well as ISO 9126 - by Grady & Caswell [8] and refined with more attributes into FURPS+ [9]. FURPS stands for Functionality (to be split into: Feature Set, Capabilities, Generality, Security), Usability (Human Factors, Aesthetics, Consistency, Documentation), Reliability (Frequency/severity of failure, Recoverability, Predictability, Accuracy, Mean time to failure), Performance (Speed, Efficiency, Resource consumption, Throughput, Response time), Supportability (Testability, Extensibility, Adaptability, Maintainability, Compatibility, Configurability, Serviceability, Installability, Localizability, Portability). The "+" addition represents an aid for remembering concerns such as: Design requirements, Implementation requirements, Interface requirements and Physical requirements.
- ECSS-E-10A + ISO 21351:2005: ECSS (European Cooperation for Space Standardization) is an initiative established to develop a coherent, single set of user-friendly standards for use in all European space activities. Among the several standards produced, 'technical requirements' are diffusely treated. ECSS-E-10A [6] was used for creating ISO 21351:2005 [7].
- IFPUG VAF: from Albrecht's initial study till IFPUG CPM v4.2, the FPA method proposed a 'value adjustment factor' (VAF) based on 14 non-functional attributes (GSC – General System Characteristics), mostly referred to the software product, some others to the software project entity. The 14 GSC are: Data Communication, Distributed Data Processing; Performance; Heavily Used Configuration; Transaction Rate; Online Data Entry; End-User Efficiency; Online Update; Complex Processing; Reusability; Installation Ease; Operational Ease; Multiple sites; Facilitate change. The aim of VAF was to 'adjust' the product functional size by a series of quality attributes for 'optimizing' the statistical relationship in historical series of adjusted product functional size vs project effort. In 1998 ISO decided to keep of such element from any FSM (Functional Size Measurement) method, because not proportional to the product functional side, stating that non-functional requirements (NFR) must be evaluated apart from FUR in a different way. In the current IFPUG CPM v4.3 such list has been maintained in Appendix C [11].
- IFPUG SNAP: more recently, IFPUG proposed a new separate methodology from FPA named SNAP (Software Non-functional Assessment Process). From the analysis of product NFR, the method calculates the number of SNAP Points (SP). The current v2.2 [12] includes 14 sub-categories grouped into 4 categories. As in FPA, each sub-category has 2+ complexity parameters for deriving for each SCU (SNAP Counting Unit) the associated number of SP. Here the list of categories and sub-categories that could be used also as a QM, not considering the SP calculation algorithm: **Data Operations** (Data Entry Validation; Logical & Mathematical Operations; Data Formatting; Internal Data Movements; Delivering Added Value to Users by Data Configuration); **Interface Design** (UI Changes; Help Methods; Multiple Input Methods; Multiple Output Methods); **Technical Environment** (Multiple Platform; Database Technology; Batch Processing System); **Architecture** (Component Based Sw Dev (CBSD); Multiple Input/Output Interface).

III. POSSIBLE CRITERIA FOR A QM

Analyzing the proposed QM it is possible to derive the following considerations in order to understand the value to be provided by a QM:

- Stakeholders – as stressed in well-recognized management guides such as PMBOK [17] or ITIL [18], it is fundamental to understand from the beginning which are the right stakeholders to involve for creating a good QM. For instance, users are fundamental but often have been considered only for providing final feedback (customer/user satisfaction), not for driving assessment criteria. Remember that a customer (the business) is not necessarily the user, but could be separate people. Remember also to involve those secondary stakeholders (e.g. foreign tourists could be useful for describing how to improve a mobile touristic app for a certain city providing a different viewpoint than a citizen from that city).
- Grouping criteria – quality represents the 'how' a product should be realized according to initial requirements. Thus, several criteria should be considered. For instance (a) Time: a lifecycle view should be included and/or linked to a QM (e.g. ISO 9126-1:2001 inserted the related process(es) from ISO 12207 and target audience for any sub-characteristic). It could be useful for improving the product during its lifetime for maintainability purposes. (b) Viewpoint/Stakeholder positioning: internal, external and quality in-use viewpoints, as proposed by ISO from 2001 with 9126-1 and now with the 25010 standards; (c) Viewpoint/Context-Content: the wider the list of attributes and sub-attributes, the more comprehensive the analysis of a product by its QM. As in the Balanced Scorecard (BSC) approach, it'd be desirable to have at least 4-5 perspectives (e.g. time, cost, risk, quality, ethics, etc.) against which grouping quality attributes.

IV. QUALITY MODELS AND THE NEXT DECADE

Looking at the content of the presented QM against the period they were produced, it is possible to list a series of thoughts, possibly useful for designing QM for the next decade:

- Content: a number of product attributes in a QM is useful for better describing and evaluating a product, but as usual – the right number of attributes is in the middle (not too many, not too few). Product observation is fundamental for listing what is needed and it could change along time. For instance, *smartphones* have created a different way to describe and define 'operability' and/or 'usability' against mobile software produced just 3-4 years ago because of the 'touching' interaction on the screen. Again, *sustainability* can be a new product quality attribute to consider for new systems/software [14] because of a 'greener' perspective on software.
- Usage: QM can be used not only for a 'retrospective' evaluation but also in early SLC phases as simple checklists for understanding the level of completeness for a product design, moving from a 'wishing list'. Another way to use QM is for estimation purposes: since QM express NFR, estimators can use needed NFR-related (quality) measures to be included (at least 2+ ones) as independent proxies in estimation models, allowing the reduction of MRE (Mean Relative Error) figures as much as possible, saving project resources and improving the overall project value for its stakeholders (e.g. ISO 9126 parts 2-3-4 define a plenty of measures to be read and applied).
- Perspectives/Viewpoint: a stakeholders' analysis is needed for understanding if the proper number of viewpoints in included (or not) in a QM. If too few perspectives have been included when designing a QM, feedbacks could be lower than expected at the delivery stage. A more comprehensive design can reduce maintenance costs along the product expected lifetime.
- Measurement: last but not least, the measurement issue, that's the lower level in a multi-tier model as a QM is. People less skilled in measurement typically affirm that not anything can be measured. But, as in the introduction, if you are able to describe an entity of interest, you'll be also able to measure it (e.g. using the GQM approach). ISO 15939 [10] refined the GQM paradigm proposing MIM (Measurement Information Model) template that could be a good way to start defining how to monitor & control a non-functional (quality) attribute for a product. The suggestion is to follow a revised version of the well-known 5W+H approach (who, why, what, when, where, how), adding a second 'H' (how much), that could represent 'targets – thresholds' for checking the process-in levels for that measure.

V. CONCLUSIONS AND NEXT STEPS

Quality Models (QM) represent a good way in Software Engineering for evaluating software products from their initial concept till their realization and in-use stage. Non-functional requirements (NFR) are composed from quality and technical requirements; thus quality is one the two sides, maybe the more complex to analyze. Since quality is a multifaceted concept, it's very difficult to find a complete and stable definition for it: quality definition can evolve along time related to newer ways users could request, of course influenced by technology (e.g. smartphones, cloud computing, etc.). QM can help sharing the view on products and be used both in a qualitative (checklists) and quantitative way (measuring low-level attributes with 1+ related measures).

This decade will consolidate some new technology paradigm and will propose new ones: the important thing will be to observe more interesting trends for proposing evolutions and integrations of new, emerging facets for quality more than creating new models at all. Again, even if trivial, we need to clearly define which the entity to be analyzed (product, process, project, organization, resources) in order to avoid effort/cost estimation issues [15]. Evolution, not revolution, can be the right way to understand more about the 'how' realize better software systems.

REFERENCES

[1] McCall J.A., Richards P.K. & Walters G.F., *Factors in Software Quality, Voll. I, II, III: Final Tech. Report*, RADC-TR-77-369, Rome Air Development Center, Air Force System Command, Griffiss Air Force Base, NY, 1977

[2] Boehm B.W., Brown J.R., Kaspar H., Lipow H., MacLeod G.J. & Merritt M., *Characteristics of Software Quality*, Elsevier North-Holland, 1978

[3] ISO/IEC, *IS 9126:1991 - Information Technology - Software product evaluation – Quality* characteristics and guidelines for their use

[4] ISO/IEC, IS 9126-1:2001 - Software engineering -- Product quality -- Part 1: Quality model

[5] ISO/IEC, IS 25010:2011 Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models

[6] ECSS, *Space Engineering – System Engineering: Part 6. Functional and Technical Specifications*, European Cooperation for Space Standardization, ECSS-E-10 Part 6A rev.1, October 31 2005, URL: www.ecss.nl

[7] ISO, 21351:2005 - Space systems -- Functional and technical specifications

[8] Grady R. & Caswell D., Software Metrics: Establishing a Company-Wide Program, Prentice Hall, 1987, ISBN 0138218447.

[9] EELES P., *Capturing Architectural Requirements,* IEEE DeveloperWorks, 2005, URL: www-128.ibm.com/developerworks/rational/library/4706.html

[10] ISO/IEC 15939:2007 - Systems and software engineering

-- Measurement process

[11] IFPUG, *Function Points Counting Practices Manual (release 4.3.1),* International Function Point User Group, Westerville, Ohio, January 2010, URL: www.ifpug.org

[12] IFPUG, SNAP (Software Non-Functional Assessment Process) APM v2.2, June 2014, URL: www.ifpug.org

[13] Jones C., Applied Software Measurement: assuring productivity and quality, 2/e, McGraw-Hill, 1996

[14] Lami G., Buglione L., Measuring Software Sustainability from a Process-Centric Perspective, Proceedings of IWSM-MENSURA 2012, 22th Int.Workshop on Software Measurement and 7th Int. Conference on Software Process and Product Measurement, Assisi (Italy), October 17-19 2012, pp.53-39

[15] Buglione L., Ebert C., Estimation, Encyclopedia of Software Engineering, Taylor & Francis Publisher, June 2012, ISBN: 978-1-4200-5977-9

[16] ISO, IS 21351:2005, *Space Systems – Functional and Technical Specifications*, May 19, 2005, URL: www.iso.ch

[17] PMI, Project Management Body of Knowledge (PMBOK), 5th ed., 2013, www.pmi.org

[18] AXELOS, ITIL v3 – IT Infrastructure Library, Refresh 2011, 2014, www.itil-officialsite.com

[19] Kaplan R., Norton D., The Balanced Scorecard: Translating Strategy Into Action, Harvard Business School Press, 1996, ISBN 0875846513

[20] Basili V.B., Caldiera G. & Rombach H.D., The Goal Question Metric Approach, Encyclopedia of Software Engineering. Wiley 1994, URL: www.cs.umd.edu/projects/SoftEng/ESEG/papers/gqm.pdf