Activity Report: PhD 2nd Year

Yann Argotti

▶ **To cite this version:**

Yann Argotti. Activity Report: PhD 2nd Year: Study of Qualimetry essentials applied to embedded software product and organization, with consideration to software entropy. Activity report linked to contract n°179808; Rapport LAAS n° 20045. 2020. hal-02491526

HAL Id: hal-02491526
https://laas.hal.science/hal-02491526

Submitted on 26 Feb 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

RISC / ISI Group

Activity Report: PhD 2nd Year

Study of Qualimetry essentials applied to embedded software product and organization, with consideration to software entropy

ARGOTTI Yann

2-24-2020

Table of Contents

Confidential C

1- Introduction

The purpose of this report is to summarize the second year of research activity related to Yann Argotti's PhD inside the RISC / ISI group at LAAS-CNRS. This PhD candidate is registered at EDSYS school and INSA Toulouse in "*Computer science and embedded systems*" specialization track. Claude Baron, RISC / ISI group responsible and Philippe Esteban, member of RISC / ISI group, are co-joint thesis directors. Thesis subject is "*Study of Qualimetry essentials applied to embedded software product and organization, with consideration*"

This second year was the continuity of the first year during which an analysis of the thesis problematic was performed, identifying several road blockers that we started to address mainly from a theoretical point of view. So, this year we began to apply our contributions to qualimetry theory against automotive software development, demonstrating their applicability, interests, and benefits. Moreover, we synthetized the thesis progress over the production of two research papers, one submitted to IEEE Transaction of Software Engineering journal [1] which was unfortunately rejected, but with some constructive feedback, and one to ERTS 2020 conference [2] which was accepted and then presented on end of January 2020 (see Annex for a copy of both papers).

In parallel, we initiated a new systematic literature review on quality models related to software product, process and project. The goal of this work is to perform a study not only to build a unique and precise landscape of quality models in this field, but also to create the necessary quality model substrate to decide which model(s) we must be using in order to complete the construction of our solution.

Thus, and because this research work is incremental and multi-annual, the next parts of this report are a complement to the 1st year activity report and cover an update of goals we are targeting for this PhD work, an update of our research work organization and a review of our current achievements before concluding.

2- Updated goals

Since Qualimetry is by definition the quantification of Quality for any object, including process, and considering the technological barriers seen in previous section, the inceptive thesis main goal is to:

Main Goal

Define and evaluate an optimized Quality Model in order to bridge and quantify quality not only for software development (ie requirements, models, source code) but also for software organization, with conformance[1] to ASPICE MAN.6 [3]**, ISO/IEC 25010** [4]**, ISO 26262** [5] **and ISO/TS 16949:2009 [36]**.

Secondary initial thesis goals are to exercise this quality model and achieve:

Secondary Goals

- Software Maturity within Continuous Integration process measurement
  - Traceability / Quality by design / Change impact
  - Test efficiency
  - Project Landing zone (Time to market vs risk & complexity)
- Software Aging measurement
  - Software Model Aging (e.g. Impact from Maintenance, FOTA and/or Complex system vs impact to safety & reliability)
  - Product Aging Landing zone

[1] The company, welcoming student here, is in automotive field which therefore drives standard choices here.

Confidential C

However, on first year and then on second year, our further study and analysis performed on Qualimetry and quality models required us to refactor slightly and consolidate these originals goals. Indeed, we were able not only to confirm our early statement that there is no obvious solution for quality model for software development and for software organization, but also that there is a misunderstanding about Qualimetry[2] [7] and we identified gaps on quality model classification and on decision for the quality model solution to apply/use. Comparison studies we can found in [8], [9] or [10] illustrate clearly that fact. Only some work done by Oriol *et al.* [11] including ontology consideration and 51 quality model references open the perspective of a solution. So, to be able to address our main initial goal, we can decompose it into:

Main Goal: sub-

1- **Build a precise landscape for quality model for software**
   a. Build a taxonomy for quality model,
   b. Perform a systematic literature review of quality models for software product, process & project,
   c. Reference and classify found quality models based on our taxonomy,
2- **Create a SW product quality model genome**
   a. Build a methodology to create a quality model genome,
   b. Build a first candidate list of SW product quality models to be used as a 1st quality model basis as genome creation,
   c. Identify SW product quality model main genes
3- **Define an oracle for decision to get optimum quality model(s) solution,**
   a. Identify key criteria for oracle definition (e.g. Non-Parametric Linkage usage against genome),
   b. Build a decision oracle for quality model,
   c. Apply to our case study that oracle to *Renault Software Labs* use case to generate a basis quality model to be used with polymorphism concept,

To help and support these tasks, we are adding a fourth sub-goal that must be achieved in parallel to these ones: this is the creation a unique tool and quality model database. Moreover, that tool will demonstrate that our proposal is viable and usable.

Main Goal: sub-

4- **Define and implement a tool to support quality model taxonomy & oracle**
   a. Describe Quality model seen in state of the art (use Yaml for quality model description, pedigree, setup bridge/dictionary of "synonym" =>http://www.atlas-semantiques.eu/?l=EN),
   b. Apply the computation over the multiple quality model combination,
   c. Quality model decision helper (oracle?),

Thanks to this goal refinement, we can rewrite the first initial secondary goal relying on the taxonomy and oracle studied and developed in our main goal. In addition, we understand that the amount of work required to achieve it may bring risk to not be able to complete on time initial secondary goals. Therefore, the rewritten secondary goals are defined below:

Rewritten Secondary Goals

- **Quality Models and Metrics linked to CI**
  a. Derived oracle identified quality model to ECU covered by CI,
  b. Apply these derived quality models to CI applicable characteristics and metrics,
- **Software Aging measurement**
  a. Summarize problematic and difference against regular (ie often associated to reliability) vs our definition of Software Model Aging

[2] In general, qualimetry is understood as "applied qualimetry" and not really as qualimetry as a science.

Confidential C

b. Takes examples: Impact from Maintenance, FOTA and/or Complex system vs impact to safety & reliability, product aging landing zone

3- Updated organization of research work

Research work is organized into several incremental steps, organized around the goals and sub-goals defined and refined in previous section.

Indeed, the first step was to proceed on study and analyze the current concepts and problematics behind the thesis subject. This study was based on a rigorous state of the art, started from the early listed references, on qualimetry, quality quantification, quality models and our fields of interested which are embedded software product and its organization. In addition, and as written in previous section, our analysis concluded on the need to build a precise quality model landscape using a specific taxonomy for quality model and then followed by a need to construct a methodology on decision / oracle to generate optimum quality model solution.

This global approach is acting on the theory field. However, we will need to exercise regularly our findings against the practical field, applying on some specific case studies, such as software product quality models and continuous integration process. These experiments phases will allow us to loop back to correct, consolidate, optimize our taxonomy, decision methodology and then quality model landscape. We are planning multiple back and forth loops between theory and practice field to be able to converge on a suitable and practicable solution. To support also these experiments, we are planning to implement a tool.

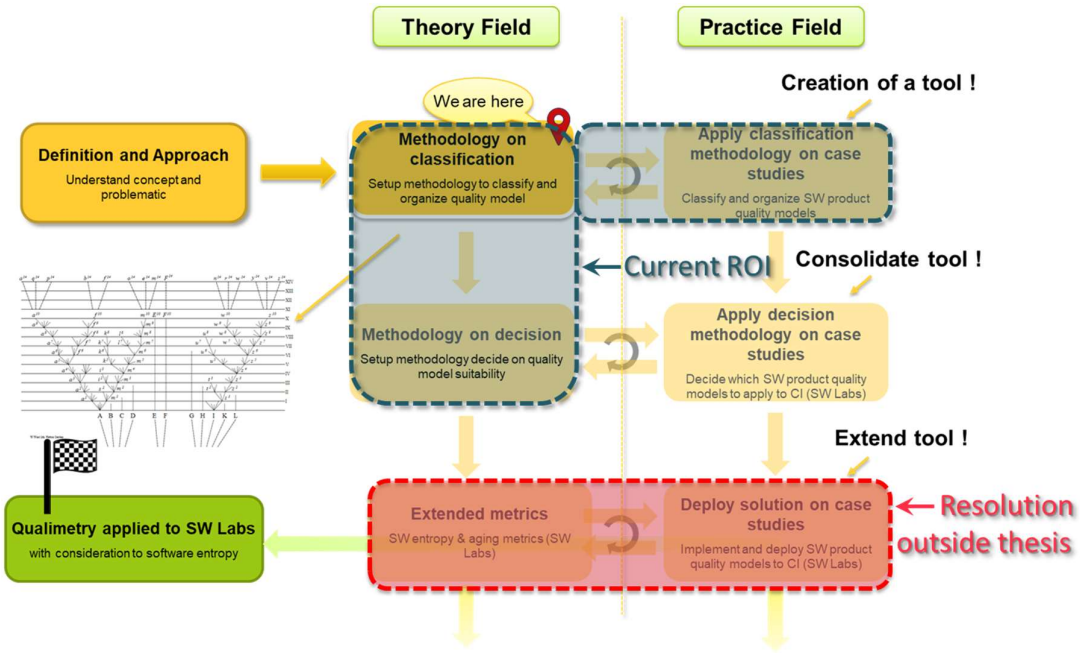

Figure 1 - Current research and development flow

The research work organization is summarized in *Figure 1*. We see the various task sequence, the theory vs practice aspects, and their loopback and the tool supporting our overall approach. We can notice that we have completed first task and are currently focusing in parallel on quality model taxonomy (or classification methodology) and decision to reflect work on quality model landscape. Also, we note that due to our current workload, we have to consider delaying the extend metrics study and realization.

Confidential C

4- Current achievements

As indicated in section 1-, our first-year contributions were done mainly from a theorical point of view to qualimetry. So, our first achievements on 2019 were the application of our theory contributions, and more particularly polymorphism ones, against one of our current use cases: automotive embedded software.

$$\pi = \sum_{ij} x_i x_j \pi_{ij} \qquad \textbf{(1)}$$

Our first achievement is the successful application of the degree of polymorphism formula **(1)**, introduced by Nei and Li in 1979 [12] in genetic domain, against two well-known quality model standards which are also successively referenced in the successive versions of Automotive SPICE [3]: ISO/IEC 9126 [13] and ISO/IEC 25010 [4]. Even if ISO/IEC 25010 is replacing officially ISO/IEC 9126 as standard software product quality model, the degree of polymorphism is indicating that the two quality models are disjoined by ~68%[3] which is not minor differences between the two models. Figure 2 is illustrating some of the differences between these two quality models.

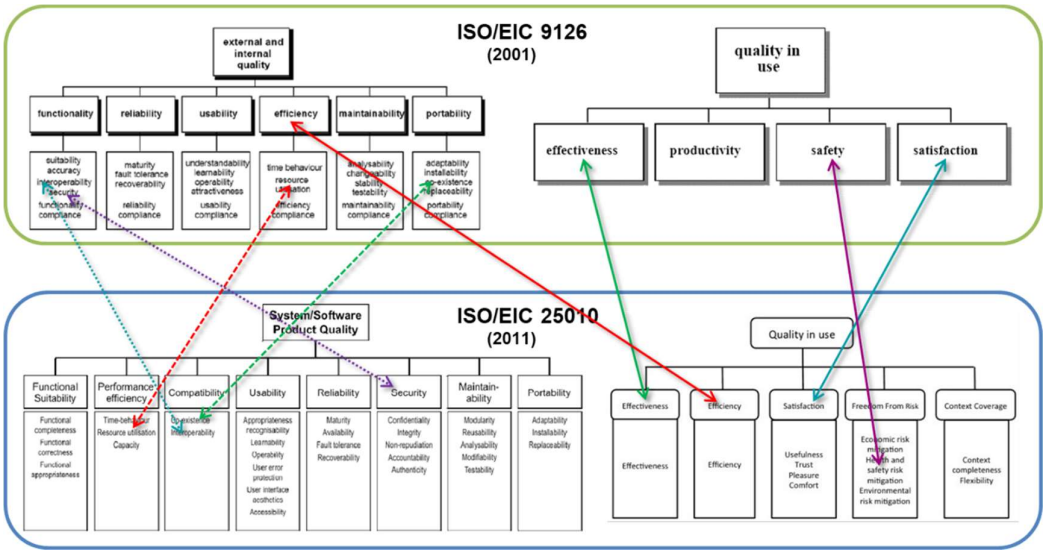

Figure 2 - Few highlighted differences between ISO/IEC 9126 and ISO/IEC 25010

An additional achievement linked to that distance formula is our exploration with regards to the impact and benefits of quality model distance. The needs to use or refer to that distance between quality models are associated to a need to evaluate, modify, change or update quality model due to a variety of root causes such as: change of life cycle stage (e.g. from design to implementation), evolution of product (e.g. addition of new features), insufficient quality area coverage (e.g. gaps in safety or security), change of targeted product (e.g. from car to truck), new or updated process or standard (e.g. from ISO/IEC 9126 to ISO/IEC 25010), …. The main benefits are enumerated here:

a. Evaluate risk linked to quality model change (low distance = low risk, high distance = high risk),
b. Evaluate change workload and cost,
c. Identify most impacted areas and characteristics,
d. Identify where quality quantification, assessment and control are changing,
e. Identify and evaluate validation path finding change (Capture of different types of bugs possibly never found before, Discard other areas and path)

[3] **Degree of polymorphism = 0.6792** (0 = identical; 1 = 100% disjoined): 53 leaf characteristics, 32 unique, 8 similar

Confidential C

f. Support decision and control change / update of quality model

Our next achievement this year concerns the application of *ad-hoc*[4] and *universal*[5] polymorphism concept to Automotive quality model product: starting from ISO/IEC 25010, we defined a generic quality model common to all vehicle Electronic Control Unit (ECU), and then deriving it into In-Vehicle Infotainment (IVI) and Body Control Module (BCM) ECU quality models (see Figure 3).

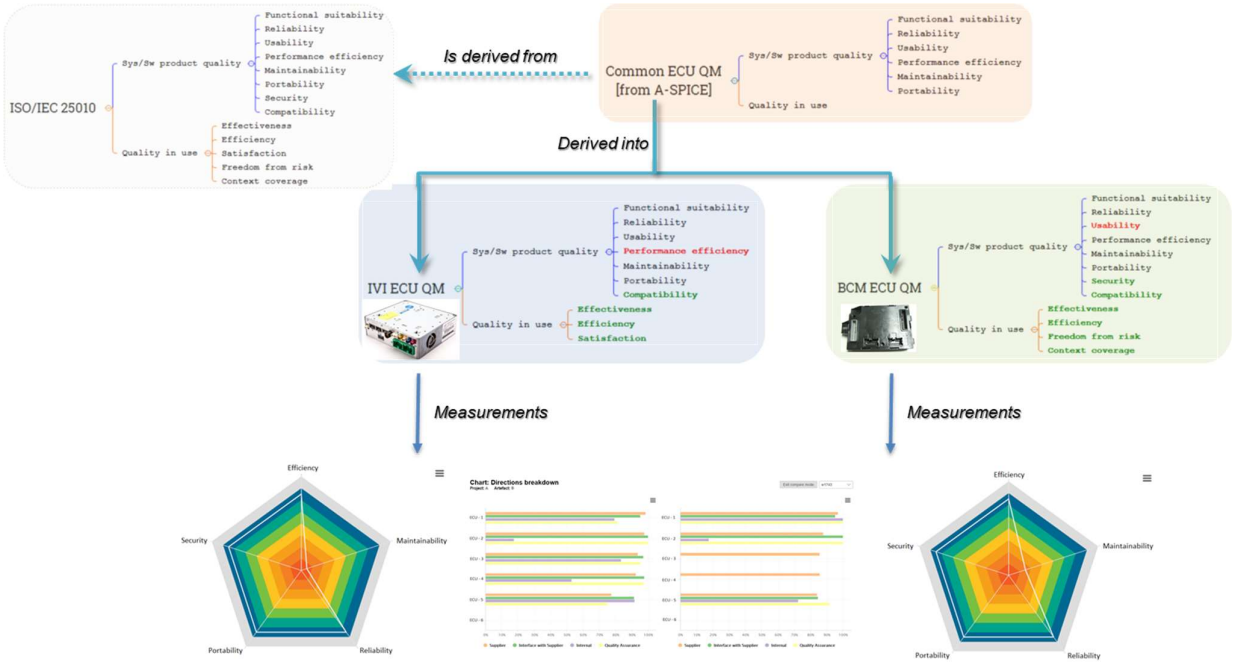

Figure 3 – Ad-hoc & universal polymorphism applied to Automotive

Another application aspect to which we contributed was the design and internal publication of a project scorecard for Renault and Renault Software Labs. This scorecard, showed in Figure 4, is part of Alliance (ie Renault – Nissan – Mitsubishi) SoftWare Process (ASWP). It integrates usage of software product, process and project metrics, includes all metric definitions, "risk vs opportunity" based thresholds and a mapping between each metrics with corresponding ASWP process. Its junction with a quality model is realized thanks to the software product quality indicator which aggregate software metrics accordingly to that quality model. Completing that scorecard, we started the implementation of automatic metrics collection and corresponding graphic generation (ie pie charts, cumulative flow charts and trend charts – see Figure 5) in python language with *yaml*, *jira*, *matplotlib*, *numpy* and *pandas* library.

[4] Common quality model characteristics or "interface" (*ie* ***Ad hoc polymorphism:*** *overloading & coercion*)

[5] Variations with heritage between quality models (ie ***Universal polymorphism*****:** *sub-classing, inheritance, or overriding, extension*)

Confidential C

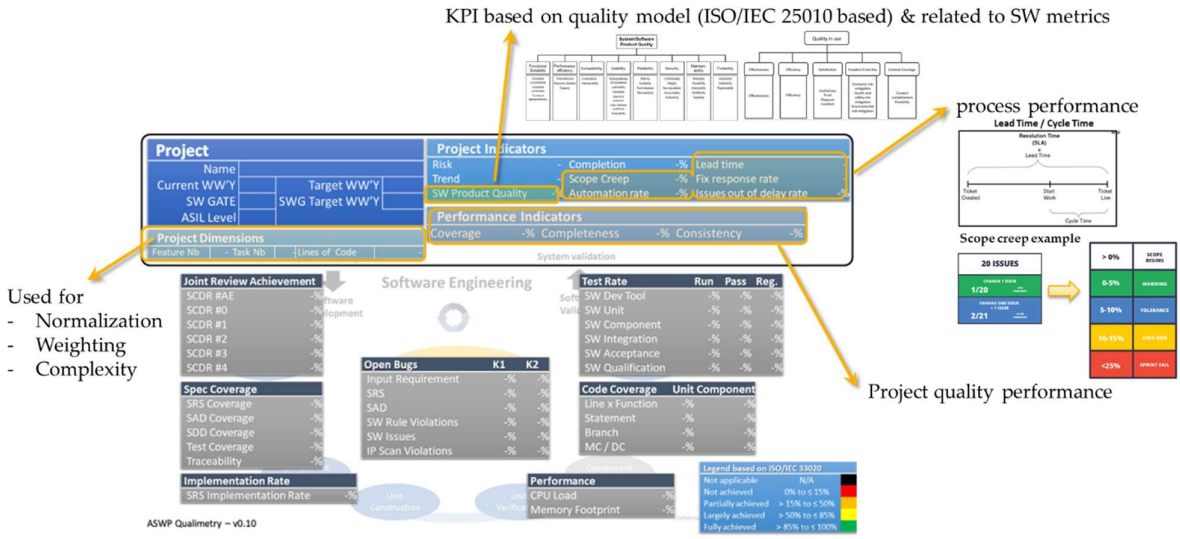

Figure 4 - Our project scorecard linked to quality model, process, product and project metrics

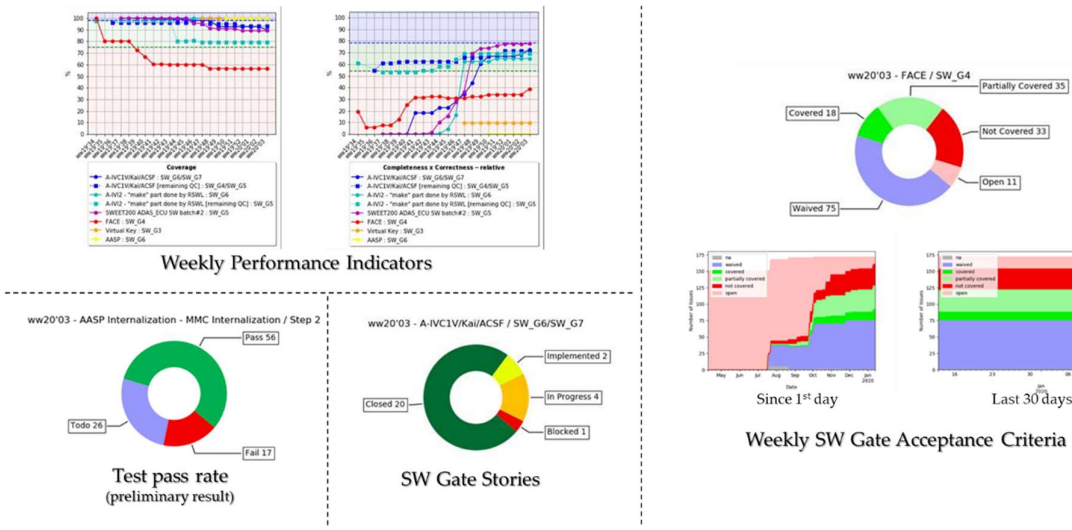

Figure 5 - Examples of several implemented metric graphics

On second half of this year, our focus was on the creation of a precise quality model landscape in the fields of software product, process and project. Therefore, to acknowledge the benefit of such effort, we first performed a coarse quality model survey, identifying 196 distinct models with their number of citations within the survey (cf Figure 6) which was much more productive than Oriol *et al.* [11] paper with 51 quality models: that paper is the one with the most important number of cited quality models to our knowledge. In addition, we analyzed that set of quality models, splitting by basic vs standard vs tailored, and per main technology: components (e.g. COTS), web, service, open-source, reuse. The conclusion of our analysis (see Figure 7), also based on citation numbers and the 9 challenges identified by Thapar *et al.* [14], allowed us to identify an early list of 10 candidate quality models which could be used as a basis for our quality model genome: McCall, Boehm, Dromey, FURPS, ISO 9126, GEQUAMO, Bawane, Alvaro (CQM), Kalaimangal and ISO 25010.

Confidential C

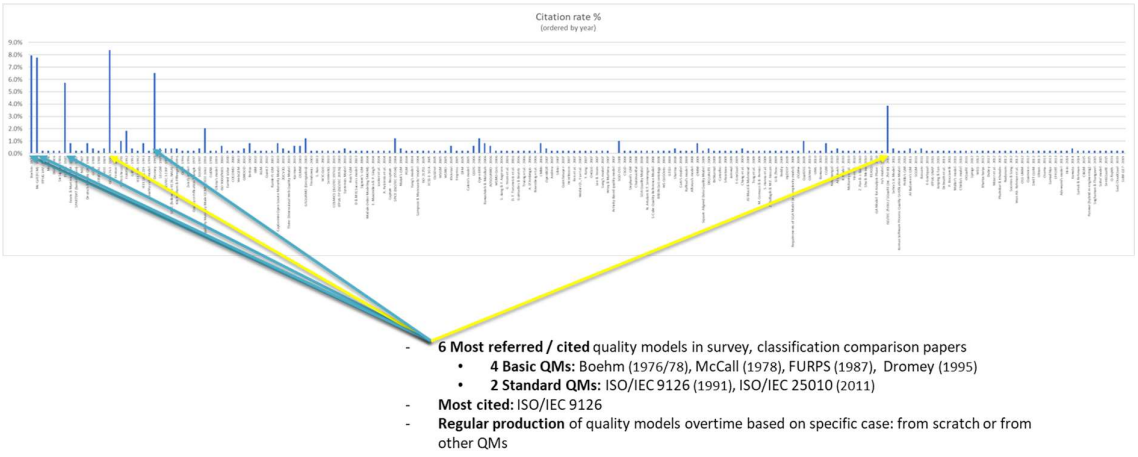

Figure 6 - Coarse quality model survey: list of 196 quality models vs their citations

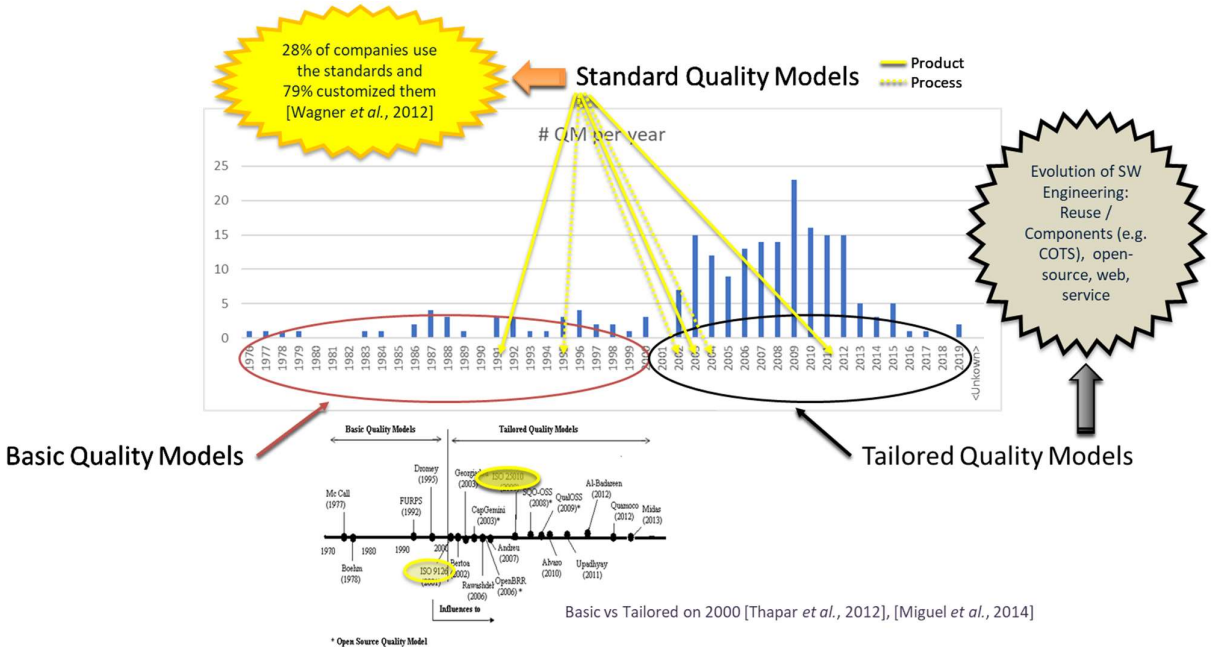

Figure 7 - Coarse quality model survey: detail results of our analysis

Once our coarse survey completed, we decided to strengthen our survey methodology, and by consequence the results here, by initiating a quality model systematic literature review following Kitchenham and Charter [15] guidelines. This our final achievement for this year. We adapted the guidelines to our needs, setting a four-filters stage to filter digital library raw publication results into a short list of papers to study. That process and results of filtering stage are described in Figure 8. We can already note that we are finding more models (see Figure 9 and Figure 10), especially before 2004 years, than with our coarse study, extending the result time range: the 2 first quality models on 1968 Rubey & Hartwick [16] and Shooman [17]. We are also taking into account predictive and statistical quality models (see Figure 11) and during our referencing of each quality model, we are classifying to them across multiple criteria: the 5 quality perspectives of Garvin [18], the DAP purpose of Deissenboeck *et al.* [19], citation number (publisher & google scholar) ….

Confidential C

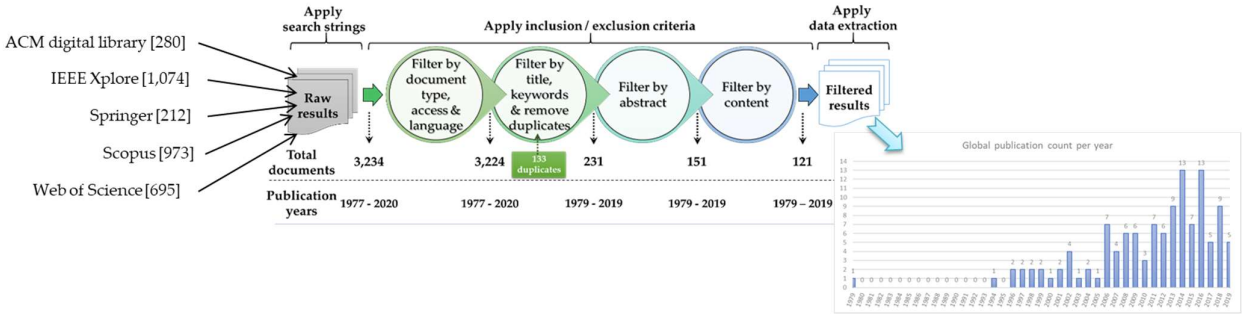

Figure 8 - Our four-filter stage publication selection process

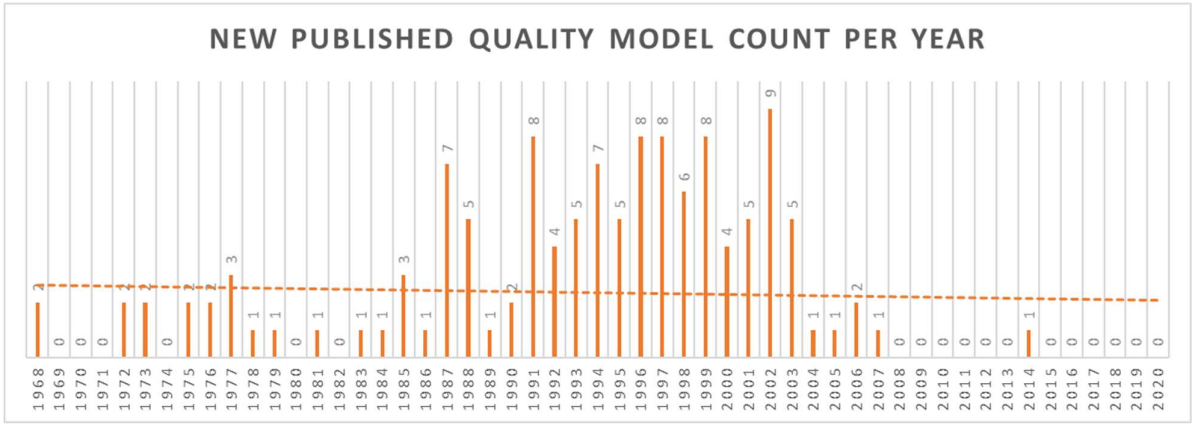

Figure 9 - Systematic literature review: referenced quality models per publication year

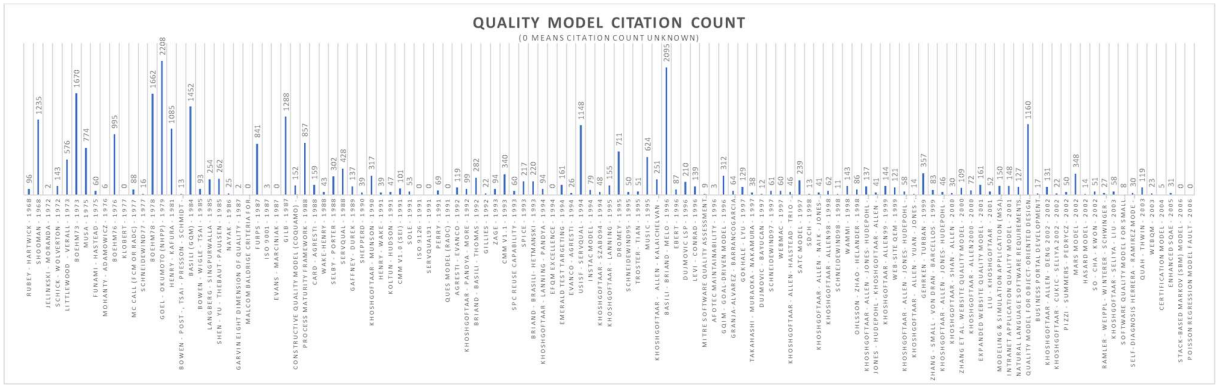

Figure 10 - Systematic literature review: list of the 125 referenced quality models vs their citation numbers

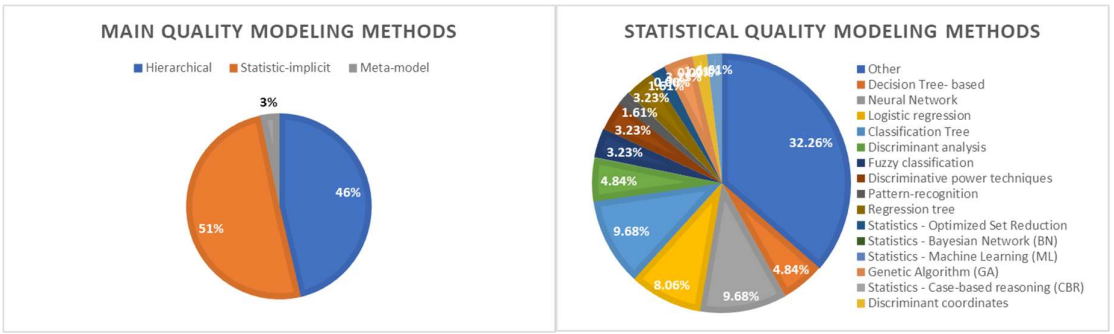

Figure 11 - Systematic literature review: distribution of quality modeling types

Confidential C

To conclude on this section, we would like to highlight that we detailed and promoted part of our contributions with conference presentations and the achievement of two submitted papers. Thus, we presented our last year accepted paper to IEEE International System Conference (SysCon) in April 2019. We submitted a research paper against to IEEE Transaction of Software Engineering journal, with a scope of software engineering. Unfortunately, it was rejected but the reviewers provided some constructive hints to refactor the paper. The second submission this year was done to Embedded Real-Time System (ERTS) 2020, with a scope of embedded systems and software, and contained practical application aspects. That second paper was successfully accepted and presented in the conference. A copy of these two papers is available in the Annex section.

5- Conclusion

In conclusion, like first year, this second year was quite productive in term of contributions and achievements: seven new contributions over which five of them are already acknowledged by peers either by a presented ERTS conference paper or company internal peer reviews and agreements. We note that the scope of our achievements is not only software engineering but also system engineering.

Moreover, we consider having performed a serious step forward in our thesis research work this year. Indeed, it was the first time that we successfully applied our contributions to qualimetry theory against a real use case, and more particularly, the application of our polymorphism concepts to the automotive embedded software.

In our next steps we are going to complete our research work on quality model systematic literature review, taxonomy, genome and then finalize our answer to our thesis subject: "*Study of Qualimetry essentials applied to embedded software product and organization, with considerations to software entropy*".

Confidential C

References

[1] Y. Argotti, C. Baron, and P. Esteban, “Qualimetry Applied to Embedded Software Development: Definition and Approach,” *IEEE Transaction of Software Engineering*, Jun. 2019.

[2] Y. Argotti, C. Baron, P. Esteban, and D. Chaton, “Quality Quantification Applied to Automotive Embedded Systems and Software,” presented at the Embedded Real Time Systems (ERTS) 10th Edition, Toulouse, France, 2020.

[3] VDA QMC Working Group 13 / Automotive SIG, “Automotive SPICE Process Assessment / Reference Model., version 3.1 - revision 656.” 01-Nov-2017.

[4] “ISO/IEC 25010:2011 - Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models,” *International Organization for Standardization*, 2011.

[5] “ISO 26262-6:2011 - Road vehicles - Functional safety - Part 6: Product development at the software level,” *International Organization for Standardization*, 2011.

[6] “ISO/TS 16949:2009 - Quality management systems - Particular requirements for the application of ISO 9001:2008 for automotive production and relevant service part organizations,” *International Organization for Standardization*, 2009.

[7] Y. Argotti, C. Baron, and P. Esteban, “Quality quantification in Systems Engineering from the Qualimetry Eye,” presented at the 13th Annual IEEE International Systems Conference (SysCon), Orlando, USA, 2019.

[8] Y. Boukouchi, A. Marzak, H. Benlahmer, and H. Moutachaouik, “Comparative Study of Software Quality Models,” *International Journal of Computer Science Issues (IJCSI)*, vol. 10, no. 6, pp. 309–314, Nov. 2013.

[9] S. Manoj Wadhwa, “A Comparative Study of Software Quality Models,” *International Journal of Computer Science and Information Technologies (IJCSIT)*, vol. 5, no. 4, pp. 5634–5638, 2014.

[10] M. Moronge Abiud and P. Mbugua, “An analytical comparative analysis of the software quality models for software quality engineering,” *Comprehensive Research Journal of Management and Business Studies (CRJMBS)*, vol. 1 (2), pp. 15–24, Oct. 2016.

[11] M. Oriol, J. Marco, and X. Franch, “Quality models for web services: A systematic mapping,” *Journal Information and Software Technology*, vol. 56, no. 10, Oct. 2014.

[12] M. Nei and W.-H. Li, “Mathematical model for studying genetic variation in terms of restriction endonucleases,” in *In Proceedings of the National Academy of Science of the USA*, 1979, vol. 76, pp. 5269–5273.

[13] “ISO/IEC 9126-1:2001 - Software engineering - Product quality - Part1: Quality Model,” *International Organization for Standardization*, 2001.

[14] S. S. Thapar, P. Singh, and S. Rani, “Challenges to the Development of Standard Software Quality Model,” *International Journal of Computer Applications*, vol. 49, no. 10, Jul. 2012.

[15] B. Kitchenham and S. Charters, *Guidelines for performing Systematic Literature Reviews in Software Engineering*. 2007.

[16] R. J. Rubey and R. D. Hartwick, “Quantitative measurement of program quality,” in *Proceedings of the 1968 23rd ACM national conference (ACM ’68)*, New York, NY, USA, 1968, pp. 671–677, doi: http://dx.doi.org/10.1145/800186.810631.

[17] M. L. Shooman, *Probabilistic reliability : an engineering approach*. New York, N.Y. : McGraw-Hill, 1968.

[18] D. Garvin, “What does ‘product quality’ really mean?,” *Sloan Management Review*, vol. 26, pp. 25–45, 1984.

[19] F. Deissenboeck, E. Juergens, K. Lochmann, and S. Wagner, “Software quality models: Purposes, usage scenarios and requirements,” in *Proceedings of the 7th International Workshop on Software Quality (WoSQ ’09)*, 2009.

Confidential C

Annexes

A- First part of quality model classification and decision tool: get_synonyms.py

This short python program is the first piece of our tool: it makes a web request to www.atlas-smeantiques.eu site in order to get a json based result of synonym and their corresponding constellations. The first text box below is an example of screen display of that python script: it looks for "functional" synonym. The second text box is the python script itself.

```
$ /get_synonyms.py functional
Request word => functional

Constellations:

Lvl: 0  =>
in working order, operable, operational, operative, practical, running, serviceable,
usable, useable, useful, utilitarian, working

Lvl: 1  =>
official
```

```
#!/usr/bin/env python
import sys
import urllib, json

## active or not some debuge trace
debug = 0

try:
  url = "http://www.atlas-semantiques.eu/view/synjson.php?r=" + sys.argv[1] + "&d=EN"
  ## here is an exemple of URL
  ## url = "http://www.atlas-semantiques.eu/view/synjson.php?r=functional&d=EN"
  response = urllib.urlopen(url)
  data = json.loads(response.read())
  if debug:
    print data

  ## Request: our root word
  print "Request word => " + data["request"] + "\n"

  ## Word list: Assume that we are asking only EN => we have only one item
  tab = data["words"][0]
  word_list = tab["word"]
  if debug:
    print word_list
  ## Fermeture : lists of words associated => constellations
  print "Constellations:" + "\n"
  lvl = 0
  tab = data["fermetures"]["fermeture"]
  if debug:
    print tab
  ## for each constellation, from the closest to the farest
  for i in range(len(tab)):
    elt = tab[i]
    print "Lvl:", lvl, " => "
    words = elt["wordRef"]
    ## get all words of that constellation (concatenate for nicer display)
    res =""
    for j in range(len(words)):
      ## we are getting only wordref, so we need to get the real value
      res += word_list[words[j]]["text"]
      if (j < (len(words)-1)):
        res += ", "

    print res + "\n"
    lvl+=1
except:
  print "Bad argument: expecting word to look-for as first parameter"
```


Confidential C

B- Journal paper: IEEE Transaction on Software Engineering

Submitted on the 17th of June 2019, decision as rejected on 23rd of September 2019, Snapshots are below:

Journal 2019 - Qualimetry applied to

Qualimetry Applied to Embedded Software Development: Definition and Approach

Yann Argotti, Claude Baron, and Philippe Esteban

Abstract— Within the aim of delivering high quality level embedded software products, this paper tackles the question of applying qualimetry to embedded software development. Beside explaining the need of improving quality in that context and clarifying the concepts related to qualimetry and its aim - providing the foundation for quality quantification - the paper identifies the main road blockers that prevents from a straight forward applicability of it. Thus, our investigations and resolution for each of the blockers conduct us to rationalize and propose solutions to reinforce the use of qualimetry, fixing some precision due to the discipline's youth. We come up with several contributions: a unified conception for quality model resulting of our quest of a pattern for quality model design, the use of polymorphism as an attribute to compare quality models, and the design of the measurement refinement process. To complete our answer, we analyze what is beneath "embedded software development" and therefore elaborate a model describing and mapping eight distinct objects that are involved in the evaluation of the quality of our object of interest. As a matter of fact, we conclude on the wider applicability of our contributions.

Index Terms—Qualimetry, quality model, polymorphism, metrics, measure, software engineering, software evolution, standards

1 Introduction

TODAY quality is a key aspect that drives software development since its beginning. Delay and coding workload reductions, technical back and forth optimization linked to qualification, customer acceptance and sustaining phases are essential stakes in project cost: solely, these activities represent around 65% of overall cost and non-quality costs 5% of total revenue [1]. As well, bad testing processes can increase the risk of project delay or cancellation by 25% to 300% [2] and we can see regularly a worst case scenario happening, turning reality into serious nightmare. Let us mention a few: over the 1985-1987 period, Therac-25 causes massive radiation overdoses to six patients [3]; on June the 4th 1996, Ariane 5 was self-destructed less than 40 seconds after ignition [4]; on September the 23th 1999, the Mars Climate Orbiter burned in the Martian atmosphere [5]; on August the 1st 2012, Knight Capital Group lost $440 million in 45 minutes [6] and the Takata's bankrupt occurred in June the 26th 2017 [7]. In the first four examples, the catastrophic event was a direct consequence of an uncaught software bug in opposite to the last one, which was directly linked to an unaddressed major defect in airbag, illustrating that quality is a matter of all system types, including all subsystems that composed them. We also note that the three first cases are more particularly within embedded software scope.

Qualimetry [8]–[10], a 50 years old science aiming at quality quantification, brings a set of good practices, fosters dysfunction detection, enhances control, increases efficiency and productivity, not only to individual contributors but also to overall organization. Mastering this scientific discipline is therefore decisive in terms of quality characterization, assessment and control through standardized or tailored quality models [11], metrics and controlled processes.

This article guides our journey of applying qualimetry to embedded software development, setting the focus on the definitions and the approaches which are the preliminary steps of this journey. Moreover, since qualimetry can be applied to any domain, narrowing the scope of our study to the embedded software development domain doesn't prevent our contributions to be extended and applicable to other domains than embedded software. On the contrary, its applicability scope is multi-systems and multi-fields oriented. Moreover, our analysis methodology is inspired from a diversity of domains, including concepts from systems and software engineering, biology/genetic or socioeconomics for instance.

Thereby, section 2 of this paper first addresses the context and problems of qualimetry applied to embedded software development. In this section, we study and rationalize concepts related to quality, qualimetry, measurement and objects of interest; we also identify the road blockers preventing our journey to move forward. Then, the next sections review our contributions and solutions to overcome the current road blockers. Consequently, section 3 introduces a unified conception for quality model, crawling over methodologies coming from various horizons, including also qualimetry one. This section ends with the usage of this unified conception in an example: the comparison between ISO/IEC 9126:2001 [11] and ISO/IEC 25010:2011 [12] quality models. With section 4, we introduce a new process to rationalize the transformation sequence of raw measurements into quality measurements with a refinement process. Section 5 focuses on our object of interest "the embedded software development", analyzing what it is really

- *Y. Argotti is with the System Engineering Department, Université de Toulouse, INSA, LAAS-CNRS, Renault Software Labs, Toulouse, France. E-mail: yann.argotti@ laas.fr*
- *C. Baron is with the System Engineering Department, Université de Toulouse, INSA, LAAS-CNRS, Toulouse, France, E-mail: claude.baron@laas.fr*
- *P. Esteban is with the System Engineering Department, Université de Toulouse III, LAAS-CNRS, Toulouse, France, E-mail : philippe.esteban@laas.fr*

xxxx-xxxx/0x/$xx.00 © 200x IEEE Published by the IEEE Computer Society

Confidential C

and which quality model, if any, is already covering adequately it. The last section, 6, finally concludes on the first part of our journey, preparing next one.

2. Context and Problems

From the introduction, we understand that it is fundamental to ensure that a right level of quality is achieved for the embedded software.

But what are really quality and qualimetry? And, what are precisely the concepts behind quality model, measurement and object(s) of interest applied to embedded software development in order to proceed it right?

2.1 Quality and Qualimetry

Quality is a quite popular field of interest for people worldwide, with a relatively constant interest trend year after year, confirmed by nowadays information technology such as internet search engines [13]. However, what also points out from these search engines is the myriad of possible definitions for this word, letting us with a sense of no precise unique definition afterwards. So, our aim here is not to deep dive on all possible definitions to conclude on the best suitable one, but rather build a core knowledge about this concept to be able to define, characterize, use, evaluate and predict accurately quality.

Starting from human history, we can consider that the premises of quality were given by the 5th century B.C. Greek philosophers that are Socrates, Aristotle, Protagoras, Heraclitus and Plato [14], [15] in the quest of "*what is knowledge?*" and later summarized by Aristotle with "*By 'quality' I mean that in virtue of which people are said to be such and such [...] The body is called white because it contains whiteness.*" [16]. Then, according to these early definitions, even if an ambiguity remains due to people interpretation, it appears that quality is an object property by itself, and an object can be anything, from someone to something.

To proceed further, we are considering several sources of word definitions. The first obvious one comes from the Oxford English dictionary [17], where quality is explained through two definitions, or interpretations:

- "*The standard of something as measured against other things of a similar kind; the degree of excellence[1] of something*"
- "*A distinctive attribute or characteristic possessed by someone or something*"

Not only we retrieve the object property or attribute characterization as well as the fact that an object can be of any entity type, but also, now we are referring to a standard for measurement or comparison versus similar objects, which leads to the measurement of the excellence degree of that object.

To complete our understanding of quality and since our scope is within engineering field, we naturally consider well-known international industry standards, limiting our focus to two major standard organizations[2] in engineering: ISO and IEEE.

Let us first consider the international standard for quality management, aka ISO 9000:2015 [22], where quality is defined as:

"*The quality of an organization's products and services is determined by the ability to satisfy customers and the intended and unintended impact on relevant interested parties.*"

with this scope:

"*The quality of products and services includes not only their intended function and performance, but also their perceived value and benefit to the customer.*"

Our second reference is not dedicated to quality by definition but is used by the International Software Testing Qualification Board (ie ISTQB) glossary [23]. This is the IEEE standard glossary [24] where quality is defined this way:

"*The degree to which a system, component, or process meets*

1 specified requirements.

2 customer or user needs or expectations."

So interestingly, even if we are noticing differences in their descriptions to reach quality and the object of focus mainly due to each applicability context, we acknowledge an existence of a convergence in the interpretation.

As a result, from these different sources, we summarize that quality is defined, or characterized, by:

the set of distinctive intended, and unintended, properties of an object of any type that "become apparent during its intended use (operation, application or consumption)" [15] within the scope of a degree of excellence for that object.

Finally, there is a last base knowledge about quality we must integrate which is 'integral quality'[3]. Indeed, integral quality is the sum of quality and cost effectiveness (1) in socioeconomic field [15] [25], where cost effectiveness is described by the object properties linked to the input capital for production and consumption of that object. The distinction of quality versus integral quality is important when we use qualimetry (introduced in next paragraph) to look for quality characteristics, quality model, measurement, assessment and quality control.

$$Integral\ Quality = Quality + Cost\ Effectiveness \quad (1)$$

Now that we have clarified what is beneath quality, let see how to evaluate it thanks to qualimetry.

Qualimetry, from the Latin *qualis* "*of what kind*" and the Greek *μετρεω* "*to measure*", can be described as the science of method and problem solving for quality quantification of any kind of object such as service, product, people, project or process [15].

The concept of quality quantification is not recent as was indicating G.G. Azgaldov et al. [15], the founder of

[1] Excellence is different from quality but, from its latin definition *excellere*, we understand that it corresponds to the notion of "surpass". Then here, the idea is to shoot for the best.

[2] Many other valuable contributions exist such as W.E. Demings [18], B. Kitchenham and S. Pfleeger [19], D. A. Garvin [20], R.M. Pirsig [21], to cite few, but again, the purpose here is not to debate on which definition is the best one or build an history of the definition of quality, and therefore we limit ourselves here to two commonly used reference standard organization in embedded software.

[3] We remarked a debate about integral quality [25], however this aspect is important as it is concerning cost characteristics which may vary for same type of object of interest, depending on project model, for instance.

Confidential C

qualimetry; J. Diez in [26] also clearly emphasized this fact, citing works done by H. Helmholtz in 1887 or by N. Campbell in 1920. However, until late 1960's the quantification of the degree of excellence was exclusively done for only one specific type of object at a time, primarily product oriented one and without direct reuse or generalization over other similar objects. On the beginning of 1968, in former U.S.S.R., a scientist group of interest around problems linked to quality quantitative evaluation and control, published a common summary paper of their workshop [8]. The force of this group was due to the fact that its members were coming from a large variety of domain horizons and shared the same concerns. It was the ignition of an international discussion that lead to the birth of qualimetry, a new scientific discipline, during 1968.

In addition, G.G. Azgaldov et al. [15] demonstrated that qualimetry was not only a scientific discipline but a real science by itself, reminding what Plato said on the 5th century B.C. "*Exclude from any science mathematics, measure and weight, and it is left with very little*". In 1981, that science was split into two distinct disciplines: theoretical qualimetry [27] (i.e. focusing on problems and method issues, with a mathematical view to object to evaluate) and applied qualimetry [28] (i.e. application of qualimetry to evaluate type of objects that were not evaluated before). We remark also that qualimetry relies on domain-independent concepts, or foundations, and therefore can be applied to any domains.

To describe the structure of these concepts, we make an analogy with architecture, borrowing Doric architecture vocabulary. Qualimetry sits on an entablature, symbolizing both theoretical and applied qualimetry aspects, which itself relies on two architraves A) quality model (i.e. deriving object characterizations, following a set of rules, into model) and B) measurement (i.e. linking characteristics to metrics, combinatory formula and decision thresholds to allow assessment and quality control). This upper structure is supported by 6 pillars: 1) object analysis, 2) derivation rules, 3) weight factors, 4) theories of measurement, 5) aggregations and 6) thresholds. Finally, the whole structure is founded on object(s) of interest (i.e. object(s) candidate to be quality quantified). Fig. 2 represents our proposal of the House of Qualimetry and its Pillars [29], making an analogy with the House of Quality [30], where we can retrieve the three key elements (object(s) of interest, quality model, and measurement) that needs our attention in order to apply qualimetry.

In the next sections of this paper, we review and detail the different elements that compose this *House of Qualimetry*, starting by the two architraves – Quality Model and Measurement - and the foundation – Object(s) of Interest.

2.2 Quality Model, Measurement and Object(s) of Interest

A. Quality Model

A quality model, for an object, is a theoretical description and characterization of its quality [32], thus its creation is a critical step in qualimetry. In the aim of creating quality

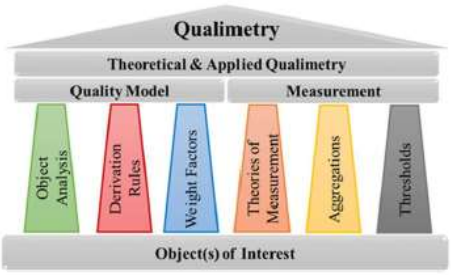

Fig. 2 The house of qualimetry and its pillars [29]

model, we can rely on several works, that are chronologically presented here after.

In parallel to the birth of Qualimetry on 1968, Rubey and Hartwirck [33] setup the first attempt to have an organized software quality based on some code attributes with their corresponding metrics. This initial study about quality attributes were used as a basis for multiple work including the ones done by Boehm *et al.* [34] with the achievement of software quality tree in 1976. Then appeared in 1977, the Factor / Criteria / Metric (FCM) method by McCall *et al.* [35] and its generalization in 1994 by Basili *et al.* [36] into the Goal / Question / Metric (GQM) method. They were the most usual methods for designing quality models; as an example, ISO/IEC 25010:2011 [12], a reference standard for systems and software quality domain, is based on GQM. Their principle was to reply to questions such as "*what are the object of interest quality goals, the questions related to these goals and their corresponding metrics?*". However, with these methods, we miss some important aspects in the design, generalization and adaptation methodology because the scope is narrowed to a specific product; for example, its GQM focus may not fit the needs for slighty different products.

In 2009, Deissenboeck *et al.* [31] defined a categorization of quality models over three different purposes: Definition, Assessment and Prediction (see Fig. 1). In 2013, Wagner [32] consolidated the knowledge related to DAP models, developing further their descriptions and characteristics. Unfortunately, he didn't provide a methodology to create such model and his focus was restrained to two types of objects: software product and software project.

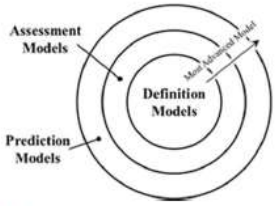

Fig. 1 The centric based DAP[4] classification [31]

[4] DAP stands for Definition, Assessment and Prediction.

Confidential C

In 2015[5], Azgaldov *et al.*[15], with a socioeconomic perspective that can be generalized to any field, provided additional characteristics, defined specific rules and methods to accomplish the design of such model, and established an algorithm for quality assessment. However, in their methodology, they restricted themselves to only one specific representation of model, a hierarchical one, also called 'tree structure', while Wagner shown that other depictions are possible, such as statistical model like the maintainability index done by Coleman *et al.* [37], for example.

In parallel to these main streams of contributions, we can cite various achievements for quality models. These achievements are resulting by the elaboration and publication of quality models. We can summarize and illustrate these types of achievement *via* the ISO/IEC 250nn [14], [43]–[47] examples done from 2007 to 2016. Unfortunately, none of them is giving either methods, rules or general characteristics about quality model design but rather provide static quality model with its characteristics linked to the object to be measured. In the ISO/IEC 250nn case, the object of interest is mainly software product, even if we could also consider system product.

Thus, we don't have a unique and obvious approach for creating quality model but rather several more or less distinct ones that need to be unified under the qualimetry banner to design properly our quality models.

B. Measurement

The second key element, fundamental to proceed from a quality model definition to quantification, assessment and quality control, is the measurement. Measuring is the action to take a measure, that is to say, to "*ascertain the size, amount, or degree of (something) by using an instrument or device marked in standard units.*"[6].

Historically, we find three may streams of measurement theory [43]. The "representational measurement" theory, initiated with the work of von Helmholtz in 1887 [44], focuses on setting a relationship between objects and number systems *via* equivalence classes. A second theory, the "operational measurement", was introduced by Bridgman in 1927 [45]; its focus is put only on operations used to proceed to measure, neglecting relationships between equivalent measured objects. The last stream is not a single theory but rather a sort of "melting pot" of various other theories [26], [46], minor in front of representational and operational theories. In many of these theories, major or minor, there is a common factor that is underlying: the scale. We note also that despites which definition is used, "*size, amount, or degree*" and "*standard units*" are direct references to the scale theory of measurement.

Indeed, here scale is essential when we measure quality attributes with the goal of either validating that the measurements are matching some criteria (e.g. being above a certain threshold) or controlling quality by comparing their difference over a certain period or even, predicting what will be quality measures. It enables measurement interpretation, standardization and comparability by describing mathematically types of element with their corresponding operators, properties and functions.

In 1946, *S. S. Stevens*[7] introduced and published a theory of scales of measurements [47] which is part of the representational measurement theory and still widely used by scientist nowadays despite criticism from statisticians [48]. Stevens categorized data into four typologies or scales: nominal, ordinal, interval and ratio.

The 'nominal' scale represents the group of data related to labels or type of numbers and words or letters (e.g. {"Red", "Green", "Blue"} or {"True", "False"}). It can be used for classification and membership assessment. The 'ordinal' scale corresponds to the data set characterized by order, rank, and therefore allows comparison and sort of elements: for instance, it is possible to tell that a quality result is better or worse to another one. The third scale, 'interval', is extending the field of possible operations one step further by adding difference and affinity operations between values to be meaningful. Thereby, distances between measures are used to take action and decision. For example, man can tell that a quality result is not only better or worse than previous one but also about how much it progresses in term of difference. And last scale is 'ratio'. At that level, and as its name is indicating, the ratio - or magnitude - between measurement values is relevant and useful. It has also the important property of having a unique and non-arbitrary zero value (i.e. Kelvin temperature scale is a good illustration of a ratio scale since its zero absolute corresponds to -273.15 ° Celsius, while Celsius temperature scale is an interval scale).

Another aspect, independent to scale definition but linked to measure, is measurement property. In fact, there are two main properties[8] that are critical to qualify a measure [50]. These properties are reliability and validity of the measure. The reliability means how repeatable over time the measurement is providing consistent results (i.e. with same exact condition and same exact way to measure, measurement values must be equal). Part of reliability is measurement error and internal consistency. Concerning validity, this property means whether or not measured value is measuring what we are expecting to measure; it includes content validity, construct validity and criterion validity. The Fig. 3 gives a good visual explanation of impact of these two measurement properties, based on the analogy of "*target shooting*". Here, our "sniper" is trying to hit the target shooting several times in the same conditions. If all shootings are in the target, result is valid, and if shootings are

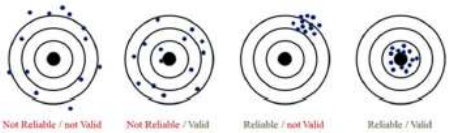

Fig. 3 Illustration of the concept of reliability and validity of measures [49]

[5] We are tented to say that this work occurred in 2013 but we didn't find any clear indication. So, we are indicating 2015 because of the date of this translation from Russian.

[6] "to measure" definition from Oxford dictionary

[7] An American psychologist who was the founder and director of psycho acoustic laboratory at Harvard University.

[8] It is possible to have additional properties of measurements likes some of the ones use in health related field [50], construct validity, responsiveness for measurement tools or interpretability, for instance.

Confidential C

close together, we can conclude that the shootings are repeatable over time and therefore result is reliable.
Starting from left to right, on the most left target, we notice that the shootings are both spread (i.e. not quite repeatable), inside and outside the target (i.e. they are not valid shootings). On the second target, the shootings are still spread but now all of them are inside the target; we can conclude that the shootings are valid but not repeatable, or reliable. With the third target, all shootings are grouped but some of them are outside; we understand that shootings are repeatable over time but not valid. Finally, in the last case not only shootings are close together and they are all inside the target; the result is therefore valid and reliable.

Parallelly to these characteristics, we have the rating of each quality characteristic, which consists in combining measured value to a scale point on an ordinal scale [51] (e.g. {Lowest; Low; Medium; High; Highest}), aggregation operators [32], [52] to combine multiple or range of measures into one, for instance, and thresholds [15], [32] to support decision of rejecting or, on the contrary, accepting during quality assessment or control.

Unfortunately, beside the enumeration of measurement properties or characteristics, the only available process related to measure consists in asking user to pay attention to properties and then aggregate measures done via some measurement functions [44]-[47]. Thus, there is no rigorous methodology or process taking benefits of all these properties or characteristics to transform raw measures into quality measures.

C. Object(s) of Interest

Our last key element in our journey of applying qualimetry to embedded software development corresponds to the objects of interest that we have behind "embedded software development". We put an 's' to objects because we already feel that there is not a single object behind this concept and immediately, we can cite two, embedded software product and project.

The life cycle of a software, like a system, is a suite of more or less distinct consecutive stages. These stages are implemented over 14 technical processes, if we refer to ISO/IEC/IEEE 15288:2015 [53] with regards to system context, with 18 additional software specific processes[9], if we refer to ISO/IEC 12207:2008 [55], and some of theses 32 processes are being performed in parallel. A typical product life cycle, shown in Fig. 4, is provided by ISO/IEC/IEEE 15288:2015, addressing "*Systems and Software Engineering - system life cycle processes*" [53]; this life cycle gets its foundation from ISO/IEC/IEEE 24748-1:2018 [56]. It promises five main typical stages. The first one is the concept stage where new concept, idea, technology or feature is studied, including some exploratory prototype, requirement elicitation and project planning. Second stage is the development one. Here, the system architecture and design are created, the initial system is implemented and qualified. In the production stage, the system is manufactured or produced and then delivered to customer. We regroup the next two stages into one because they are happening in parallel. Indeed, utilization and support stages correspond to the operational and evolution life of the system which includes deployment, use, corrective and evolutive maintenance. The final stage corresponds to the retirement of the system, or say in other words, the system end of life management.

Fig. 4 Typical System Life Cycle from ISO/IEC/IEEE 15288:2015 and ISO/IEC/IEEE 24748-1:2018

Another model, not antinomic to this typical one but integrating much better the system or software evolution aspect, is the Staged Model [57], [58] illustrated in Fig. 5. The move across the five stages, that are Initial development, Evolution, Servicing, Phase - out and Close down, is conditioned by the level of conservation of the software or the system familiarity, for instance.

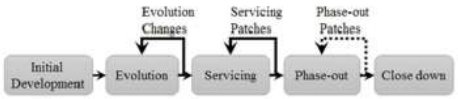

Fig. 5 Life Cycle: Staged Model

In this model, the software product is moving then from a stage where it is under development to operational stages where it is in use and in evolution, until its close - down, corresponding to its retirement. During the initial development stage, the software is created according to customer requirements. The end of this stage is triggered with the first version delivery of the software product. The second stage corresponds to evolutions, including corrections, of the software product. Here, the developments are associated to the product in use feedback, environment evolution and, customer and market needs. The next stage, servicing, is close to the evolution one, but customer support is more corrective and there are only minor and specific evolutionary changes. The software product is moving to that stage because of a serious loss of familiarity[10]. When the loss of familiarity reaches a level where effort cost is considered too high by the software organization, we are entering to the phase-out stage which should be maintained until the software product replacement is developed and delivered to customer.

Both ISO/IEC/IEEE 15288:2015 and Staged Model life cycle highlight that software product quality model is somehow a polymorphous quality model of our product of interest. The polymorphism means that a generic quality model depicting software product, for example, can be derived to a variety of more precise quality models, depending on the type of objects (e.g. web-service software, operating system

[9] These 18 software specific processes are grouped into three categories: 7 software implementation processes, 8 software support processes and 3 software reuse processes. However, since 2017 and the new version ISO/IEC/IEEE 12207:2017 [54], these processes are included into technical processes for implementation processes, into technical management processes for support processes and into for reuse processes.

[10] Software product complexity, major environment changes or / and loss of key elements (e.g. a software architect) prevents evolutionary changes to be handled with a reasonable cost for the software organization [58].

Confidential C

software, engine control embedded software ...). It means also that the quality model may evolve based on the process of life cycle stage, for instance development versus utilization stage. If we take the ISO/IEC 25010:2011 [12], we have two strictly distinct quality model forms associated to software product. The first, focusing on system/software product in development, is composed of 8 characteristics and 31 sub-characteristics while the second one, focusing on software in use, is composed of 5 characteristics and 12 sub-characteristics (see Fig. 6). Unfortunately, there are nothing related to the development phase versus evolution phase.

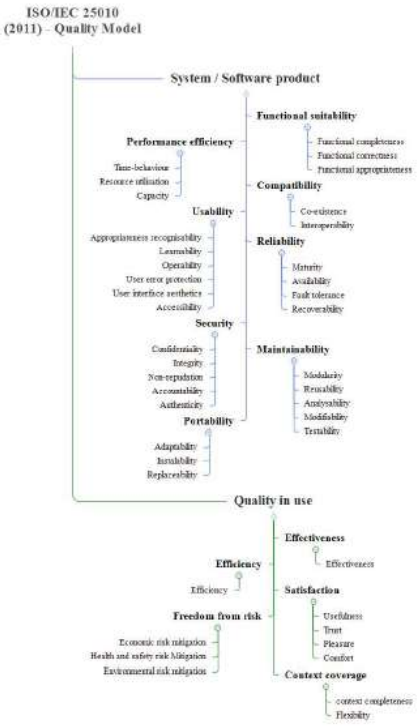

Fig. 6 ISO/IE 25010:2011 quality model, including "System / Software Product" model and "Quality in use " model

In conclusion, this quick analysis demonstrated that there is not an obvious nor existing answer to identify the objects of interest for our case study and their respective quality models. Therefore, we must perform a deeper analyze not only to list all those disctinct objects of interest but also clearly identify the relationships between them in a matter of understanding their mutual impacts[11].

[11] This is a mandatory aspect when considering prediction and result analysis, associated to assessment or quality control.

[12] We use definition of "*model*" from the Oxford dictionary [60] in the domain of "*quality*".

Over section 2.2, we have seen that we have three types of problems to solve in order to be in a position where we can apply qualimetry to embedded software development: quality model conception, measurement transformation and objects of interests belonging to embedded software development. Next sections sequentially address each of them.

3. A Unified Quality Model Conception

During the previous sections, we settled the foundations of the quantification of the quality. Indeed, we understood that quality is the reflection of distinctive intended and unintended properties of an object that "*become apparent during its intended use (operation, application or consumption)*" [15], and that qualimetry is the science to quantify quality, using quality models.

Unfortunately, and even if there were some commonalities between the various characterization approaches of quality model, none of them encompass the others. So, in the following paragraphs, we elaborate an inventory of quality model attributes, reviewing and comparing the detail attributes with the aim of identifying a pattern in quality model design and consequently propose a unified conception for quality model.

1) "*Evaluation context and plan*" attribute: The first consideration that requires our attention is the definition of quality model evaluation context [15] and plan [59]. These elements correspond to the documentation aspect about requirements, motivation, purpose, limitation, applicability, stakeholders, object(s) of interest, usage context and any details about evaluation within the quality model. This is close to a master test plan [23] where we aim at documenting all major information that describe how testing is planned, covering and managed at the various levels. Most of the time, this critical document is missing and without properly performing that step, we won't be able to have a quality model that fits our needs.

2) "*Purpose*" attribute: By definition[12] a quality model is a representation of the quality of an object of interest over a proposed data structure. That representation can be also used to "*assess and/or predict quality*" [32]. Therefore, we see that a quality model can be characterized by its intended purpose, starting from quality definition, then assessment and ultimately prediction.

Definition. As we indicated previously in section 2.2.A, this categorization was first introduced by Deissenboeck *et al.*[31] via the centric based DAP classification (cf. Fig. 1). We found at its center the quality model describing all the quality characteristics and properties related to the object of interest. A good illustration[13] of definition models are the models described by ISO/IEC 25010:2011 [12] (cf Fig. 6

[13] We consider ISO/IEC 250nn series as good illustration because we share the same context: software product.

Confidential C

where we created a jointed version of the two[14] quality models of this standard). Indeed, a quality model of software product can be decomposed into two categories:

- the main quality characteristics of software product, composed of one functional characteristic, itself refined into three sub-characteristics, and 7 non-functional characteristics with their 28 sub-characteristics in total.
- the quality characteristics of software product in use, described over five characteristics and 12 sub-characteristics.

Assessment. If we complete the definition model with further information like, for instance, metrics, aggregation methods, acceptance or reject threshold, assessment process, then we come to the assessment model. If we take the previous ISO/IEC 25010:2011 example, assessment -included in evaluation process- is described by ISO/IEC 25040 [61], completed by ISO/IEC 2502n [38]–[42] with regard to measurement aspect (see Fig. 7 illustrating a possible extension for reliability/maturity assessment and linked to software product quality model definition).

Prediction. The last advancement level (cf. Fig. 1) of the quality model is the prediction one which is used to predict object quality, relying on statistical and data learning or mining methods applied to assessment model. So, jumping back to our example, we are reaching the current limit of the ISO/IEC 250nn standard. Prediction is the area of business intelligence and decision support system (i.e. DSS) tools. Iqbal and Babar [62] are providing an example of such DSS based on ISO/IEC 25010:2011 and fuzzy logic.

Eventually, we can find a model combining these three purposes: this is multi-purpose model. We can cite as a multi-purpose model example COQUAMO [63] model. Nevertheless, such model has limitations due to unclear relationships between the diverse types of measurements that are linked to project phase, quality factor (also known as quality characteristics) and purpose of the measurements (e.g. specification and design metrics, module metrics or testing metrics).

3) *"Quality Evaluation Method (QEM) to assess quality"* attribute: The third attribute that describes a quality model is Quality Evaluation Method (QEM). We can find the concept in socioeconomics field [15]. The idea behind is to identify how accurate will be our assessment and what type of source of information is going to be used to identify quality properties. The QEMs are split into two parts done sequentially.

First, we must decide which method to apply to assess quality over three possibilities: rigorous, short-cut and approximate. With a *rigorous method*, as we could imagine, we are considering all possible known valid approaches and techniques to evaluate quality property value, integrating all parameters that affect this property. Certainly, such method gives minimum error in the assessment, and then maximum of reliability, but the drawback of this exhaustivity is that it requires an extreme work effort. At the opposite, we have the *short-cut* method where we consider only single value for each property. The result is less intensive in term of workload, but we have to deal with a possible maximum error and minimum reliability allowed. The third method is simply a mix between rigorous and short-cut; that tradeoff is called *approximate method*.

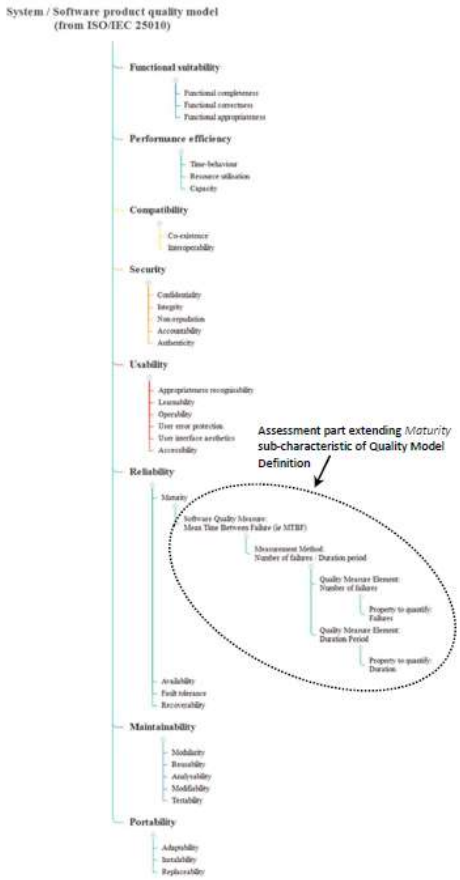

Fig. 7 Example of assessment model extending ISO/IEC 25010 software product quality model

4) *"Source of information for value in QEM"* attribute: Once decided how exhaustive and accurate our quality model is going to be to assess quality, the second step with QEM is to pick up which method we are going to apply to identify the sources and weight factors of each indices and properties. As previously, we have three levels based on the

[14] In this standard, we have a third quality model but dedicated only to data quality that may be handled in the software product but not characterizing the quality of the software product. So, we are skipping it here.

Confidential C

implication of experts to handle those identification tasks. Proportionally to expertise involvement magnitude, we have expert method, hybrid method and non-expert method, also called analytical method.

5) "*Data organizational type*" attribute: A fifth attribute of quality model is about the way of combining and organizing the data, and their relationship, related to the quality knowledge such as quality characteristics and sub-characteristics, for instance. We can identify three main distinct data organizational model types [32]: hierarchical, meta-model based, statistical and implicit model.

The most common type of quality model is the hierarchical (or tree) model (e.g. Fig. 6). Quality elements are organized starting from the most general concept down to the finest details. Moreover, each level of the tree corresponds to the same level of characteristic abstraction and can regroup same type of sub-characteristics under them. Also, due to the simplicity of how to arrange knowledge and link elements together, this is the preferred solution for definition model. We note as well that one of the first software quality model, done by Mc Call in 1977 [35] (cf. Fig. 8), was a hierarchical based model composed of three product stage categories subdivided into a total of 11 quality factors, themselves regrouped into set of two to five quality criteria.

The next quality model data organizational type, meta-model, is aiming to help building specific model. It is composed of more generic and abstract elements such as quality characteristics, factors, properties, metrics and rules linked together. Fig. 9 is representing a meta-model we propose, which can be used to derive specific assessment quality model for software product. To illustrate the meta-model concept applied to quality model, we build in UML this meta-model generalizing the software product quality measurement reference model defined in ISO/IEC 25020 [32] together with the relationship among properties to quantify, measurement method, quality measure elements and quality measure that are described in ISO/IEC 25021 [33]. However, even though a metamodel brings some genericity and over the existing meta-model set, it does not exist a quality model applicable to any case yet. This is also confirmed by Wagner is in [32].

The third and last type we are considering is the statistical and implicit model. Its particularity is to link not deterministically measurement with corresponding quality characteristic, supporting the explanation of influences or impacts, but rather statistically, exploiting tools like, for instance, Bayesian Network [64] or Bayesian Belief Network [65]. Such model is often used for prediction purpose (e.g. the maintenance index model defined by Coleman *et al.* [37]).

6) "*Derivation rules*" attribute: Next, and therefore sixth attribute to include into our quality model creation, is directly bonded to the hierarchical or tree structure. As a matter of fact, tree conception must rely on a set of rules to derive it from set of quality property or characteristics. We can certainly exercise some computer science theory about tree balancing, but theoretical qualimetry has formalized around 30 rules to support adequately those tasks. Azgaldov *et al.* [15] emphasize the most important ones. TABLE 1 is listing the 10 main global derivation rules while TABLE 2 is focusing on the 6 main specific ones.

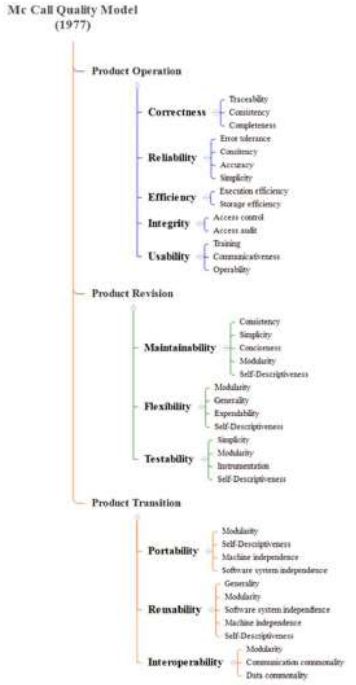

Fig. 8 – One of the first software product quality model: McCall (1977)

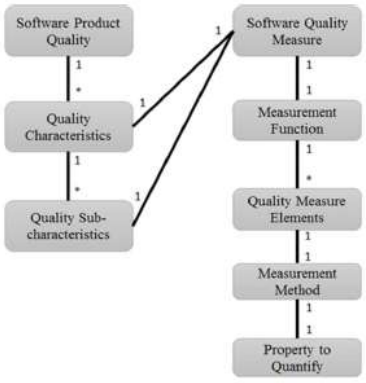

Fig. 9 A meta-model proposal of ISO/IEC 25010 with ISO/IEC 2502n

7) "*Weight factor*" attribute: Seventh attribute of quality model interest is weight factors. That attribute is defined for

Confidential C

each characteristic, sub-characteristic, property and index of our quality model. It corresponds to the importance that each element has against its peers' elements (i.e. the elements located at the same level of hierarchy in a tree case, for instance), so their impact could be decisive. Hence, specifying them may require a certain expertise.

TABLE 1
GENERAL RULES FOR QUALITY MODEL TREE DERIVATION

Id	Rule
Global rules	
1	Maximum height of tree
2	Branch a tree until only simple or quasi-simple properties remain at its top tier
3	Preference Indifference of Properties in a Group
4	Exhaustive consideration of the Application features of an object
5	Exclusion of reliability [15] properties (ie this must be part of integrated quality index => K_{use})
6	Structural rigidity of the primary tiers of a tree
General sub-tree rules	
7	Division by an equal characteristic
8	Functional orientation of property statements
9	Necessary and sufficient number of properties in a group
10	Reference number of purpose properties within a group

TABLE 2
SPECIFIC RULES FOR QUALITY MODEL TREE DERIVATION

Id	Rule
Specific rules for the application of the expert method to weight factor	
11	Random order of properties in a group
12	Minimum number of properties in a group (max ≤ 9 [16])
Specific rules to be used if the amount of information obtained in a quality assessment can be reduced thought the use of the rank scale	
13	Exclusion of equally expressed properties when the rank scale is admissible
14	Truncated tree when the rank scale is admissible
Specific rules to be used if the amount of information obtained in a quality assessment may/may not be reduced by more precise methods	
15	Incomplete tree when a short-cut assessment of quality is admissible
16	Complete tree when exact quality assessment alone is admissible

8) "*Polymorphism*" attribute: In addition to the attributes we enumerated, to take into account our remark in section 2.2.C related to the capacity a quality model to describe different type of objects, we are proposing to add a last attribute that we call polymorphism. This is the same concept that we have in object oriented programming [66]. We note that a quality model can cover multiple "*forms*" of objects, but also it can be a "*form*" of another quality model, inheriting from it. That polymorphism can be linked to either differences between objects of interest or the development life cycle phases of the object(s) of interest. We complete this concept with a characterization of its degree versus other quality models.

Indeed, we evaluate the degree of polymorphism π using nucleotide diversity formula (2) introduced by Nei and Li in 1979 [67] where we consider the analogy of DNA sequences as quality model characteristics sequences and therefore x_i is the frequency of the i-th sequence in the quality characteristic / sub-characteristic population and π_{ij} is the number of quality sub-characteristic differences per quality characteristic site between the ith and jth sequences.

$$\pi = \sum_{ij} x_i x_j \pi_{ij} \qquad (2)$$

To finish on the quality model topology enumeration, we have been reviewing height distinct elements that must be taken into account in order to appropriately construct the expected quality model: "*Evaluation context & plan*", "*Purposes*", "*QEM to assess quality*", "*Source of information about values in QEM*", "*Data organizational types*", "*Derivation rules*", "*Weight factors*" and "*Polymorphism*". Furthermore, we have noticed in the literature three main streams of works that summarized the characterization of quality model. Wagner [32], with a focus around software product and project, explains and details general attributes of quality model. The second one, ISO/IEC 250nn [12], [38]–[42], [61], like the myriad of quality models that are published in the literature, is providing specific quality models with no characterization of quality model to help to design other ones; we also note that the focus here is software product. The third one, Azgaldov *et al.* [15], is following a socioeconomics approach; it gives quality model descriptions and an assessment methodology.

It is interesting to note that none of these approaches simultaneously contain all those attributes with their full set of distinct contents (e.g "purpose" attribute has {"Definition", "Assessment", "Prediction", "Multi-purpose"} set of distinct contents). We remark that 6 of theses attributes are shared among the different approaches, one attribute is unique, the derivation rules, belonging to Azgaldov *et al.*, and one attribute is new: the polymorphism. Furthermore, with the comparison synthesis of these works over the quality model attributes, done in TABLE 3, we can easily highlight the element composition, or union, that we must perform to create a unified conception for quality model. Our result is summarized in Fig. 10 and represents all the ordered steps to follow sequentially during quality model conceptions.

Thus, our proposal consolidates current qualimetry field related to quality model conception, based on knowledge and methodology coming from system and software fields. In addition, we add a missing concept that we consider as fundamental on both quality model design, implementation, comparison and evaluation: the polymorphism.
To illustrate the use of this work, we exercise our 8 quality model attributes against two standards: ISO/IEC 9126:2001[11] and ISO/IEC 25010:2011[32]. The second standard is the official new substitute of the first one, and therefore corresponds to its evolution which is confirmed in ISO/IEC 25010:2011 document: "*This International Standard revises ISO/IEC 9126-1:2001, and incorporates*

[15] Reliability include storability, faultless operation, maintainability and durability ; Published via a Russian regulatory documents (GOSTs) decree [15].

[16] There exists a "magic number" which is 7 ±2" in experimental psychology corresponding to the number of units an individual can handle.

Confidential C

the same software quality characteristics with some amendments.

• *The scope of the quality models has been extended to include computer systems, and quality in use from a system perspective* [...]". TABLE 4 summarizes the results. We remark that the degree of polymorphism which represents the distance (i.e. variety) between the two quality models is clearly showing that they are 67.9% different despite the fact that the other attributes are similar. Therefore, the proofed reality is that is it not a simple extension or evolution but rather a "drastic" evolution that may brings risk if ISO/IEC 9126:2001 was previously used successfully, for example.

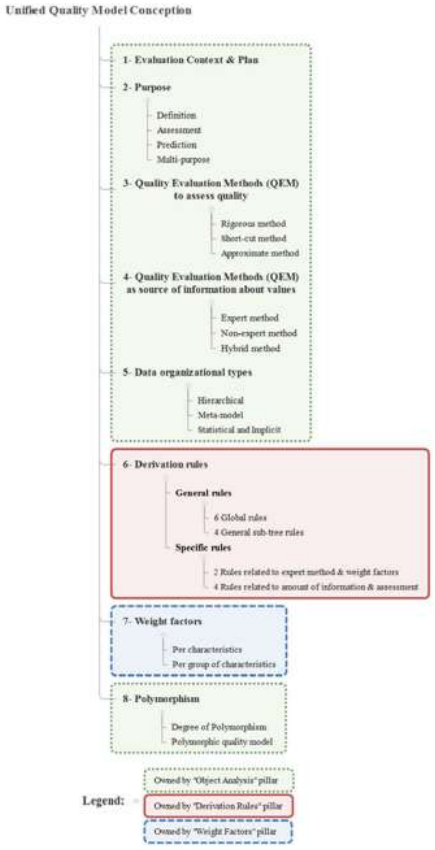

Fig. 10 Our unified conception for Quality Model

To complete our unified characterization proposal, and since two of the quality model purposes are assessment and prediction, we must consolidate it with the measure aspect. Next section deals with this topic.

TABLE 3
COMPARISON OF THE THREE MAIN DISTINCT APPROACHES SUPPORTING QUALITY MODEL SPECIFICATION

	Stream of approach	Wagner [32]	Quality models such as ISO/IEC 250nn [12], [38]–[42], [61]	Azgaldov *et al.* [15]
	Quality model scope	Project and Software product	System & Software product and in use	Any area
#1	Evaluation context & plan	-	Evaluation plan	Evaluation context
#2	Purposes	• Definition • Assessment • Prediction • Multi-purpose	• Definition • Assessment (evaluation part)	• Definition • Assessment
#3	QEM: method to assess quality	Not specified but assumes approximate method	Not specified but assumes approximate method	• Rigorous method • Short-cut method • Approximate method
#4	QEM: source of information about values in QEM	Not specified but assumes expert method	Not specified but assumes expert method	• Expert method • Non-expert method (i.e. analytical method) • Hybrid method
#5	Data organizational types	• Hierarchical • Meta-model • Statistical and Implicit	• Hierarchical (• Meta-model)	• Hierarchical
#6	Derivation rules	-	-	~30 rules
#7	Weight factors	Per property / characteristic	Per property/ characteristic	Per property / characteristic
#8	Polymorphism	-	-	-
	Additional Method	-	-	Assessment algorithm

TABLE 4
OUR 8 QUALITY MODEL ATTRIBUTES EXERCISED VS. SAME STANDARD EVOLUTION: ISO/IEC 9126 & ISO/IEC 25010

	Attribute	ISO/IEC 9126:2001	ISO/IEC 25010:2011
#1	Evaluation context & plan	Information Technology Software product quality & quality in use	System (computer oriented) & Software product quality, quality in use & data quality
#2	Purposes	Definition & Assessment (evaluation)	Definition & Assessment (evaluation)
#3	QEM to assess quality	Short-cut method	Short-cut method
#4	QEM as source of information about values	Hybrid method	Hybrid method
#5	Data organizational types	Hierarchical	Hierarchical (& meta-model)
#6	Derivation rules	Respect of global rules with exception of rule #5 (reliability)	Respect of global rules with exception of rule #5 (reliability)
#7	Weight factors	Not weighted	Not weighted
#8	Degree of polymorphism	0.6792 (0 = identical; 1 = 100% disjoined) [53 leaf characteristics, 32 unique, 8 similar]	

Confidential C

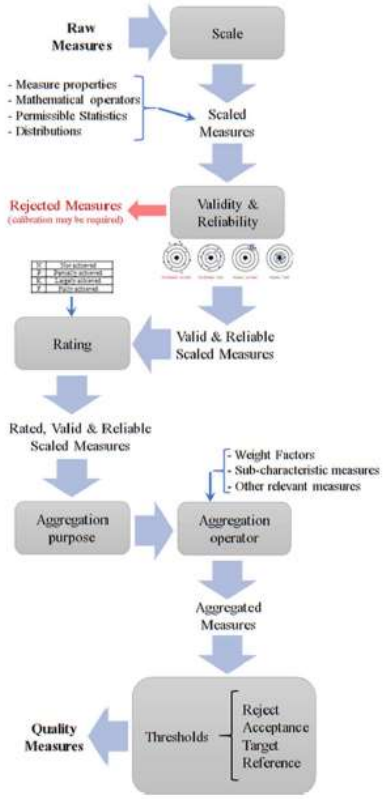

Fig. 11 The measurement refinement process

4. The Measurement Refinement Process

Measurement and metrics are fundamental complements to quality model. Thanks to them, we are in the position where quantification, assessment and quality control are possible. Thus, we start from the quality characteristic elements, in our quality model, that don't have any sub-characteristics called indices[17], and process up to the object to evaluate. We have discussed in section 2.2.B, a certain number of properties or characteristics. Regrettably there isn't any defined process integrating all of them in order to transform a raw measure into a quality measure, in spite of an existing natural order[18] between some transformation operations. Therefore, we propose to fill this gap by introducing a measurement refinement process depicted by Fig. 11.

The first transformation step for a raw measure is to associate it to the right scale, starting from *S. S. Stevens'* scales [47]. This is the most crucial step because this operation allows the measure to be stuck to a mathematical set and operators. We are then in a position where we can operate, process and manipulate the measures. Moreover, at this level, we get the knowledge of permissible statistics and distributions that are appropriate on this measure based on the associated scale.

The next processing task is the assessment of reliability and validity of the measure. Thanks to the scale we have the mathematical tools to evaluate these two key properties. If the measure is neither reliable nor valid, we reject this measure since we consider it as noise[19].

Now that we have a valid & reliable scaled measure, we can start interpreting it by rating it. The rating task, which depends on the type of scale that it is used, consists to give a coarse interpretation of the measurement. TABLE 5 and TABLE 6 are examples of rating values depending on two scale types, nominal and ordinal scales.

TABLE 5
RATING NOMINAL SCALE ACCORDING TO ISO/IEC 33020

Rating	Meaning	Description
N	Not achieved	There is little or no evidence of achievement of the defined process attribute in the assessed process.
P	Partially achieved	There is some evidence of an approach to, and some achievement of, the defined process attribute in the assessed process. Some aspects of achievement of the process attribute may be unpredictable.
L	Largely achieved	There is evidence of a systematic approach to, and significant achievement of, the defined process attribute in the assessed process. Some weaknesses related to this process attribute may exist in the assessed process.
F	Fully achieved	There is evidence of a complete and systematic approach to, and full achievement of, the defined process attribute in the assessed process. No significant weaknesses related to this process attribute exist in the assessed process.

TABLE 6
RATING ORDINAL SCALE IN PERCENTAGE VALUES ACCORDING TO ISO/IEC 33020

Rating	Meaning	Range values
N	Not achieved	0 to ≤ 15% achievement
P	Partially achieved	> 15% to ≤ 50% achievement
L	Largely achieved	> 50% to ≤ 85% achievement
F	Fully achieved	> 85% to ≤ 100% achievement

To continue exploiting and enriching the measure, we aggregate it. For this operation, we have to select the aggregation purpose, that is to say what type of aggregation

[17] In quality model represented via a hierarchical (or tree) structure, the indices are also called the leaves (of the tree).
[18] For example, scale transformation should be done before aggregating measure, aggregation processing should occur before acceptance thresholding.
[19] However, we are not throwing away the measure. We are keeping track of the noisy measure because it may tell us that our measurement system needs to be recalibrated, for instance, or there is a defect that may require our attention.

Confidential C

we want to perform. By definition, aggregation is an operation to combine elements together. According to Wagner [32], purpose can be: assessment, prediction, hot spot identification, comparison and trend analysis. Based on the purpose, we are ready to use aggregation operators[20] which are obviously the real mathematical and statistical tools. We note that at this stage, we may use other measures, as well as the quality model weight factors, as inputs for aggregation operation. Moreover, if we are using "*identity*" aggregation operator, it means that this process task is simply copying our input measure to our output process.

The final task of our treatment process is the carry out of the refined measure against thresholds. Usually two types of threshold are used extensively[21]; however, we can distinguish four ones [15]: reject, acceptance, target and reference. The reject threshold is the inclusive barrier where all quality measures conduct to reject decision. The acceptance one is the worst threshold above the reject threshold and can be renamed as the "good enough" threshold. Starting from this level, the quality measure is at least at the minimum allowed. The target threshold corresponds to the quality measured level that is aimed while the reference threshold is the best reached quality measure in the world and industry, at the moment of the measure is done, for same type of object that is under qualification. We can identify a fifth and last threshold, the "forecasted" which is linked to prediction and enhancement traceability required, for example, by ISO 26262 [68] for safety.

This measurement refinement process we propose, described here above and synthetized in Fig. 11, is a new way to have clear and systematic treatment chain for the measures done jointly with our quality models. It prevents mistakes in measurement manipulation and ensure that critical steps are done at the right time.

On the first steps of the way of our journey of applying qualimetry to embedded software development, we learnt how to define quality *via* object characterization done under the form of quality model and how to proceed consistently on the measures. The remaining part to resolve is to identify what are the objects beneath "*embedded software development*" and how they are articulated together.

5. Our Case Study: The Embedded Software Development

Over sections 2 to 4, our analyses, approaches and contributions were not restricted to software field but rather had a wide focus, applicable to any system. Nevertheless, as we were indicating in our introduction, our case study is targeting embedded software product and therefore we now have to narrow our scope of work around it. This means that the objects of interest against which we are going to apply qualimetry are not only the product by itself, but also the software development life cycle (cf. Fig. 4), which covers its specification, development and maintenance states, and the software development organization that realizes it. So, in the next sub-sections we study first what are the objects of interest beneath embedded software development life cycle and product, completing secondly with some consideration to software organization. Then, in last sub-sections, we evoke the question related to which quality models, if any, should be applied to our objects of interest.

5.1 Embedded Software Development Life Cycle and Product

Before jumping into software development life cycle concept, let's first review the fundamental piece that we are aiming to build: the embedded software product.

By referring to IEEE glossary [24], embedded software is simply defined as a software which is part of a larger system and performs a sub-set of that system requirements. In addition, embedded software is often an assembly of software modules and, or components, themselves composed of software units. We recall also that a system has three dimensions - physical, computational (or logical) and human - and every system is a combination of these three dimensions [69]. Thus, we understand that an embedded software product is characterized, or described, by its specifications, expressed as functional and non-functional requirements, coming from the three dimensions of the system it is part of. The realization of such product is done by implementing and fulfilling these specifications within a dedicated project [53].

Furthermore, we understand that our embedded software is favorable to change or evolution. The cause is linked to the fact that embedded software is part of a system defined by three types of components (i.e. physical, computational and human) subject themselves to change or evolution due to real world interaction and execution environment. So, we can state that our embedded software product is subject to continuous evolution once the first product version is released.

To go one step further, we would like to refer to the work done by Lehman on software evolution and his software program, or system, categorization [70], [71]. Lehman's three types, S-, P- and E-, are summarized *via* TABLE 7. Obviously, we can associate the system which encompasses embedded software product, to Lehman's E-type system: systems that operate in and resolve a problem of real world. Such systems are evolutionary oriented. As opposed to S-type and P-type system, the height Lehman' software evolution laws [71] (cf. TABLE 8) are applicable to that system. Even if there may be some limitations on their direct applicability, they are useful guidelines with regard to the software development life cycle.

So, the System Development Life Cycle reflects all the activities from system initiation up to its disposal. Morever, one of its subsets focus on the software product: this is the Software Development Life Cycle illustrated in Fig. 13 that the authors built based on ISO/IEC/IEEE 15288:2015 [53], ISO/IEC/IEEE 12207:2017 [55] and the System Engineering handbook [72].

[20] We can cite arithmetic or weighted mean, k-mean, median, minimum and maximum, variance and standard deviation, t-norms and t-conorms...We invite to refer to Detyniecki [52] for an introduction to a variety of aggregation operators.

[21] These are generally acceptance and target thresolds

Confidential C

TABLE 7
THE THREE LEHMAN'SYSTEM OR PROGRAMS TYPES ORDERED BY COMPLEXITY

Type name	Scope	Examples
S-type or S-program	Rigorous and static specification oriented	Calculator, DVD player
P-type or P-program	Practical real-world problem solution oriented	Chess program
E-Type or E-Program	Real world evolutionary oriented	Smartphone, Automotive or aeronautic system

TABLE 8
THE EIGHT SOFTWARE EVOLUTION LAWS OF LEHMAN

Law Number	Year of creation	Name
I	1974	Continuing Change
II	1974	Increasing Complexity
III	1974	Self-Regulation
IV	1980	Conservation of Organizational Stability (invariant work rate)
V	1980	Conservation of Familiarity
VI	1980	Continuing Growth
VII	1996	Declining Quality
VIII	1996	Feedback System (first stated in 1974 and formalized as law in 1996)

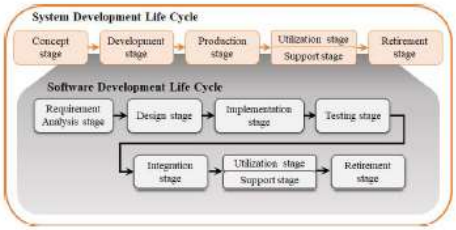

Fig. 13 Software Development Life Cycle is part of System Development Life Cycle, based on typical phases.

Both system and software life cycles are a suite of more or less distinct consecutive phases, implemented themselves over 14 technical processes [53] and where some of them can be performed in parallel. A typical life cycle, shown in (see Fig. 4) is provided by the ISO/IEC/IEEE 15288:2015 [53]. As seen in section 2.2.C, there is another model fitting both cases and not antinomic to the typical one. This model integrates better the system, or software, "evolution" phases and relies on the level of conservation of familiarity, for instance. This is the staged model [57], [58] illustrated in Fig. 5 where we have initial product release versus evolution, servicing, phase-out and close down. However, the structuration, behaviors and characterizations of these stages are conditioned by the development methodology[22] that is deriving the life cycle. Among all available methodologies, the ones shown in Fig. 12 are part of the main most different ones, illustrating the variety of available solutions: waterfall [73][23][74], V-model [76], spiral [77] and agile [78].

[22] We remarked that development methodology concept is assimilated to System Development Life Cycle in INCOSE Handbook [72] which is not case of ISO/IEC/IEEE 15288:2015 [53].

[23] We would like to raise that even if Waterfall development process origin may be linked to W.W. Royce [73], the first use of the term was done by T. E. Bell and T. A. Thayer [74]; note that some lectures of the original paper bring another interpretation of it, as [75].

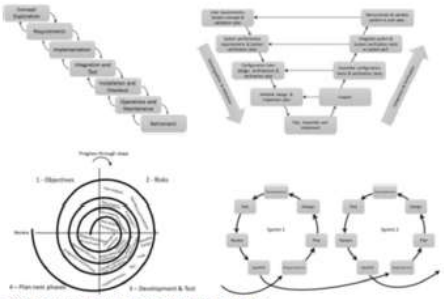

Fig. 12 Development methodology samples
. From top to bottom & right to left: waterfall, V-model, Spiral, Agile

To summarize, we identified several kinds of objects against which we need to apply qualimetry (see Fig. 14 for the complete map of our Objects of Interest).

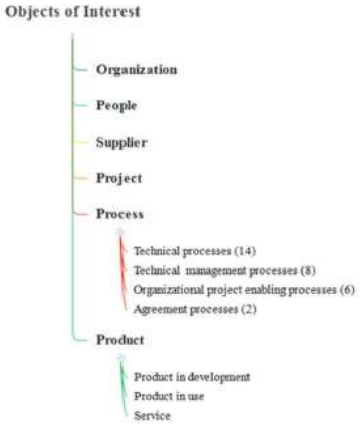

Fig. 14 Our Objects of Interest to be applied against qualimetry

First, accordingly to software development life cycle, we have the product, and more precisely the embedded software product, that we split into *product in development* (or internal product stage) and *product in use* [24] (or external product stage). These two product types correspond to the two distinct active states in the product life cycle. Moreover, we note that the *product in use* is a released product which is impacted by evolution and feedback: Lehman's law 3 "self-regulation" and 8 "feedback system". It is obvious that we have interest to quantify the quality of our product not only during its conception and development phase, but also during its use or life. Then, since by definition a product is the result of a process of an organization [22], we must pay attention to that specific "process" which is the *project*.

[24] We apply to software product the same decomposition that is defined in ISO/IEC 25010 [12] : *software quality* against *quality in use*.

Confidential C

Finally, to conduct all these project and development activities, despite the methodologies that are applied, we are relying on a "*set of interrelated or interacting activities that transforms inputs into outputs*" [53], or *process*; and therefore we must have a strong control on them. ISO/IEC/IEEE 15288:2015 [53] is helping us for their identification: the software development lifecycle processes are defined by 14 technical processes and the project by 8 technical management processes.

5.2 Software Organization

Section 5.1 reviewed two types of products that need our attention and the central parts that are project and process to generate them. However, we missed a fundamental aspect from our system, which makes product, project and process useless: this is the human dimension, as indicated also in this section 5.1.

Indeed, people are the mandatory and elementary brick that empowers the organization, makes possible innovation, creation and delivery of products. Applying qualimetry on people ensures that we balance and optimize the use of this critical resource. But this is not a trivial task and requires respect of ethic and data privacy. A good example is the new European law on "*General Data Protection Regulation*" [79] showing some of the strengthened limits that a company must avoid to cross: this regulation European law is acting on data protection and privacy for all individuals, asking companies to have business process in place to manipulate these sensitive data. An example of protection here can be the pseudo or full anonymization of these data.

Moreover, people, all together with facilities, are the essence of the organization [53] that we ultimately need to quantify in term of quality. In our case, we consider the software organization because we are dealing with embedded software product, but the approach is identical for any type of systems. Again, likes project and software development life cycle, we have some dedicated set of processes [53] that are helping the execution here. These are two agreement processes and 6 organizational project-enabling processes, show in Fig. 14 and Fig. 15.

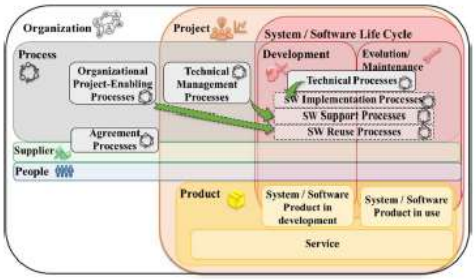

Fig. 15 Our objects of interest for qualimetry: organization, process[25], project, people, supplier, service and software product.

[25] In this figure, green arrows highlight to wich the process set the SW specific processes are belonging to since ISO/IEC/IEEE 12207:2017 [54] in 2017.

To complete our overview of the objects to assess and control, we have to introduce two additional objects. The first one corresponds to another type of product that the organization delivers to customers: this is the service [22]. It can be either produced standalone of jointly with a software product, increasing the global value delivered to the external or internal customer. The last object is the supplier, or provider [22], [80] which is responsible for some deliveries to the organization and therefore may have serious impact to it. To handle suppliers, we have two agreement processes defined [53]: acquisition process and supply process.

In conclusion, our analysis resulted in the identification of six main objects or areas that we must assess and control using qualimetry science: organization, people, supplier, process, project and product. Our proposal is close to what can be found in the standard such as Automotive-SPICE [80][26], with MAN.6 process (i.e. Measurement process) but with two exceptions: we consider risk as part of project, that choice being confirmed by ISO/IEC/IEEE 15288:2015 [53], and there is no counterpart of our organization object in A-SPICE. TABLE 9 is summarizing this comparison while Fig. 15 is a visual recap of our objects of qualimetry interest enumeration with respect to each other (i.e. reflecting dependencies).

TABLE 9
AREAS OR OBJECTS TO BE MEASURED: COMPARISON BETWEEN OUR ANALYSIS AND AUTOMOTIVE SPICE MAN.6

Id	Our objects to assess and control		A-SPICE MAN.6
1	Organization		-
2	People		Personnel Performance
3	Supplier		(*Managed in A-Spice ACQ.4 process: Supplier monitoring*):
4	Process (includes 30 processes)		Process
5	Project		Project Risk
6	Product	Software product in development	Quality
		Software product in use	Field
		Service	Service

5.3 Which Quality Model(s) to apply to our Objects?

From the last subsections 5.1 and 5.2, it obviously appears that for each of the eight objects that we highlighted, we will need not only to determine right quality definition models but also, to extend their respective definition quality model with the corresponding assessment quality model because object quality properties and measurements may be different.

Regarding product, and most particularly the software product[27], since 1977 and the publication of one of the first software product quality model by McCall [35] (cf. Fig. 8), myriads of software quality models were regularly defined

[26] We choose to compare our results to A-SPICE because this standard is focusing on automotive software which is embedded software likes our scope.

[27] We recall having also service as a product (see section 5.2).

Confidential C

or derived (i.e. tailored) from others, including some effort of normalization with IEEE model [81] in 1992, SIE model [82] in 1995, ISO/9126 [11] in 2001 and ISO/IEC 25010 (i.e. SQuARE) [12] in 2011. Concerning tailoring of these types of model quality, Miguel *et al.* [83] confirmed than prior to 2000, efforts were put on the study and creation of such models, and since year 2000, investments have been increasingly set to reuse and tailor those set of models. Then based on the large number of available models[28] and their comparison studies [84]–[87] that don't conclude on any best "solution", we state that current situation is not satisfying.

Furthermore, even if we take the latest version of software product quality model standard ISO/IEC 25010 [12] (cf. Fig. 6), we reach the same conclusion: in this quality model we have *reliability* characteristic while general tree derivation rule 5 (cf. TABLE 1) states that *reliability* must be excluded since it is part of cost effectiveness, itself part of integral quality (see section 2.1). So, at least, we have to exclude this characteristic and its sub-characteristics, and therefore alter this quality model.

By consequence, the solution is to proceed first to a classification of existing software quality models based on our unified conception proposal, introduced in section 3, as categorization landmarks. Then, decomposed our three types of product into multi-levels of characteristics, properties and indices. Finally, map each of our object decompositions to the model which has the closest distance to it[29]. We may be in the situation where this distance is too far (for instance, cases where at best less than 50% of characteristics are matching), and, for that reason tailoring won't be sufficient: a new model will be necessary. We remark after all, if this is not defined in theoretical qualimetry, this object mapping into quality model is similar to projecting this object into quality space.

For our remaining objects that are project, process, people, supplier and organization, there are only partial solutions, such as, for example, A-SPICE [80], CMM [88], COCOMO[30][89]. The two first ones are giving some guidelines about object attributes that have interest in their scope. For the last one, the focus of the software project quality model is put on effort, cost and schedule estimation. Obviously, we can find other characteristics that must be took into account likes productivity, defects, estimations, technical management process efficiency... Even so no direct quality models are suitable, we have identified that Software Process Improvement (i.e. SIP) have the fundamentals - confirmed by Unterkalmsteiner *et al.* studies [90], [91] - for designing the right quality model for each our objects[31].

In any case, we proved that the answer to our question on "*which quality models to apply to our objects?*" is not obvious and require a more intensive effort of inventory, classification, object of interest characteristic identification, quality model assessment and most probably tailoring. However, with our unified conception (section 3), measurement refinement process (section 4) and the 8 identified objects of interest (section 5) we have now all elements to proceed accurately on setting the different quality models to then assess and control their quality.

6. CONCLUSION

This paper describes the first part of our journey of applying qualimetry to embedded software development: the planification of this journey.

Indeed, through a brief study and discussion, we rationalized the concept about quality and qualimetry, inventing a synthesized view of the ideas behind the quality quantification science *via* the schema "*the house of qualimetry and its pillars*".

With a subsequent discussion, we went further by clarifying "*embedded software development*" notion and explained how qualimetry is applied to it, revealing the existence of several road blockers that we removed over five new and unique contributions.

First, we elaborated a unified conception for quality model to guide quality model design, classification and assessment. That conception relies on height attributes including our proposal to add polymorphism concept to quality model, encapsulating the idea of quality model implementation can be of different forms and the fact that many existing quality models may be a closer form to some other ones. This last aspect is evaluated following our proposal to apply polymorphism degree concept, coming from biology (i.e. genetic) field, to quality model conception and classification. This measure is used successfully in genetic and its usage in qualimetry is indubitably useful.

In addition, our measurement refinement process, rationalizes the multiple transformation sequences that converts raw measures into quality measures. And last, we proposed an "*embedded software development*" characterization over a height objects of interest map. That contribution is resulting of a study related to objects linked to embedded software development, their evolution and a comparison against A-SPICE standard which shares the same boundary. This contribution is not apart to our other ones because its appropriateness domain is finally also system, even if we considered embedded software case.

In conclusion, we made six contributions that can be applied to any type of system, such as the variety of our study sources (i.e. socioeconomics, biology, system and software engineering), without limitation. This paper develops the first part of our journey and certainly one of the most critical one since we setup our foundations. Next part is proceeding on quality model classification, selection and then tailoring, or creation, for all our 8 objects of interests. To achieve this task, we will rely, for example, on polymorphism concept and measure to support our investigation and decisions. Obviously, our contributions are opening the field of new perspectives and solutions.

[28] Proper query to portal.acm.org, www.ieeexplore.org, www.scopus.com, isiknowledge.com or link.springer.com, as used also in [84], gives us more than 44 distinct software product quality models.

[29] Polymorphism will be of a great help by providing a measurement with the degree of polymorphism for a quality model computed against the other candidate ones.

[30] The acronym stands for COnstructive COst MOdel which deals with cost evaluation of software engineering.

[31] All our objects are under SIP radar and we recall that standard such as CMM and A-SPICE, as we were previously referring to, have their highest maturity level closely linked to process improvement.

Confidential C

REFERENCES

[1] "Enquête Nationale : les Coûts de la Non-Qualité dans l'Industrie." Afnor Group (in French), Oct-2017.

[2] R. Black, *Critical Testing Processes*. Addison-Wesley, 2003.

[3] N. G. Leveson and C. S. Turner, "An Investigation of the Therac-25 Accidents," *Computer*, vol. 26, no. 7, pp. 18–41, 1993.

[4] R. L. Baber, "The Ariane 5 explosion: a software engineer's view," *Risks*, vol. 18, no. 89, Mar. 1997.

[5] J. Oberg, "Why the Mars probe went off course," *IEEE Spectrum*, vol. 36, no. 12, pp. 34–39, Dec. 1999.

[6] S. Gandel, "Why Knight lost $440 million in 45 minutes," *Fortune*, 03-Aug-2012.

[7] Reuters, "Takata's U.S. Unit Reaches Deal Paving Way for Sale.," *The New York Times*, 12-Feb-2018.

[8] G. G. Azgaldov *et al.*, "Qualimetry: the Science of Product Quality Assessment," *Standart y i kachest vo*, no. 1, 1968.

[9] G. G. Azgaldov, "Development of the theoretical basis of qualimetry," doctoral dissertation, Kuibyshev Military Engineering Academy, Moscow (in Russian), 1981.

[10] G. G. Azgaldov, *The Theory and Practice of Product Quality Assessment. Essentials of Qualimetry*, Moscow: Ekonomika (in Russian). 1982.

[11] "ISO/IEC 9126-1:2001 - Software engineering - Product quality - Part1: Quality Model," *International Organization for Standardization*, 2001.

[12] "ISO/IEC 25010:2011 - Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models," *International Organization for Standardization*, 2011.

[13] "Google Trends on 'Quality,'" 2018. [Online]. Available: https://trends.google.com/trends/explore?date=all&q=Quality.

[14] P. Antman, "From Aristotle to Descartes a Brief history of quality." [Online]. Available: https://blog.smartbear.com/software-quality/from-aristotle-to-descartes-a-brief-history-of-quality/.

[15] G. Azgaldov, A. Kostin, and A. Padilla Omiste, *The ABC of Qualimetry, toolkit for measuring the immeasurable*, Ridero. 2015.

[16] G. P. Stavropoulos, *The Complete Aristotle*. Free GPS Library, 2013.

[17] "Online Oxford Dictionary - quality definition," 2018. [Online]. Available: https://en.oxforddictionaries.com/definition/quality.

[18] W. E. Deming, "Out of the crisis: quality, productivity and competitive position," 1988.

[19] B. Kitchenham and S. Pfleeger, "Software quality: the elusive target," *IEEE Software*, vol. 13, no. 1, pp. 12–21, 1996.

[20] D. A. Garvin, "Managing Quality - the strategic and competitive edge," in *New York, NY: Free Press [u.a.]*, 1988.

[21] R. M. Pirsig, *Zen and the art of motorcycle maintenance : an inquiry into values*, New York, N.Y.: Morrow. 1974.

[22] "ISO/IEC 9000:2015 - Quality management systems - Fundamentals and vocabulary," *International Organization for Standardization*, 2015.

[23] "ISTQB glossary 3.1," *https://www.istqb.org/downloads/category/20-istqb-glossary.html.*.

[24] "IEEE Standard Glossary of Software Engineering Terminology, IEEE Std 610.12-1990." Institute of Electrical and Electronic Engineers, Inc., New York, NY, 10-Dec-1990.

[25] A. S. Lobanov, "The Basic Concepts of Qualimetry," *Scientific and Technical In formation Processing*, vol. 40, no. 2, pp. 72–82, 2013.

[26] J. A. Diez, "A Hundred Years of Numbers. An Historical Introduction to Measurement Theory 1887-1990 Part I: The Formation Period. Two Lines of Research: Axiomatics and Real Morphisms, Scales and Invariance," *Studies in History and Philosophy of Science*, vol. 28, no. 1, pp. 167–185, 1997.

[27] P. A. Florenskii, "Some Remarks on Product Quality Assessment," *Vestn. teor. eksperiment. elektrotekhniki*, no. 11, 1928.

[28] G. G. Azgaldov and A. V. Kostin, "Applied qualimetry: its origins, errors and misconceptions," *Benchmarking: An International Journal*, vol. 18, no. 3, pp. 428–444, 2011.

[29] Y. Argotti, C. Baron, and P. Esteban, "Quality quantification in Systems Engineering from the Qualimetry Eye," presented at the 13th Annual IEEE International Systems Conference (SysCon), Orlando, USA, 2019.

[30] J. R. Hauser and D. Clausing, "The House of Quality," *Harvard Business Review. Archived from the original on April 16, 2016.*, May 1988.

[31] F. Deissenboeck, E. Juergens, K. Lochmann, and S. Wagner, "Software quality models: Purposes, usage scenarios and requirements," in *Proceedings of the 7th International Workshop on Software Quality (WoSQ '09)*, 2009.

[32] S. Wagner, *Software Product Quality Control*, Springer-Verlag Berlin Heidelberg. 2013.

[33] R. J. Rubey and R. D. Hartwick, "Quantitative measurement of program quality," in *Proceedings of the 1968 23rd ACM national conference (ACM '68)*, New York, NY, USA, 1968, pp. 671–677.

[34] B. W. Boehm, J. R. Brown, and M. Lipow, "Quantitative Evaluation of Software Quality," in *Proceedings of the 2nd international conference on Software engineering (ICSE '76)*, Los Alamitos, CA, USA, 1976, pp. 592–605.

[35] J. A. McCall, P. K. Richards, and G. F. Walters, "Factors in Software Quality," *Griffiths Air Force Base, N.Y. Rome Air Development Center Air Force Systems Command*, 1977.

[36] V. Basili, G. Caldiera, and H. D. Rombach, "Goal Question Metric Approach," *Encyclopedia of Software Engineering, John Wiley & Sons, Inc.*, pp. 528–532, 1994.

[37] D. Coleman, B. Lowther, and P. Oman, "The application of software maintainability models in industrial software systems," in *In Selected papers of the sixth annual Oregon workshop on Software metrics*, New York, NY, USA, 1995, pp. 3–16.

[38] "ISO/IEC 25020:2007 - Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Measurement reference model and guide," *International Organization for Standardization*, 2007.

[39] "ISO/IEC 25021:2012 - Systems and software engineering - System and software product Quality Requirements and Evaluation (SQuaRE) - Quality measure elements," *International Organization for Standardization*, 2012.

[40] "ISO/IEC 25022:2016 - Systems and software engineering - Systems and software product Quality Requirements and Evaluation (SQuaRE) - Measurement of internal quality," *International Organization for Standardization*, 2016.

[41] "ISO/IEC 25023:2016 - Systems and software engineering - System and software product Quality Requirements and Evaluation (SQuaRE) - Measurement of system and software product quality," *International Organization for Standardization*, 2016.

Confidential C

[42] "ISO/IEC 25024:2015 - Systems and Software engineering - Systems and Software product Quality Requirements and Evaluation (SQuaRE) - Measurement of data quality," *International Organization for Standardization*, 2015.

[43] D. J. Hand, "Statistics and the Theory of Measurement," *Journal of the Royal Statistical Society*, no. 159, pp. 445–492, 1996.

[44] H. von Helmholtz, *Epistemological Writings, The Paul Hertz/Moritz Schlick centenary edition of 1921, with notes and commentary by the editors, Chapter 3: Numbering and Measuring from an Epistemological Viewpoint*, vol. 79. 1977.

[45] P. W. Bridgman, *The Logic of Modern Physics*. New York : Macmillan, 1927.

[46] J. A. Diez, "A Hundred Years of Numbers. An Historical Introduction to Measurement Theory 1887-1990 Part II: Suppes and the Mature Theory and Uniqueness Representation," *Studies in History and Philosophy of Science*, vol. 28, no. 2, pp. 237–265, 1997.

[47] S. S. Stevens, "On the Theory of Scales of Measurement," *Science, New Series*, vol. 103, no. 2687, pp. 677–680, Jun. 1946.

[48] P. Velleman and L. Wilkinson, "Nominal, Ordinal, Interval, and Ratio Typologies are Misleading," *The American Statistician*, no. 47:1, pp. 65–72, 1993.

[49] A. C. de Souza, N. M. C. Alexandre, and E. de Brito Guirardello, "Psychometric properties in instruments evaluation of reliability and validity," in *Epidemiol. Serv. Saude*, Brasília, 2017, vol. 26.

[50] L. B. Mokkink *et al.*, "The COSMIN study reached international consensus on taxonomy, terminology, and definitions of measurement properties for health-related patient-reported outcomes," *Journal of Clinical Epidemiology*, vol. 63, pp. 733–745, 2010.

[51] "ISO/IEC 25030:2007 - Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Quality requirements," *International Organization for Standardization*, 2007.

[52] M. Detyniecki, "Fundamentals on Aggregation Operators," Computer Science Division University of California, Berkeley United Sates of America, 2001.

[53] "ISO/IEC/IEEE 15288:2015 -Systems and software engineering -- System life cycle processes," *International Organization for Standardization*, 2015.

[54] "ISO/IEC/IEEE 12207:2017 -- Systems and software engineering - Software life cycle processes," *International Organization for Standardization*, 2017.

[55] "ISO/IEC 12207:2008 -- Systems and software engineering - Software life cycle processes," *International Organization for Standardization*, 2008.

[56] "ISO/IEC/IEEE 24748-1:2018 - Systems and software engineering - Life cycle management - Part 1: Guide for life cycle management," *International Organization for Standardization*, 2018.

[57] K. H. Bennet and V. T. Rajlich, "The staged model of the software lifecycle: A new perspective on software evolution," pp. 1–14, 2007.

[58] E. Arch, "Lehman's Laws of Software Evolution and the Staged-Model," 2011. [Online]. Available: https://blogs.msdn.microsoft.com/karchworld_identity/2011/04/01/lehmans-laws-of-software-evolution-and-the-staged-model/.

[59] "ISO/CEI 25001:2014 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Planning and management," *International Organization for Standardization*, 2014.

[60] "Online Oxford Dictionary - model definition," 2018. [Online]. Available: https://en.oxforddictionaries.com/definition/model.

[61] "ISO/IEC 25040:2011 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) -- Evaluation process," *International Organization for Standardization*, 2011.

[62] H. Iqbal and M. Babar, "An Approach for Analyzing ISO / IEC 25010 Product Quality Requirements based on Fuzzy Logic and Likert Scale for Decision Support Systems," *(IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 7, no. 12, pp. 245–260, 2016.

[63] B. A. Kitchenham and J. G. Walker, "A quantitative approach to monitoring software development," *Software Engineering Journal*, vol. 4, no. 1, p. pp 2-13, 1989.

[64] S. Wagner, "A Bayesian Network Approach to Assess and Predict Software Quality Using Activity-Based Quality Models," in *Proceedings of the 5th International Conference on Predictor Models in Software Engineering : PROMISE '09*, Vancouver, BC, Canada, 2009.

[65] M. Neil and N. Fenton, "Predicting Software Quality using Bayesian Belief Networks," *In Proceedings of 21st Annual Software Engineering, Workshop NASA/Goddard Space Flight Centre*, Dec. 1996.

[66] C. Strachey, "Fundamental Concepts in Programming Languages," *Higher-Order and Symbolic Computation*, no. 13, p. pp 11-49, 2000.

[67] M. Nei and W.-H. Li, "Mathematical model for studying genetic variation in terms of restriction endonucleases," in *In Proceedings of the National Academy of Science of the USA*, 1979, vol. 76, pp. 5269–5273.

[68] "ISO 26262-6:2011 - Road vehicles - Functional safety - Part 6: Product development at the software level," *International Organization for Standardization*, 2011.

[69] A. Pyster and R. Adcock, "Report of the Workshop on the relationship between Systems Engineering and Software Engineering," Stevens Institute of Technology, Cranfield University, SERC and INCOSE, Hoboken, New Jersey, Jun. 2014.

[70] M. M. Lehman, "Programs, Life Cycles, and Laws of Software Evolution," *In Proceedings of the IEEE*, vol. 68, no. 8, pp. 1060–1076, Sep. 1980.

[71] M. . M. Lehman, J. F. Ramil, P. D. Wernick, D. E. Perry, and W. M. Turski, "Metrics and Laws of software Evolution - The nineties View, journal," in *Proceedings of the 4th International Symposium on Software Metrics, IEEE*, 1997, pp. 20–32.

[72] D. D. Walden, G. J. Roedler, K. J. Forsberg, D. R. Hamelin, and T. M. Shortell, *Systems Engineering Handbook: a Guide for System Life Cycle Processes and Activities*, Fourth. Wiley, 2015.

[73] W. W. Royce, "Managing the Development of Large Software Systems:," *In Proceedings of IEEE WESCON*, Aug. 1970.

[74] T. E. Bell and T. A. Thayer, "Software requirements: Are they really a problem?," in *Proceedings of the 2nd international conference on Software engineering. IEEE Computer Society Press*, 1976, pp. 61–68.

[75] Pragtob, "Why Waterfall was a big misunderstanding from the beginning – reading the original paper," 02-Mar-2012. [Online]. Available: https://pragtob.wordpress.com/2012/03/02/why-waterfall-was-a-big-misunderstanding-from-the-beginning-reading-the-original-paper/.

Confidential C

[76] K. Forsberg and H. Mooz, "The Relationship of Systems Engineering to the Project Cycle," *First Annual Symposium of the National Council On Systems Engineering (NCOSE)*, Oct. 1991.

[77] B. W. Boehm, "A Sipral Model of Software Development and Enhancement," in *IEEE Computer*, 1988, vol. 21, pp. 61–72.

[78] K. Beck *et al.*, "Manifesto for Agile Software Development," 2001. [Online]. Available: http://agilemanifesto.org/.

[79] European Parliament, "Regulation (EU) 2016/679 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)," 27-Apr-2016. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32016R0679.

[80] VDA QMC Working Group 13 / Automotive SIG, "Automotive SPICE Process Assessment / Reference Model., version 3.1 - revision 656." 01-Nov-2017.

[81] IEEE Std 1061-1998, "textitIEEE Standard for a Software Quality Metrics Methodology, R2009, Revision of IEEE Std 1061." 1992.

[82] M. Barbacci, M. H. Klein, T. A. Longstaff, and C. B. Weinstock, "Quality Attributes," Carnegie Mellon University, Technical Report (CMU/SEI-95-TR-021), ESC-TR-95-021, 1995.

[83] J. P. Miguel, D. Mauricio, and G. Rodríguez, "A Review of Software Quality Models for the Evaluation of Software Products," *International Journal of Software Engineering & Applications (IJSEA)*, vol. 5, no. 6, pp. 31–53, Nov. 2014.

[84] Meng Yan, Xin Xia, Xiaohong Zhang, Ling Xu, and Dan Yang, "A Systematic Mapping Study of Quality Assessment Models for Software Products," presented at the 2017 International Conference on Software Analysis, Testing and Evolution (SATE), Harbin, 2017, pp. 63–71.

[85] Y. Boukouchi, A. Marzak, H. Benlahmer, and H. Moutachaouik, "Comparative Study of Software Quality Models," *International Journal of Computer Science Issues (IJCSI)*, vol. 10 (6), no. 1, Nov. 2013.

[86] S. Manoj Wadhwa, "A Comparative Study of Software Quality Models," *International Journal of Computer Science and Information Technologies (IJCSIT)*, vol. 5 (4), pp. 5634–5638, 2014.

[87] M. Moronge Abiud and P. Mbugua, "An analytical comparative analysis of the software quality models for software quality engineering," *Comprehensive Research Journal of Management and Business Studies (CRJMBS)*, vol. 1 (2), pp. 15–24, Oct. 2016.

[88] M. C. Paulk, B. Curtis, and M. B. Chrissis, "Capability maturity model, version 1.1," *IEEE Software*, vol. 10, no. 4, Jul. 1993.

[89] B. W. Boehm, "Software Engineering Economics.," *IEEE Transactions on Software Engineering SE-10*, pp. 4–21, 1984.

[90] M. Unterkalmsteiner, T. Gorschek, A. L. M. M. Islam, C. K. Cheng, R. B. Permadi, and R. Feldt, "Evaluation and Measurement of Software Process Improvement—A Systematic Literature Review," *IEEE Transactions on Software Engineering*, vol. 38, no. 2, pp. 398–424, Apr. 2012.

[91] M. Unterkalmsteiner, T. Gorschek, A. L. M. M. Islam, C. K. Cheng, R. B. Permadi, and R. Feldt, "A conceptual framework for SPI evaluation," *Journal of Software Maintenance and Evolution: Research and Practice*, 2013.

Yann Argotti Mr Yann ARGOTTI graduated in Computer Science and Electrical Engineering from the French Grande Ecole *ESIEA* in 1996 and received the postgraduate diploma in Fundamental and Applied Computer Science from Paris-Est / Marne-la-Vallée University (France) in 1997; he is currently a PhD Candidate in Computer science and Embedded Systems at the University of Toulouse (France), doing his research at LAAS-CNRS laboratory. Since 1996 he accumulated experience over a wide range of positions and technologies. He joined several major companies, such as Intel and Renault Software Labs -the "*Renault-Nissan-Mitsubishi*" Alliance Software Center- where he is currently acting as a specialist in qualimetry. His fields of interest are on qualimetry, software development and evolution of embedded systems with an application to automotive field. He authored international articles and is an INCOSE member.

Claude Baron. Pr Claude BARON is Full Professor at the University of Toulouse (France). She conducts research in systems and software engineering at the LAAS-CNRS laboratory where she leads the Systems Engineering and Integration team. She is interested in optimizing and smoothing product development process. Her research work is based on a multidisciplinary and collaborative vision of the design of complex systems. She addresses system modeling and process monitoring, considering embedded and critical systems and applications in avionics and automotive. She (co)authored books and international journal articles and received awards for her results. She is an INCOSE and IEEE member.

Philippe Esteban. Dr Philippe ESTEBAN is Associate Professor at the University of Toulouse (France). He received his graduate diploma from the School of Engineers of Tarbes. He is interested in the systems and their commands. He carries out his research activities in Systems Engineering in the "*Systems Engineering and Integration*" team of the Systems Analysis and Architecture Laboratory (LAAS-CNRS). He authored international conference and journal articles. He is an INCOSE member.

Confidential C

C- Conference paper: ERTS Conference 2020

Accepted and then presented on the 30th of January. This paper is available on HAL at: https://hal.laas.fr/hal-02382316. Snapshots are below:

ERTS2020 - Regular Paper- Argotti, Baron,

Quality Quantification Applied to Automotive Embedded Systems and Software

Advances with qualimetry science

Yann Argotti[123], Claude Baron[23], Philippe Esteban[43] and Denis Chaton[1]

Summary — Quality quantification is an unavoidable topic in today daily company life. In this paper, the authors review why quality quantification is critical, what are the main difficulties with the current approaches and highlight the qualimetry approach as the solution. After a state of the art on qualimetry and on quality model concept strengthened with polymorphism, the first steps of their applications to automotive embedded systems and software in Renault are showcased. The results are not only the benefits in quality quantification for complex systems, such as homogeneity, consistency and compatibility, but also the highlighted risks with the changes in versions of quality models in *Automotive SPICE* and how to define a derivable quality model over electronic control units and vehicle.

Keywords — Qualimetry, quality model, polymorphism, metrics, measure, automotive, standards

I. Context and research objectives

A. The need to evaluate and quantify quality

Nowadays Renault is producing automotive systems at a high cadence. These automotive systems are very complex and embed many sub-systems. Evaluating and quantifying the level of quality of a system and of each sub-system is important, for different reasons exposed below.

First, a company such as Renault has to comply with many standards and regulation. This is obvious when we consider transportations systems such as cars or airplanes where we have to follow functional safety standards such as ISO26262 [1], ARP4754A[5] [2] and DO-178C [3]. Therefore, properly quantifying quality will tell us if we fulfill or not those standards.

Moreover, "quality quantification" covers both quality aspects (supporting the identification of the systems main characteristics) and quality models (supporting the organization of these characteristics). Quantification helps optimizing and controlling the large flow of metrics and measurements, and extracting the subset that makes most sense to Renault (or which is the most useful for Renault).

We can certainly find many other good reasons why quality quantification is important. However, missing some steps in quality quantification may sometimes turn into catastrophic scenarios. We can quickly cite a few well-known examples: the issue of software update with Therac-25 causing irradiation and death of 6 patients during 1985-1987 [4], Ariane 5 explosion on its first launch on the 4th of June 1996 [5] due to the reuse of the previous navigation system that was not aligned with the new rocket version velocity and then resulting on the loss of $370 million, on the 26th of June 2017 Takata's bankrupt happened due to an unaddressed known bug in their airbag [6] and on 2018, Toyota recalled 2.4 million hybrid cars because of a failure in the "failsafe" driving mode linked to an uncaught software issue [7]. Through these four examples, we have four different systems with four different quality quantification contexts, and an obvious demonstration that their consequences, measured in term of people loss and / or budget, were catastrophic, thus highlighting the need to have not only a reliable and accurate quality quantification approach, but also adapted to system usage context.

The quality addressed in this paper is the quality of product during its entire life cycle, including development (requirement analysis, design, implementation), maintenance and operation.

---

[1] Renault SW Labs
[2] INSA Toulouse
[3] CNRS - LAAS
[4] Université de Toulouse III, Paul Sabatier

[5] The authors would like to point out that Aerospace Recommend Practice (ARP) is a guideline coming from SAE International, and originally SAE International was initially established as the Society of Automotive Engineers on 1905.

Confidential C

B. Many possibilities but difficulties to find the optimum approach

We understand from above that quality quantification is critical; but depending on which quality quantification approach is used, we may face different types of challenge. The first case we can face is when the solution we are considering to quantify quality is too general and then requires a certain level of refactoring or tailoring without any guarantee to get to the right solution. This is often the case with references or standards like CMMI [8], ISO/IEC9126 [9] or ISO/IEC25010 [10] which have the ambition to cover as many types of systems/domains of application as possible. A study conducted by Wagner *et al.* [11] showed that 79% of companies that use standards customize them. At the opposite, the solution can be too specific and then reuse against another more or less close systems can be hard; this is the case for instance with Factor / Criteria / Metric from McCall *et al.*[12] or with Basili *et al.* [13] and the Goal / Question / Method approaches. A third possibility is that the solution set we have is too large, and by consequence there is not an obvious right solution; for instance, it is the case of software products, for which more than 40 distinct quality models can easily be identified in the literature. Finally, the last case is when we have both theoretical and applied aspects for quality quantification, like Wagner [14] on software product quality control or Azgaldov *et al.*[15], [16] on general quality assessment; these approaches may be a little bit heavier to use but they offer a large potential: they are part of *Qualimetry*.

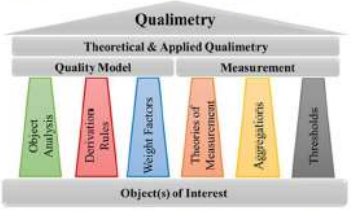

Fig. 1 -The house of qualimetry and its pillars [17]

In the next sections, we complete an overview of the state of the art about qualimetry and quality model concept, strengthening this last one with the innovative introduction of polymorphism. Then we apply these concepts to the automotive domain, encompassing embedded systems and embedded software before concluding on the results, the benefits and the next steps.

II. State of the art on qualimetry and on quality model concept

A. Qualimetry

Qualimetry is the science of quality quantification [16] and consequently covers both "*intellectual and practical activity encompassing the systematic study of [...]*"[18] quality quantification. These two activities are reflected by the theoretical and applied qualimetry. In addition, even if this concept is not recent, this science is relatively young because it was born in former USSR in 1968 [15], following the creation of a working group composed of scientists coming from various areas (e.g. economists, architects, civil engineers, car makers) who were sharing the same goal [16]: generalizing quality quantification approach. In order to foster its understanding and support its knowledge leveraging within quality quantification scope, we have proposed a synthetic representation, the *House of Qualimetry* (see Fig. 1).

By definition, qualimetry is a science, so it addresses both theory and application dimension. These two dimensions themselves rely on two domains: quality model -focusing on the quality characteristics- and measurement -addressing the quantification part. Quality model sits on three pillars. The first one is related to the analysis for identifying quality characteristics while the second one support their organization and add some rules to control analysis depth. The last pillar here is covering characteristic importance among other characteristics. On the measurement side, we retrieve the same topology. The first measurement pillar is key because it is about the different theories of measurement and therefore brings all the mathematical and statistical tools. Second pillar is reflecting the measurement combination or data aggregation together depending on their purpose. Third and final pillar supports the use of measurements during assessment, control and prediction: this is the threshold pillar. Completing the description, the basement of the house is the object(s) of interest that is to say the object(s) candidate for quality quantification. The two ISO/IEC25010:2011 quality models (*i.e.* "*systems/software product quality*" and "*quality in use*") [10] illustrate the result that can be achieved by applying the two quality model first pillars against automotive embedded software product[6].

[6] The reason why we are linking to ISO/IEC 25010 quality model is due to the fact that Automotive SPICE [19] -the process assessment and reference model in automotive field- is referring to that standard for software product quality model.

Confidential C

B. Quality Models

To understand the quality model concept more precisely, we can rely on ISO/IEC IS 9126-1 [9] where a quality model is defined as "*the set of characteristics, and the relationships between them that provides the basis for specifying quality requirements and evaluation*". This definition can be completed by ISO/IEC 25010 [10] which highlights that a quality model is "*convenient breakdown of product quality*", "*serve as a framework to ensure that all characteristics of quality are considered*" and "*provide a set of quality characteristics relevant to a wide range of stakeholders, such as: software developers, system integrators, acquirers, owners, maintainers, contractors, quality assurance and control professionals, and users.*".

To go one step further on the quality model knowledge, we have conducted a study [17] to be able to isolate a pattern related to quality models relative to their design, conception or adaptation. We have identified a set of eight attributes (cf. TABLE 1): six shared between all approaches (*evaluation context & plan, purposes, Quality Evaluation Methods (EQM) to assess quality, QEM as source of information about values, data organizational types* and *weight factors*), one unique, the *derivation rules* coming from qualimetry field and one new, absent from any previous related streams of work: the *polymorphism*. Moreover, the notable fact with these attributes is that, if we handle or consider them sequentially, we land with a unified conception process to create or adapt quality model, starting from "*evaluation context and plan*" up to "*polymorphism*".

TABLE 1 – THE 8 QUALITY MODEL ATTRIBUTES EXERCISED AGAINST SAME STANDARD EVOLUTION: ISO/IEC 9126:2001 & ISO/IEC 25010:2011

	Attribute	ISO/IEC 9126:2001	ISO/IEC 25010:2011
#1	**Evaluation context & plan**	Information Technology Software product quality & quality in use	System (computer oriented) & Software product quality, quality in use & data quality
#2	**Purposes**	Definition & Assessment (evaluation)	Definition & Assessment (evaluation)
#3	**QEM to assess quality**	Short-cut method	Short-cut method
#4	**QEM as source of information about values**	Hybrid method	Hybrid method
#5	**Data organizational types**	Hierarchical	Hierarchical (& meta-model)
#6	**Derivation rules**	Respect of global rules with exception of rule #5 (reliability)	Respect of global rules with exception of rule #5 (reliability)
#7	**Weight factors**	Not weighted	Not weighted
#8	**Degree of polymorphism**	**0.6792** (0 = identical; 1 = 100% disjoined) [53 leaf characteristics, 32 unique, 8 similar]	

C. Polymorphism

With regards to this last attribute, we consider two aspects for polymorphism concept applied to quality model: *ad-hoc* and temporal. To explain what is behind these aspects, we can make an analogy with biology: let us compare a quality model to a butterfly. For the first aspect, we start with a generic butterfly which has a set of characteristics (two wings, a trunk, three pairs of thoracic legs, two antennas ...). In the real world, we have many variants of this generic butterfly that can be more or less close to each other (wing color, pattern and shape, size, lifestyle ...). Each of this variant inherits from the generic butterfly characteristics. Thereby, we can have variants of quality models inheriting from a generic quality model. The second aspect is linked to a temporal consideration. Like the butterfly, starting as an egg, then becomes a caterpillar, chrysalis, then a new born butterfly and comes up with a flying butterfly, a quality model can change, evolve depending of the systems or software product life cycle.

Continuing one step further with biology, and more particularly with genetic, we borrow a formula (1) introduced by Nei and Li in 1979 [20], used to compute the degree of polymorphism between DNA sequences and we apply it to quality model. Thanks to this mechanism we are able to compute distances between quality models from a polymorphism or variety point of view.

$$\pi = \sum_{ij} x_i x_j \pi_{ij} \qquad (1)$$

The π_{ij} coefficients are calculated by considering the probability to have a specific quality characteristic / sub- characteristic. This calculus is based on a pool of quality models. For instance, if a characteristic recurrently appears in those specific quality models, its probability is 1. If half of the time the characteristic is present and the other half is another close (*i.e.* not disjoint) characteristic, then their respective probability is 0.5. When applying this approach to ISO/IEC 9126 and ISO/IEC 25010, we identified: 53 leaf characteristics, 32 unique, 8 similar (*i.e.* close but not identical: for instance, "*Modifiability*" versus "*Changeability*"), and 13 identical. This lands us to find with (1) 67.92% of differences.

Confidential C

On the measurement side, we had to enhance current measurement process to include consideration to the pillars of quality model and measurements. Also, like a processor, we are cadencing the measurement process with the systems or software development life cycle, to integrate the temporal aspect of polymorphism we indicated previously.

To summarize, polymorphism is an help in system engineering where we have a context of systems and/or sub-systems that define a system. Polymorphism brings consistency and support to adaptation due to the context and stakeholder variety.

D. Quality Model distance impacts

The use of polymorphism variety formula is a great tool that help us estimating and explaining what the impacts and consequences are to change, update or adapt current quality model or to apply one quality model instead of another one.

Indeed, the consequences are directly linked to what we aim to do with quality model. For instance, let say that a company was currently using ISO/IEC 9126 and want to be compliant with latest available standard, which is ISO/IEC 25010, then this distance will help to understand and estimate:

- what the risks are linked to such change (low distance = low risk, high distance = high risk),
- what are the areas that are the most impacted (where we have more change, declining the distance for each quality characteristics),
- how much work it will cost,
- where quality quantification, assessment and control are changing,
- how much validation path is changed finding, allowing to capture different types of bugs possibly never found before and discarding other areas and path,

Changes of quality models can occur due to change or evolution of targeted product or stage in its life cycle. Consequently, this may lead in different results and the distance can predict that we may get different results.

In addition, this formula can be used to support decision and to control change or update quality models, including the case of polymorphism: when distance is low, change can be ignored while a high distance tells us that we need to apply this change. Finally, the distance formula can help to split quality model changes into reasonable, from a workload and risk point of view, change increments.

III. Application to automotive

A. With regards to embedded systems

Thanks to this overview of the quality quantification from the qualimetry perspective, we are in a position where we can apply those concepts to the automotive field, thus answering Renault's needs which are: to have a robust, efficient, homogeneous, compatible and consistent quality quantification as well as specify a joint "vocabulary" over the entire complex system that a vehicle is.

Indeed, a car is an instantiation of a vehicle platform which is then addressing a set of car variants such as mini-compact, convertible, super car, cross over, commercial, van.... Therewith, a car is a complex system, composed of more than 40 systems themselves distributed over more than 60 Electronic Control Units (ECU), depending on whether this is a low-end, medium or premium car.

Moreover, besides the fact that each ECU is itself composed of a hardware and an embedded software, an ECU has some common characteristics shared with other ECUs (e.g. diagnostic, connection interface, power), a set of specific characteristics (e.g. HMI, communication, safety) and its own context (e.g. door control, engine control, telematic control, seat control). As a matter of fact, each such subsystem has a vocabulary more or less specific and quality quantification which vary more or less from the other sub-systems. This system complexity description depicts and corresponds to the complexity we have in Renault.

B. With regards to embedded software

Concerning quality model for automotive embedded software, Automotive SPICE[7] [19] advises to rely on ISO/IEC 25010:2011 for embedded software product quality model in its measurement process, called

[7] Renault is relying on Automotive SPICE with regard to its software development activities

Confidential C

MAN.6. However, we remark that in previous versions of Automotive SPICE, such as v2.5 [21], the standard that was referred to for embedded software quality model was ISO/IEC 9126:2001. Moreover, ISO/IEC 25010:2011 standard is the extension, or evolution [10], of ISO/IEC 9126:2001standard. Thus, to see how close or diverse the quality models from these two standards are together, we extract the corresponding pattern instantiations from the height quality model attributes we saw in previous paragraph. TABLE 1 summarizes the comparison results. We notice that most of the attributes are equal except the first one which deals with "*evaluation context and plan*", but this is something that we could expect since ISO/IEC 25010:2011 is an evolution of the other one with a wider scope.

Now if we compute the degree of polymorphism between these two quality models (as seen in II.C), we obtain a result indicating a diversity of almost 68% which means that finally those quality models are quite different. Therefore, this is a drastic evolution, or change despite the fact that in ISO/IEC 25010:2011 document it is claimed this is an extension. Also, upgrading quality model from previous standard to new one can bring risk, particularly if your current quality model works well. Fig. 2 highlights some quality model differences between these two standards.

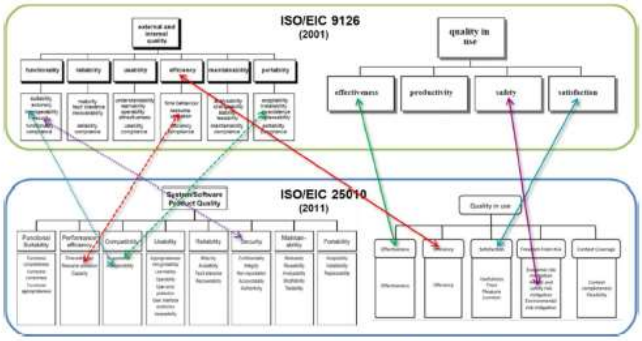

Fig. 2 - Example of some differences between ISO/IEC 9126:2001 & ISO/IEC 25010:2011

C. With regards to polymorphism

In the above sub-sections A and B, we have captured enough knowledge to begin applying polymorphism to automotive embedded software, for instance. In the following paragraphs, we limit our polymorphism application to one level of quality characteristic enumeration[8].

First, we initiate the elaboration of a common ECU quality model using A-SPICE 3.1 guidelines [19]. We note that these guidelines refer also to a subset of ISO/IEC 25010 [10]. The result is a quality model composed of 6 quality characteristics: functional suitability, reliability, usability, performance efficiency, maintainability and portability. All those quality characteristics are aligned with a scope of embedded software. Then, in order to apply polymorphism to this common ECU quality model, we consider two distinct variants, among many, of this common ECU in our study case here: In Vehicle Infotainment (*i.e.* IVI) ECU and Body Control Module (*i.e.* BCM) ECU. The IVI ECU is responsible for infotainment and is the main human user Interface to control different options of the vehicle. In that sense, the performance efficiency is not as important as quality in use aspect which must include efficiency, effectiveness and satisfaction. Indeed, since human-machine interface is key for this ECU, the right performance criteria must be relying on the user perception of the performance rather than pure processing time for instance. Regarding BCM, this ECU can be seen as the main vehicle gateway, dealing with various communication protocols but with no direct interaction with the user (*i.e.* there is. human-machine interface). Then security and safety aspect, included into freedom from risk characteristics, are major quality characteristics to cover for this ECU.

Fig. 3 below illustrates this example of how to apply polymorphism with a subset of quality characteristics from generic ECU quality models to IVI ECU quality model and BCM ECU quality model,

[8] we don't include any sub-characteristics in our example, but we encourage the reader to consolidate current quality models with further sub-characteristics and metrics.

Confidential C

using ISO/IEC 25010 as complement guidelines. In black characteristics defined in Common ECU quality models, in red characteristics that may be discarded/not used, in green complementary characteristics.

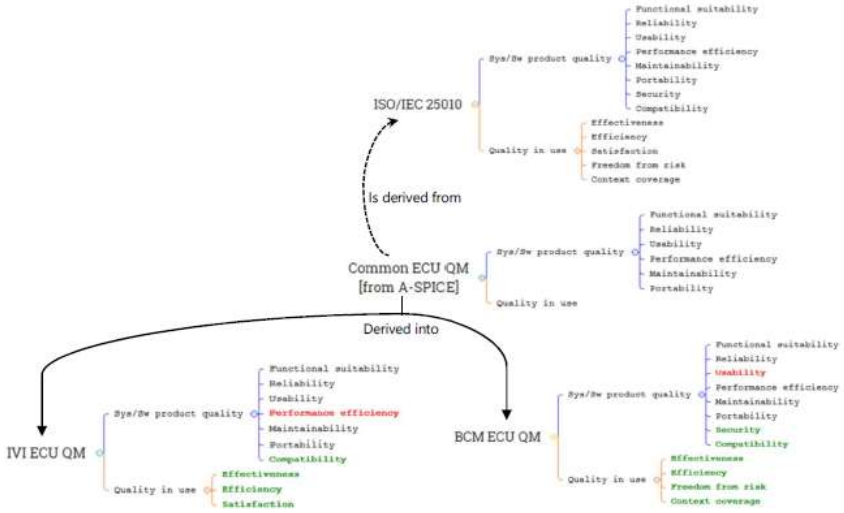

Fig. 3 - Quality model polymorphism example applied to two ECUs

The early results of these concept appliance in Renault are depicted through several dashboards: on code for a subset of non-functional characteristics from ISO/IEC 25010:2011 (see Fig. 4) and with regards to generic quality model for ECU linked applied to their related domain[9] (see Fig. 5).

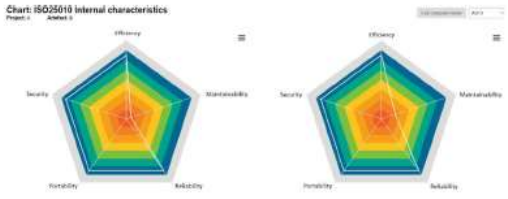

Fig. 4- Example of an evolution of AD ECU code metric results vs a subset of ISO/IEC 25010:2011 characteristics

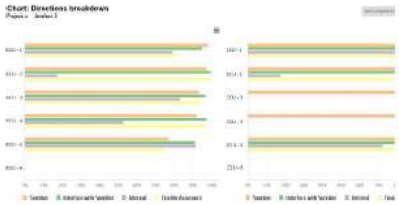

Fig. 5- Example of an evolution of generic quality model result per ECU domain

IV. Conclusion

In conclusion, we have seen what is qualimetry, consolidating some of its aspects, and how it can support generalization, adaptation and repetition in quality quantification activities. It can be as well applied and used for other activities in systems and software engineering, even at the early stage, supporting, for instance, conception, requirement elicitation and architecture design. In addition, our

[9] In Renault, a domain is covered by a specific department or "*direction*".

Confidential C

qualimetry based approach brings homogeneity, consistency and compatibility to quality quantification in the complex environment which is the automotive one. It helps specifying a joint vocabulary where each domain has its own. We also are in a position where we can define a derivable quality model for ECU and vehicle platform thanks to polymorphism. And finally, in a context of agile development methodology, our approach allows a smooth incremental change management.

Our next steps in Renault will focus on the consolidation and then deployment of our current generic quality model as well as the various specific quality models related to the ECU variants.

References

[1] "ISO 26262-6:2011 - Road vehicles - Functional safety - Part 6: Product development at the software level," *International Organization for Standardization*, 2011.
[2] "ARP4754A - Guidelines for Development of Civil Aircraft and Systems," *SAE International*, Dec. 2010.
[3] "DO-178C - Software Considerations in Airborne Systems and Equipment Certification," *Radio Technical Commission for Aeronautics*, Dec. 2011.
[4] N. G. Leveson and C. S. Turner, "An Investigation of the Therac-25 Accidents," *Computer*, vol. 26, no. 7, pp. 18–41, 1993.
[5] R. L. Baber, "The Ariane 5 explosion: a software engineer's view," *Risks*, vol. 18, no. 89, Mar. 1997.
[6] Reuters, "Takata's U.S. Unit Reaches Deal Paving Way for Sale.," *The New York Times*, 12-Feb-2018.
[7] S. McLain, "Toyota Recalls More Than 2 Million Vehicles Over Hybrid-System Fault," *The Wall Street Journal*, 05-Oct-2018.
[8] M. C. Paulk, B. Curtis, and M. B. Chrissis, "Capability maturity model, version 1.1," *IEEE Software*, vol. 10, no. 4, Jul. 1993.
[9] "ISO/IEC 9126-1:2001 - Software engineering - Product quality - Part1: Quality Model," *International Organization for Standardization*, 2001.
[10] "ISO/IEC 25010:2011 - Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models," *International Organization for Standardization*, 2011.
[11] S. Wagner, K. Lochmann, S. Winter, A. Goeb, M. Klás, and S. Nunnenmacher, "Software Quality Models in Practice: Survey Results," Technische Universität München Insitut für Informatik, TUM-I19, 2012.
[12] J. A. McCall, P. K. Richards, and G. F. Walters, "Factors in Software Quality," *Griffiths Air Force Base, N.Y. Rome Air Development Center Air Force Systems Command*, 1977.
[13] V. Basili, G. Caldiera, and H. D. Rombach, "Goal Question Metric Approach," *Encyclopedia of Software Engineering, John Wiley & Sons, Inc.*, pp. 528–532, 1994.
[14] S. Wagner, *Software Product Quality Control*, Springer-Verlag Berlin Heidelberg. 2013.
[15] G. G. Azgaldov *et al.*, "Qualimetry: the Science of Product Quality Assessment," *Standart y i kachest vo*, no. 1, 1968.
[16] G. Azgaldov, A. Kostin, and A. Padilla Omiste, *The ABC of Qualimetry, toolkit for measuring the immeasurable*, Ridero. 2015.
[17] Y. Argotti, C. Baron, and P. Esteban, "Quality quantification in Systems Engineering from the Qualimetry Eye," presented at the 13th Annual IEEE International Systems Conference (SysCon), Orlando, USA, 2019.
[18] "Online Oxford Dictionary - science definition," 2019. [Online]. Available: https://en.oxforddictionaries.com/definition/science.
[19] VDA QMC Working Group 13 / Automotive SIG, "Automotive SPICE Process Assessment / Reference Model., version 3.1 - revision 656." 01-Nov-2017.
[20] M. Nei and W.-H. Li, "Mathematical model for studying genetic variation in terms of restriction endonucleases," in *In Proceedings of the National Academy of Science of the USA*, 1979, vol. 76, pp. 5269–5273.
[21] Automotive SIG, VDA, "Automotive SPICE Process Assessment, version 2.5." 10-May-2010.

Confidential C