ACTA IMEKO
Month 2015, Vol. x, Number Y, p-p
www.imeko.org

Some Thoughts on Quality Models: Evolution and Perspectives

Luigi Buglione[1]

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

ABSTRACT
'Quality' is an evolving concept, quite difficult to be defined, whatever the application domain observed. As Tom Demarco said, it's true that "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 a common, shared definition is the priority for any activity and creates also measures from any measurement and benchmarking activity. Along the years, several 'quality models' (QM) have been produced: the *scope* for non-functional attributes (part of the 'quality' definition, also according to ISO) is enlarging. This paper discusses the evolution of the quality concept (broader than the one referred to the solely 'product', e.g. service quality) in order to observe which perspectives can be drawn up for the next years.

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

1. 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 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.

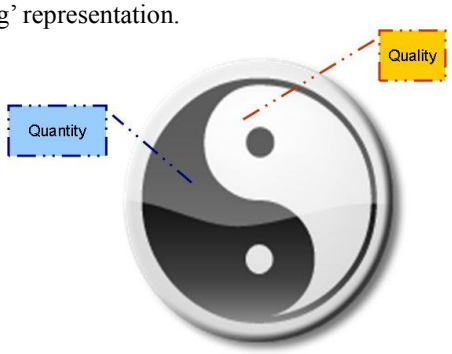

Figure 1. Quantity and Quality – a 'Yin-Yang' representation[2]

[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

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.

2. 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 (<u>what</u> the software product – not the software project! - is asked to do), the 'quality' side should express the non-functionalities (<u>how</u> 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. Moving also from other similar studies [24][25][26][27][28], 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]. It contained 11 factors (the first layer-tier) and 23 criteria (the second layer). Each factor was linked to two or more criteria. Of course, as in any QM, each element needs to have a clear definition with unambiguous statements. Factors were classified into three moments in time along the software life cycle (SLC): product operation, product revision, product transition.

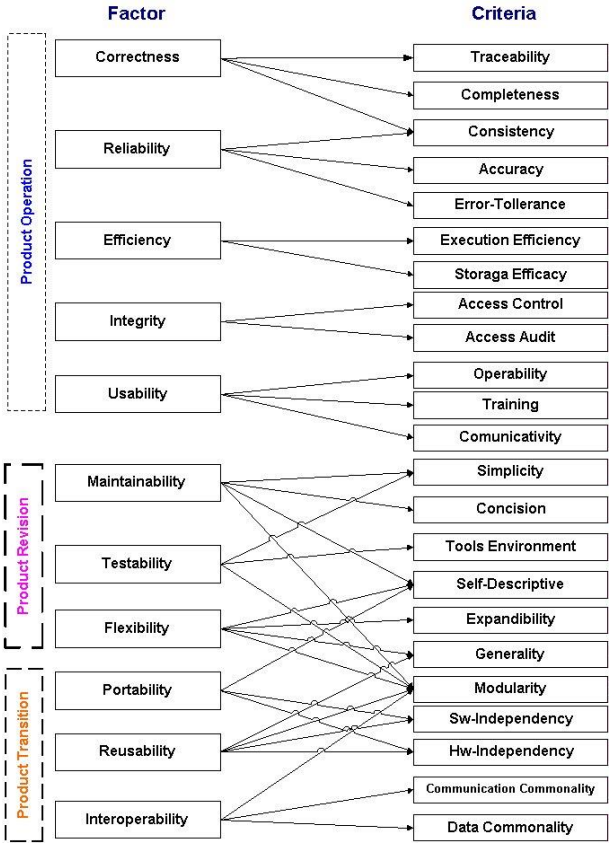

Figure 2. Factor-Criteria-Model

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.

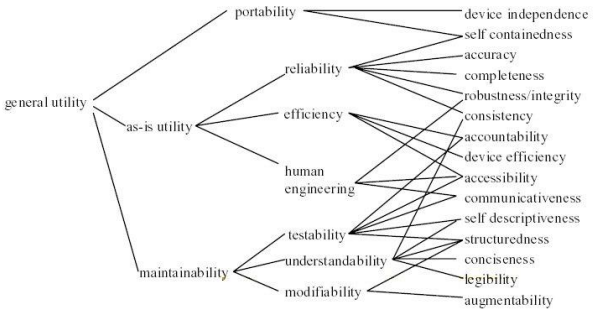

Figure 3. Boehm's Quality Model

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.

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.

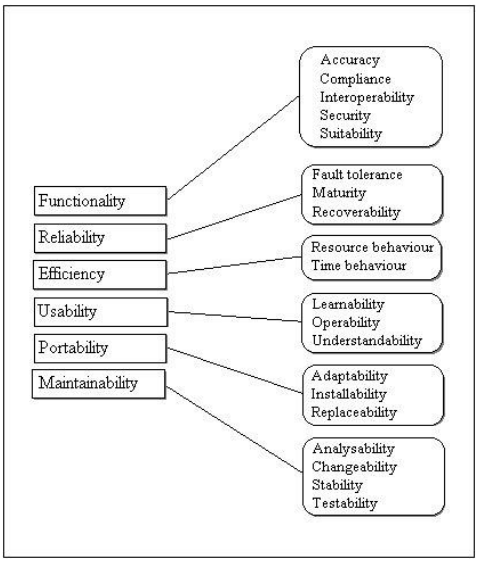

Figure 4. ISO 9126:1991

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.

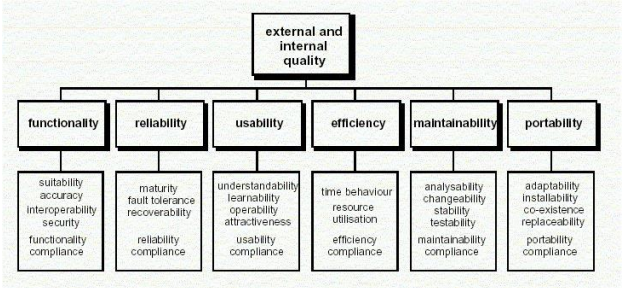

Figure 5. ISO 9126-1:2001 – External/Internal Quality views

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

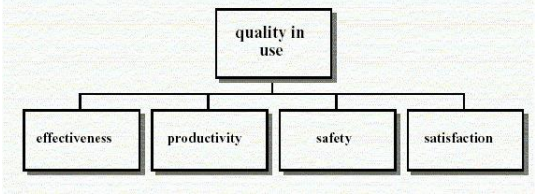

Figure 6. ISO 9126-1:2001 –Quality in-use view

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].

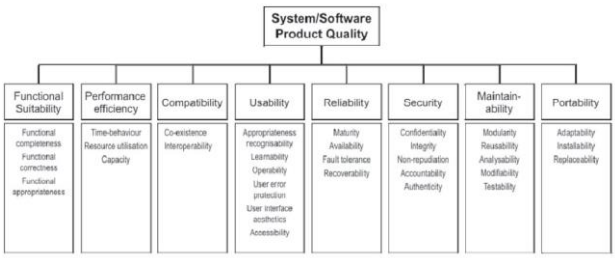

Figure 7. ISO 25010:2011 – Quality Model

ISO 25010:2011 now includes 8 characteristics and 38 sub-characteristics. Refined some characteristics (e.g. Usability is stressing more the Accessibility issue than before) and introduced others as Security. Also the quality in-use part evolved, including now five characteristics, maintaining from the previous version effectiveness and satisfaction and adding efficiency, freedom from risk and context coverage, defining also a lower-level.

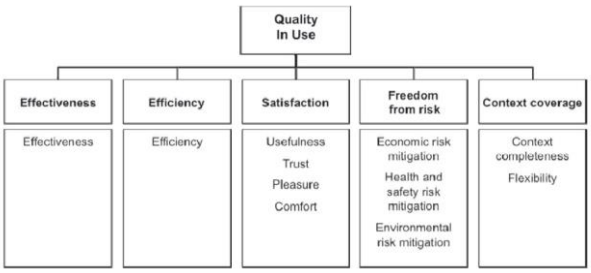

Figure 8. ISO 25010:2011 – Quality in-use

F. Other QMs and NFR-related approaches

After observing the evolution of the ISO view on QMs, this section will briefly introduce and list other possible QM produced in the Software Engineering arena from the '90s on:

- 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 [29], passing for a single revision in 1983 [30] 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).

Table 1 summarizes the different QMs and approaches discusses above.

Name	Year	Author(s)	No.Chars/Sub-Chars	Notes
FCM	1977	McCall et al.	11 Factors (3 groups), 23 criteria	N:M relations
Boehm QM	1978	Boehm	3 main chars, 6 second-lev chars, 15 third-lev chars	---
VAF	1979	Albrecht	10 GSCs (General Systems Chars)	Adj Factor to FPA
VAF	1983	Albrecht-Gaffney	14 GSCs	Adj Factor to FPA
FURPS(+)	1987/92	Grady-Caswell	5 chars, 28 sub-chars	---
ISO 9126	1991	ISO	6 chars, 21 sub-chars	---
ISO 9126-1	2001	ISO	6 chars, 27 sub-chars + 4 quality-in-use chars	2 models (int-ext quality; quality in use)
ISO 21351	2005	ISO	16	Funct. And Tech Req.
ISO 25010	2011	ISO	8 chars, 31 sub-chars + 5 quality-in-use chars and 11 sub-chars	2 models (int-ext quality; quality in use)
SNAP	2011	IFPUG	14	Method

Table 1 – List of main QM – a short summary

3. POSSIBLE CRITERIA FOR EVALUATING 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. It could be applied at the beginning of a project, for planning the expected quality level and/or at the end of a project, for controlling the actual quality levels; (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 would be desirable to have at least 4-5 perspectives (e.g. time, cost, risk, quality, ethics, etc.) against which grouping quality attributes; (d) Measurable entity: a QM should take into account one, single entity of interest (EoI). For instance, QM presented before are about the software product.

4. EAM: TAXONOMIZE A QM

The presented QM can have structurally a two (or three)-level structure, but they are all about (software) products. What about a QM for a different entity? One of the main problems in benchmarking activities is that often comparisons are done between not homogeneous entities and attributes. A simple example from the software industry can be the so-called 'backfiring', that's a linear conversion between LOCs and Function Points (FP). But a Line of Code (LOC) expresses only the length of the source code for a software product: two programmers could generate the same amount of functionalities from the user viewpoint but producing different numbers of LOCs, according to their experience, programming style, if the LOC definition for that organization includes (or not) commented and blank lines etc. FP born in the late '70s just to overcome such contractual problem, willing to express a neutral amount of functionalities for a software system provided by a provider to a customer.

But also in this case there is often a confusion for many ICT professionals about what a FP is sizing: a FP – whatever the FPA variant - expresses the functional size for a software product, thus not the size for a whole software 'project'. Simple demonstration: the so-called BFC (Base Functional Components) in a FSM (Functional Size Measurement) Method are *all* about the 'product', not the 'project' managing it. Again, if a project activity included into a Gantt chart does not vary (directly or indirectly) the number of FP, it means that such activity cannot be seen as related to a FUR (Functional User Requirement), that's the needed input to calculate FP.

EAM (Entity-Attribute-Measure) [15] is an effective way to taxonomize measures trying to depict a three-layered table for allowing understanding if your own set of entities of interest (EoI) has been properly analyzed through a palette of quality attributes. Table 2 presents the previous example using LOC and FP:

E – Entity	(software) **product**	(software) **product**
A – Attribute	*Code Length*	*Functionality*
M - Measure	**Lines of Code (LOC)**	**Function Point (FP)**

Table 2 – Entity (E), Attributes (A), Measures (M) [15]

It easily explains why LOC and FP cannot be 'backfired' or that McCabe's cyclomatic complexity index v(G) - expressing the complexity for a chunk of software – is something different from the 'functionality'. Thus, a more complex software cannot be measured and sized by FP but by v(G) or other similar measures, and so on. EAM could be a complement way to run a GQM (Goal-Question-Metric) [20] and BPM [22] analysis in order to check if we are measuring the 'right things' and 'things right', also because one of the main reasons why organizations measure a few is their perception of a higher cost than expected within the project budget. But often the risks are to measure too much (over-measuring), too few (under-measuring) or the wrong phenomenon (bad-measuring) [22]. EAM would like to be a way to do that.

Looking at the 'project' entity, Fig. 9 shows proposal by PMI (www.pmi.org) for a list of 14 attributes (or characteristics) for a generic 'project'[21], but it's not shared e.g. with the other worldwide project management association such as IPMA (www.ipma.ch) or APMG (www.apmg-international.com).

Again, looking to processes, it's sufficient to take a look to the different attributes proposed in the CMMI constellations (GP - generic practices) and in SPICE (ISO/IEC 15504, the so-called PA - process attributes) for evaluating and rating a process using the N/P/L/F (Not/Partially/Largely/Fully achieved) ordinal rating scale.

Thus, the match between a typical QM architecture and EAM can help an organization to clearly link operational measures in a project with the related entities and attributes (eventually two-layered) for the adopted QM, returning which could be the (sub)characteristic(s) with missing data/information to work on.

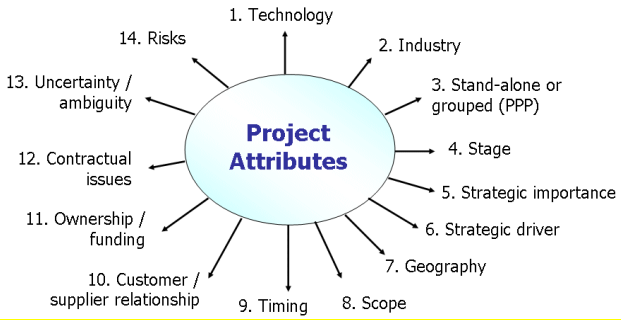

Figure 9. Project Attributes [21]

5. BMP: BALANCING MULTIPLE PERSPECTIVES IN QM

After understanding the quality level adopting the current version of a QM, next step could be to check and improve it over time. As in life, one fundamental criterion for being successful is to properly balance elements, possibly finding the most correct alchemy. It is not a simple task, but it implies a deeper knowledge about the way an organization works and the leverages to be used for achieving better results. BMP (Balancing Multiple Perspectives) [22] is a technique helping organizations in analyzing whether they are properly balanced against a set of criteria or not: are you managing your projects only by Time and Cost measures or including also Quality, Risk and/or Ethics?

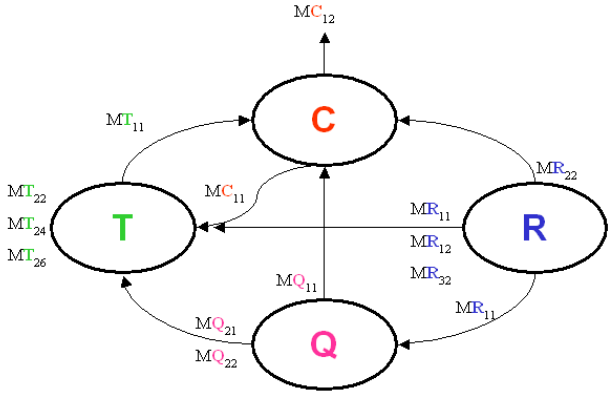

Figure 10. BMP approach [22] – Perspectives and Measures

Each 'perspective' (in this example: Time, Cost, Quality, Risk) could represent the trigger for a specific QM. And each 'specific' QM could be deployed till the third level adopting the EAM approach presented before (measures – here shown as 'M' followed by the initial of the perspective (T, C, Q, R) and an ordinal number), in order to check the level of completeness for each 'leaf' of this tree. Such approach, as described in [22], would help to drive an organization in obtaining more informative value from the same set of measures yet adopted, analyzing them in groups of two, or three or four, using a RCA (Root-Cause Analysis) approach.

An example: the number of software defects before the project will be delivered or a certain number of LOC (or FP) could be good or bad, according to the thresholds established for each single measure. But correlating them – without gathering necessarily new measures, thus adding costs to the project – could give the project back additional information, e.g. measuring the 'defect density' for that chunk of software. And so on. Thus, in this context BMP perspectives could be interpreted as:

- At a higher level, as Entity of Interest (EoI) – in this case the 'bubbles' could be represented by e.g. the five entities described in the STAR taxonomy (organization, project, resources, process, product) [23]
- At a lower level, as QM attributes – in this case the balancing would be, within an EoI, among the different attributes (and related sub-attributes) till the measure layer.

Fig. 11 proposes a graphical view of what discussed, merging the typical QM structure with EAM, the two possible application of the BMP technique. The Project Historical Database (PHD) represents the knowledge base where all the information should be stored for being shared along the organization.

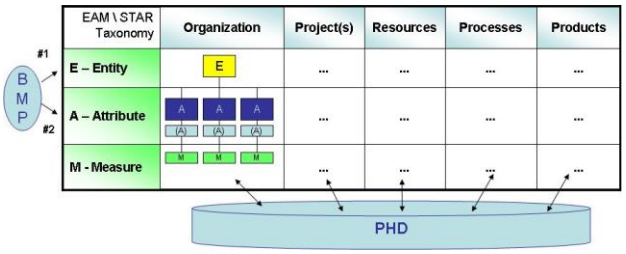

Figure 11. Merging EAM, BMP, STAR taxonomy and QM

For instance, ITIL proposes a 'rule of thumb' when dealing with how many KPI per each goal to formalize and use, proposing 2-3 KPI per goal and 2-3 base measures per each KPI. Measurement should be more and more an opportunity more than a cost: thus, organizations and projects could start to calculate also a ROM (Return On Measurement) as a variant of the traditional ROI, considering also the benefits and not only the cost from a proper measurement & analysis (MA) process, as stated e.g. from CMMI or ISO 9001 as a process to be deployed as soon as possible.

6. A SHORT EXAMPLE

In order to show an application of the concepts above presented, a short example follows, presenting before an EAM table summarizing the involved elements and later a GQM-like tree, depicting three levels (goal, KPI, measure).

Such measurement plan has been produced taking into account four measurable entities (project, resources, process, product), planning one (or more) KPIs per each measurable entity and two (or more) measures per each KPI:

Entity	Project	Resources	Process
Attribute	*Estimation Capability*	*Teaming Capability*	*Req. Elicitation Capability*
Measure	**M.01~ M.04**	**M.05~ M.08**	**M.09~ M.11**

Entity	Process	Product	Product
Attribute	*Req. Elicitation Capability*	*Defectability*	*Req. Granularity*
Measure	**M.09~ M.11**	**M.14~ M.16**	**M.17~ M.18**

Table 3 – EAM table s (split in two blocks)

And now the GQM-like tree for the four (4) goals:

- **G1. Improve projects' estimation capability (→ Entity: Project)**
  - *KPI.01 – Improve the 'historicization' of project data*
    - M.01 – No. of total projects managed per quarter by Business Unit
    - M.02 – No. of projects uploading 'project closing' data per quarter
    - M.03 – Avg/Median estimated % effort for (PMBOK) 'Project Closure' phase
    - M.04 - Avg/Median actual % effort for (PMBOK) 'Project Closure' phase
- **G2. Improve the Requirement Elicitation process (→ Entity: Resources)**
  - *KPI.02 – Optimize the project teaming process*
    - M.05 – Avg/median capability level per project role (periodically monitored)
    - M.06 –Avg/median seniority ratio per project role by that functional domain
    - M.07 - Avg/median cost per project role (by m/d)
    - M.08 – Avg/median No. of changes in the team (per project)
- **G3. Improve the Requirement Elicitation process (→ Entity: Process)**
  - *KPI.03 – Minimize the number of implicit requirements*
    - M.09 – Analysts avg/median capability level (periodically monitored)
    - M.10 – Analysts avg/median seniority ratio by that functional domain
    - M.11 - % of scope creep by m/d (economical)
  - *KPI.04 – Optimize the FP/SP counting capability*
    - M.12 – Avg/median % of scope creep per project by a certified FSM-people
    - M.13 - Avg/median % of scope creep per project by a certified NFSM-people
- **G4. Improve Software Quality (→ Entity: Product)**
  - *KPI.05 - Reduce the number of Defects per delivery*
    - M.14 - No. of defects FUR-related
    - M.15 - No. of defects NFR- related
    - M.16 - No. of defects Project-related
  - *KPI.06 - Reduce the number of not granular (product) URs*
    - M.17 – No. of (product) FURs not expressed at the 'Elementary Process' level
    - M.18 – No. of (product) NFRs not expressed at the 'Elementary Process' level

Gathering periodically those measures allows a project manager to better control a project with a close balanced monitoring by four EoI, not only one and not only looking at time and costs. In that way it can be easier eventually to detect where root-causes can be placed for being removed quickly, lowering the Total Cost of Ownership (TCO) for that project (not necessarily once at a time, but 2 or more concurrently contributing to a certain final effect). A high number of defects could be due to a real low quality of that software, but also to a low coverage of test cases vs user requirements and that low coverage could be due (second step back in this example root-cause analysis) to a reduced budget for the Analysis+Testing phases or because of unexperienced Test Specialists, not including the right number of tests or – if the coverage level would be ok - the proper test cases for verifying such part of the software. Thus, a balanced measurement plan can easily drive to better solutions, not necessarily spending more than in the 'traditional way'.

7. 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 in order to 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: another aspect to consider is 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.

- Entity of Interest (EoI): even if such QM born in the Software Engineering arena, they could be redesign or sometimes simply applied to a different EoI. Looking at the ISO 25010 quality model, it can be applied with very small changes for evaluating a service, not necessarily an ICT one. Imagine needing to evaluate a service desk service, as defined by ITIL or another IT Service Management (ITSM) best practice. You should take into account its reliability, continuity, security or 'reusability' of some components of a service. Of course some definitions could need to be slightly modified, but also the 'quality in use' part could be applied as is. Sometimes we act during time consolidating habits and concepts, but maintenance is a service, thus ITIL, CMM-SVC or other ITSM guides could be applied not only looking to Business Continuity service or something similar, but also for planning and managing a software maintenance service. By the way, some organizations can apply CMMI-DEV practices and processes also for maintenance projects, but CMMI-SVC would have a better fit.

8. 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 one or more 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 and resources) in order to avoid effort/cost estimation issues [15]. But – as said before – such QMs can be applied to any EoI, thus also to services, processes and projects. Evolution, not revolution, can be the right way to understand more about the 'how' realize better software systems and ICT services.

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: http://goo.gl/7PtM2z

[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, Encyclopaedia 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: http://goo.gl/uU5jJC

[21] Turner R.J., Huemann M., Anbari F.T., Bredillet C.N., Perspectives on Projects, Routledge, 2010, ISBN 978-0-415-99374-6

[22] Buglione L. & Abran A, *Multidimensional Project Management Tracking & Control - Related Measurement Issues*, Proceedings of SMEF 2005, Software

Measurement European Forum, 16-18 March 2005, Rome (Italy), pp. 205-214, URL: http://goo.gl/2dRL5j

[23] Buglione L. & Abran A., *ICEBERG: a different look at Software Project Management*, IWSM2002 in "Software Measurement and Estimation", Proceedings of the 12th International Workshop on Software Measurement (IWSM2002), October 7-9, 2002, Magdeburg (Germany), Shaker Verlag, ISBN 3-8322-0765-1, pp. 153-167, URL: http://goo.gl/t5Y0Hl

[24] Jamwal R.S., Jamwal D., PadhaD., Comparative Analysis of Different Software Quality Models, Proceedings of the 3rd National Conference; INDIACom-2009, Computing For Nation Development, February 26 – 27, 2009, URL: http://goo.gl/kDvkYl

[25] Al-Badareen A.B., Selamat M.H., Jabar M.A., Din J., Turaev S., Software Quality Models: A Comparative Study, in Software Engineering and Computer Systems, Part I: Second International Conference, ICSECS 2011, Kuantan, Malaysia, June 27-29, 2011. Proceedings, pp. 46-55

[26] Suman, Wadhwa M., A Comparative Study of Software Quality Models, International Journal of Computer Science and Information Technologies, Vol. 5 (4) , 2014, pp.5634-5638, URL: http://goo.gl/rTSyQe

[27] Sharma K., Comparison Of Various Software Quality Models, Proc. of the Intl. Conf. on Recent Trends In Computing and Communication Engineering -- RTCCE 2013, pp.48-51

[28] Sanjay Kumar Dubey, Ghosh S., Rana A., Comparison of Software Quality Models: An Analytical Approach, Int. Journal of Emerging Technology and Advanced Engineering, Vol. 2, No.2, Feb 2012

[29] Albrecht A., Measuring Application Development Productivity, Proceedings of the IBM Applications Development Symp., Monterey, CA (USA), Oct.14-17, 1979, URL: http://goo.gl/f0RN26

[30] Albrecht A. & Gaffney J.E., Software Functions, Source Lines of Code, and Development Effort Prediction: A Software Science Validation, IEEE Transactions on Software Engineering, vol. 9, no. 6, Nov. 1983; URL: http://goo.gl/qYikl9