Practices to Define Software Measurements

Káthia Marçal de Oliveira

LAMIH CNRS UMR 8201
Université Polytechnique Hauts-de-France, Valenciennes, France
kathia.oliveira@uphf.fr

RESUME. *Métriques, mesures, indicateurs, estimations, etc. Bien que nommés de différentes manières et explorant différents angles, un fait est reconnu : les mesures de systèmes logiciels sont essentielles pour évaluer leur qualité, favoriser leur amélioration et contrôler leur production. Différentes méthodologies ont été définies (GQM, GQIM, PSM, etc.). Différentes études pratiques ont été publiées. Cependant, la définition des nouvelles mesures semble toujours une tâche non triviale. Dans cet article, nous présentons notre expérience sur la définition des mesures avec un ensemble de pratiques simples qui abordent des questions clés concernant différentes méthodologies. Ces pratiques ont été appliquées dans la définition de mesures pour différents types de systèmes (des « legacy systems » aux applications IoT modernes).*

ABSTRACT. *Metrics, measures, measurements, indicators, estimates and so on. Although named in different ways and exploring different angles, one fact is recognized: measuring software systems is essential for assessing their quality, promoting their improvement and controlling their production. Different methodologies have been defined (GQM, GQIM, PSM, etc.). Different studies in practice have been published. However, defining new measures still seems a not trivial task. In this article, we present our experience on how to define measures with a set of simple practices that address main issues of different methodologies. These practices have been applied in the definition of measures for different types of systems (from legacy systems to modern IoT applications).*

Mots-clés : mesure logicielle, métriques, indicateurs de qualité.

KEYWORDS: *software measurement, measures, metrics, quality indicators.*

1. Introduction

Web systems, Ubiquitous computing, Internet of Things (IoT), Smart Cities, and so on brought a significant diversity of software systems that support several of our daily activities. To ensure the improvement and adoption of these applications, it is essential to assess their quality. Measurements can help address some of the most critical issues in software development and provide support for evaluating, improve, and control the production of software systems (Briand *et al.* 2002). In this context, several well-defined classical measures can be applied (e.g. complexity of the code,

size, coupling, defect density, reuse, etc.). However, these new kinds of applications present specific particularities that need also to be measured to assure their quality. We quote for instance, the need to evaluate context-awareness, a common feature of personalized systems, IoT and ubiquitous software applications, that implies in the perceived quality of the software system (Lee and Yun, 2012; Carvalho *et al.* 2017, 2018). Several other studies to evaluate the quality of different particularities of these new kinds of systems can be found in literature, such as: privacy in ubiquitous systems (Jafari *et al*., 2011), transparency interaction in smart homes (Wu and Fu, 2012), user immersion degree in ubiquitous services (Lee and Yun, 2012), trust in adaptive systems (Evers *et al.*, 2010)), efficiency of data transfer in IoT applications (Paschou *et al.*, 2013). However, besides not being exhaustive, the defined measures are not always described in a way that allows their use. Carvalho *et al.* (2017) showed that more than 80% of measures found in a systematic review for ubiquitous applications were not formally defined. We are, therefore, faced with defining or redefining measures in order to evaluate these new kinds of systems.

Several works have been presented in the literature for the definition of measures, improvement of process, and institutionalization of measurement programs in the industry (see, for instance, an overview of these approaches in Tahir *et al.*, 2016). Nevertheless, defining new measures is always considered a complex activity. First of all, we need experts for the definition and also for the interpretation of the results, and they are not always available. Once we have experts, we are faced with the lack of adequate tools to collect and evaluate data. Moreover, the definition of the experimental protocols is also complex, there are not always professionals to proceed the evaluations and the evaluation itself is usually expensive (Oliveira *et al*., 2012). Due to these difficulties, it is common to give up measuring and miss good opportunities to perform simple but significant measurement studies.

The idea of this article was to take a step back and organize a set of practices we have applied to deal with those difficulties when defining measures (around of 90) for different software systems (ubiquitous systems (Carvalho *et al.*, 2018), legacy system (Ramos *et al.* 2004), interactive systems (Assila *et al.*, 2016; Gabillon *et al.*, 2013), web systems (Lima *et al.*, 2009)) and software process (Monteiro and Oliveira, 2010). These practices came from the application of different proposals from literature and became a roadmap we have followed to investigate and define measurements. We put together in this paper what worked for us showing different real examples so that it can be applied directly in new definitions of measurement.

We start this paper (section 2) by briefly presenting some concepts about measurements used as a basis for the set of practices described in section 3. Section 4 presents our conclusions.

2. Background

2.1. Basic Concepts

Some basic concepts are important to clarify why working on the definition of measures. First of all, measurement can be defined as follows: (i) The process by

which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules (Fenton and Pfleeger, 1997); and (ii) a set of operations having the object of determining the value of a measure ISO/IEC 15939 (2007).

In the first definition, an entity is an object (such as a person, a model, or a room) or an event (such as the testing phase of a software project). ISO/IEC 15939 (2007) summarizes that an entity is an object (a process, product, project, or resource) that is characterized by measuring its attributes. An attribute is a property of an entity (such as the color of a room, or the elapsed time of the test phase). The attributes are often defined using numbers and symbols (such as, a number of hours or the different labels for colors). Thus, we measure the attributes of entities by using specific measurement methods. The entity to be measured is the start point for a measurement definition. ISO/IEC 15939 (2007) organizes all these concepts associated with the measurement process defining some important concepts. It defends that a measurement process is driven by information needs (also named in literature as measurement goals), which are "insights necessary to manage objectives, goals risks and problems". To address the information needs, one can define:

- Base measures (named quality measure element in SQuaRE (ISO/IEC 25000, 2014)), that are independent measures defined in terms of an attribute (of an entity) and the method for quantifying it.
- Derived measures (named quality measure in SQuaRE (ISO/IEC 25000)), 2014) that are measures defined as a function of two or more values of base measures.
- Indicators - a measure defined using derived and base measures, and that is the basis for analysis and decision-making based on a model that combine one or more measures with associated decision criteria (thresholds or targets used to determine the need for action or further investigation, or to describe the level of confidence in a given result).

The methods used in the measures can be of two types: subjective, when the quantification of an attribute involves human judgment; or, objective, when the quantification is based on numerical rules such as counting, performed manually, or with automated tools. Finally, one of the following scales is associated with the measure (ISO/IEC 9126, 2001): nominal, ordinal, interval, or ratio. Measures using nominal or ordinal scales produce qualitative data, and measures using interval and ratio scales produce quantitative data (ISO/IEC 25000, 2014).

The literature is rich in measures for software products (models, code, software design, etc.), process and project. We can also find several systematic literature reviews (e.g. Nuñez-Varela et al. (2017), Carvalho et al. (2017), Hall et al., (2011), Bellini et al. (2008), Gómez et al (2008)) summarizing measures proposed in the last years.

2.2. Software Measurement Approaches

Several approaches have been proposed for the definition of measures. Some of the best known are: Goal-Question-Metric (GQM) (Basili *et al.*, 1994; Solingen and Berghout, 1999); the Goal-Question-Indicator-Metric (Park *et al.*, 1996), Practical

Software Measurement (PSM)[1] (McGarry *et al.*, 2002), the ISO/IEC 15939 (2007)) and the GQM/Metric Definition Approach (GQM/MDEA) (Briand *et al.*, 2002). All these approaches propose a set of steps in order to define software measurement for a product, process or project (see Figure 1). These approaches are differentiated by the number of steps presented and the detail given to carry out these steps. Some of them include some steps for the planning and integration of the measurement definition activities in the enterprise (see the first step of GQM, ISO/IEC 15939 and PSM in Figure 1). A common aspect in all these approaches is that they are goal-oriented, as introduced by GQM. Indeed, in a systematic literature review about software measurement, Tahir *et al.* (2016) concluded that the majorities of measurement planning models (83%) and measurements tools (90%) are extensions or improvement of GQM thanks to the goal-oriented measurement focus.

All these approaches are useful while defining measures and can be chosen without distinction, even if some (GQM, PSM and ISO/IEC 15939) are more used for measurement of software process and other (GQM, GQIM and GQM/MDEA) for software product (code, models, documentation etc.). By analyzing the steps particularly related to the software measurement definition (not considering the organization steps and the data collection), we note that three main issues are addressed in these approaches (Figure 3):

i. the **measurement goal definition** (steps highlighted in green - ♣ symbol in Figure 2), that focus on explicitly state the need for the measurement by formalizing the goal of the measurement in a clear and structured way;

ii. the **measure definition** itself (steps highlighted in blue - ♦ symbol in Figure 2), where, entities are identified and attributes are formalized via generic properties that characterize their measure (see previous section); and,

iii. the **measure evaluation** (steps highlighted in red -♠ symbol in Figure 2), that shows how to validate the measures defined and how to apply them for refinement and improvement of their definitions.

From our experience, regardless of which of the approaches from Figure 3 we follow, these issues are the most time-consuming and about which we have to overcome the main difficulties in the measurement definition (Dupuy-Chessa et al., 2014; Oliveira et al., 2012). The issue (i) should be deeply investigated by the team responsible for the measurements since it will guide the whole measurement program, being clear described, what is not always easy if we do not follow some standards template. Regarding measurement definition (issue (ii)), several difficulties are recognized: the need of an expert in the definition and interpretation of the measure, the insufficiency of adequate assessment tools, the definition and use of threshold and the large number of measures in literature. Finally, for the measurement evaluation (issue (iii)), validation procedures are not usual and experimental protocols are generally complex and difficult to define. However, we have to work with these difficulties and try to define as better as possible measurements that can support the quality evaluation of the software systems.

[1] New version of PSM v4.0b1 is available at http://www.psmsc.com/PSMGuide.asp

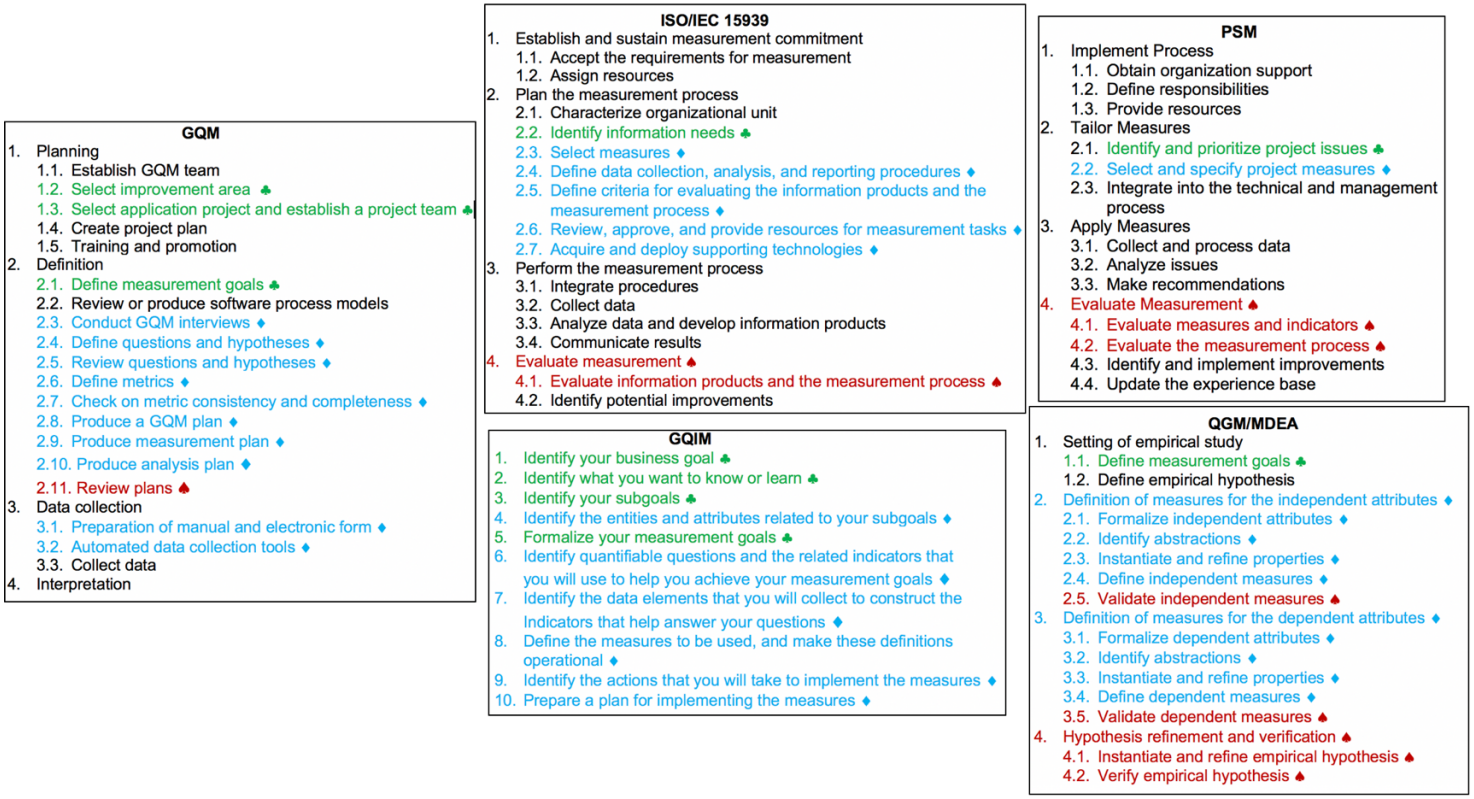

Figure 1. Steps of different software measurement approaches

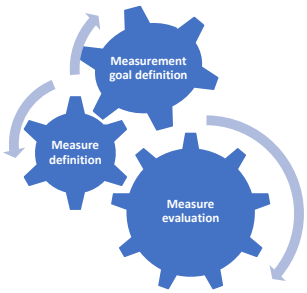

Figure 2. Main issues of software measurement approaches

3. Practices for Software Measurement Definition

Having been worked with measure definition for different proposes (e.g. ubiquitous systems (Carvalho *et al.*, 2018), legacy system (Ramos *et al.* 2004), interactive systems (Assila *et al.*, 2016; Gabillon *et al.*, 2013), web systems (Lima *et al.*, 2009) and software process improvement (Monteiro and Oliveira, 2010)), we have applied different practices that helped us to deal with the difficulties previously mentioned. These practices are presented in this section according to the main issues identified for software measurement presented in Figure 2.

3.1. Practices for Measurement Goal Definition

All approaches for software measurement definition starts by clearly establishing the measurement goal. In fact, measurement is a timing consuming and costly task. Therefore, the real need for the measurement should be identified since the beginning and should be revisited throughout the measurement definition to keep aligned with that goal. To that end, we should take into account the corporate objectives (Briand *et al.*, 2007), identify the business objectives that guide the organization's efforts and identify what one would like to know, in order to understand, evaluate, predict and improve activities related to the achievement of its objectives (Park *et al.*, 1996). A good practice is brainstorming with the team interested in the measurement trying to answer open questions, such as: What are the strategic goals of the organization/ project?, What are the major concerns (problems)?, What do we want to know/learn/improve?, How can we reach improvement goals?. After that, the measurement goal should be clearly written. We used as practice defining the goal following the structure defined in (Basili *et al.* 1994, Solingen and Berghout, 1999):

Analyze the *object* under measurement
For the purpose of understanding, evaluation, prediction, controlling, improvement
With respect to the *quality focus* of the object that the measurement focuses on
From the viewpoint of the *people* that measure the object
In the context of the environment in which measurement takes place.

Although it is a simple structure, each one of the lines of this definition is really important. From the beginning, it is essential to clearly define what is the object (the entity to be measured). As defined in section 2.1, the entity is the basis for the measurement process since we will measure the attributes of this entity.

For the *purpose* of the measurement, we should keep on mind what we want. It is very common to measure to obtain an accurate *characterization* or *evaluation* of the object for further analysis. *Prediction*, *controlling*, and *improvement* may require specific analysis models, comparisons among measurements, or continua data collection for some time slot.

Another important decision at this moment is the quality focus for the measurement definition. It is usually recommended to be decided according to the main interest of the organization by interviewing the sponsors. When in a research project, the choice regards the main interest of the project (e.g., usability, performance, etc.). In both cases, we usually use as practice look for a list of pertinent quality characteristic to support this decision. To that end, ISO/IEC 25010 (2011) can be used since it presents a set of quality characteristics (13) divided into sub-characteristics (42) related to outcomes of interaction with a system (the quality in use model) and related to the system/software product quality properties (the product quality model). Since this standard can be applied to any kind of software system, it is a rich source of quality focus when the entity to be evaluated is a product. However, a good practice is also look in the literature searching quality characteristics specific for the object been measured. This is particularly important when dealing with new kind of software systems like ubiquitous applications, IoT applications (e.g., smart cities), conversational agents, etc. Systematic mappings studies (Petersen *et al.*, 2015) proven to be quite useful in this case. It may reveal different quality focus not presented in the standards, and that covers particularities of that kind of system. Our study for ubiquitous systems (Carvalho *et al.*, 2018) showed particular quality focus regarding context-awareness, transparency, calmness, attention, and mobility.

Finally, it is important to define the *point of view* we are interested in precisely. That means the people that will measure and will benefit from the measurements. This will guide the definition of the measures to decide which attributes of the entity we are interested in measure to the purpose defined.

Figure 3 presents different examples of measurement goal definitions. It is worth mentioning some peculiarities of each of these definitions, as follows:

- in (a), the business goal that motivates the measurement was to evaluate an existing legacy system, in a short time in order to support outsourcing companies to establish the maintenance contracts to other companies. We aimed to have a view about the documentation and code of the legacy system to have an idea of the amount of work required.
- In (b), the interest was collecting data to define indicators for a service catalog of an SLA (Service Level Agreement) about website accessibility in order to support the definition of SLAs by the Brazilian Government while contracting companies to develop and maintain their web sites;

- In (c), the motivation was to evaluate quality characteristics that impact directly on the human-computer interaction in mobile applications;
- In (d), the context of transport applications imposes that the user interface should be as simple and complete as possible to support decision-making in real-time;
- In (e), the business goal behind this measure is to control the software development to monitor the time and cost to be aligned with the established contracts for a service in a software house.

Analyze the system documentation **for the purpose** of assessing **with respect to** completeness and consistency **from the viewpoints of** analysts and programmers **in the context of** the outsourced maintainer.	**Analyze** the system source code **for the purpose of** assessing **with respect to** the complexity to understand and modify it **from the viewpoints of** analysts and programmers **in the context of** the outsourced maintainer.

(a) Measuring legacy systems (Ramos *et al.*, 2004)

Analyze content on websites **for the purpose of** evaluation **with respect to** accessibility, **from the viewpoint of** the user with visual disabilities **in the context of** governmental services	**Analyze** ubiquitous systems **for the purpose of** evaluating **with respect to** context-awareness, mobility, transparency, attention and calmness **from the viewpoint of** user and developer **in the context of** mobile applications

(b) Measuring web applications (Lima *et al.*, 2009)

(c) Measuring ubiquitous applications (Carvalho *et al.*, 2018)

Analyze user interface **for the purpose of** improving **with respect to** information density **from the viewpoint of** developer **in the context of** transport application	**Analyze** project scope (time and cost) **for the purpose of** controlling **with respect to** performance **from the viewpoint of** managers **in the context of** software houses**.**

(d) Measuring interactive systems (adapted from (Assila *et al.*2016))

(e) Measuring software process (adapted from (Monteiro and Oliveira 2011))

Figure 3. Examples of measurement goal definition

3.2. Practices for Measure Definition

The measurement definition is the core of any measurement approach. First of all, it is important to be aware that there are a lot of measures already defined and being used in the literature. Consequently, a practice that we have applied is to look for measures in the literature considering the quality focus we have defined in the measurement goal before starting any activity for defining measures. Again, systematic mapping study is quite adequate. However, although we can find measures in literature, they are not always described and formalized in a way that we can reuse it. For instance, in the systematic mapping for ubiquitous application we have done (Carvalho *et al.*, 2018), we found 218 measures, but analyzing them against the formalization of measures purposed by SQuaRE (ISO/IEC 25000, 2014) and

(ISO/IEC 15939, 2007) we note that only a small part (39) was well defined presenting measurement functions and quality measurement elements for a definition. Nevertheless, even in the case we do not have the complete description of the measure, to have a list or ideas of measures for the quality focus at hand is very helpful.

By looking the literature, we can also find several measures to measure the same quality characteristics (for instance, we found several measures for code size and complexity of the code while evaluating legacy systems (measurement goal (a) in Figure 4). Therefore, we can either choosing one to work with or applying all of them to have different perspectives on the quality focus being measured.

To define the new measures, two main points should be considered. The first one is to have the entity to be measured as the main element. That means we should consider the entity, explore and define all attributes this entity has, and consider other entities (and also their attributes) that have some impact/relation on the entity in the study. All this information will be useful while defining the measure. The second point is to consider experts in the kind of object being measured. It is worth to choose different experts and perform individual and group interviews with them. Since group interviews are not easy to organize (problem of schedule and availability), it is good to have some support to integrate opinions and guide the measure definition. To that end, a good practice we have applied is to use abstraction sheets (Solingen and Berghout, 1999). This document summarizes the key issues about the measurement goal into four parts as follows:

- **Quality Focus** - in which experts should choose or define possible measures for the defined quality focus. At this moment, the list of measures we collected from the literature (even with only names of the measure and not a complete formalization) is really useful. They work like insights for the experts, promote discussion and stimulate the definition of measures;
- **Baseline Hypotheses** - in which experts use their experiences (from other projects) to set a possible interpretation value that means what they expect to find as acceptable values. We consider that at least one baseline hypotheses for each measure should be defined;
- **Variation Factors**, in which experts identify potential factors that can impact the suggested possible measures; and,
- **Impact of Variation Factor**, in which experts should answer how the various factors could impact the measures and what kind of dependencies exists between the measures and the factors.

Figure 5 presents an abstraction sheet regarding the measurement goal (c) presented in Figure 4.

With the abstract sheet filled in, we can (with the experts, if possible) define the measures starting from the measures listed in the quality focus quadrant, and then trying to define a measure for each variation factor that impact those measures. The definition of those measures should be based on the attributes of the entities related to them. From or experience, abstract sheets are effective in defining measures, even if we have only one expert or a study group of researchers interested on the entity being evaluated since it helps to brainstorm and to structure what can be measured.

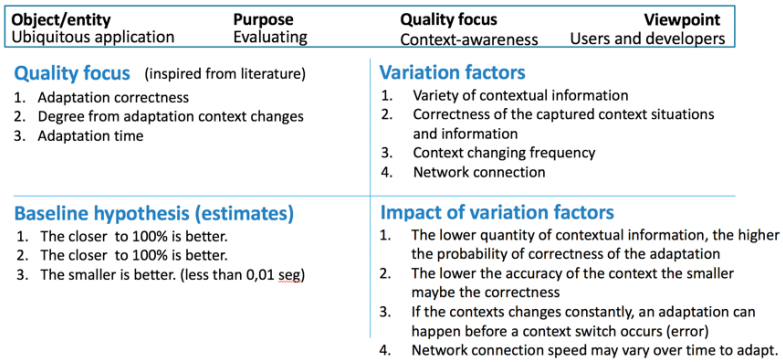

Figure 4. Example of abstraction sheet for ubiquitous applications evaluation (adapted from (Carvalho et al., 2018))

When defining a measure, we can conclude that to better evaluate the quality focus in study is necessary the combination of two measures in the same view. To that end, indicators as defined by ISO/IEC 15939 (2007) can be applied. For instance, for evaluating the information density of interactive systems, we concluded that it was important to have measures not only about the user interface itself but also about the users' opinion concerning their perception of the density of information while executing some tasks (Assila *et al.*, 2016). Indicators can also give some prevision for the object been evaluated. For instance, to control the software project (measurement goal (e) in Figure 4), we crossed measures of cost and time in control charts that support the analysis of stability over time.

To support the measure definition, a good practice is to follow a template for the measurement description. ISO/IEC 25000 series and ISO/IEC 15939 (2007) provide a list of elements we should consider while describing the measures. In general, at least the following information must be provided:

- Name – defined for convenience and that express the main meaning of the measurement;
- Description of the measure – the information described by the measure or gathered for the measures. This description must be cleared enough to make an easy understanding of the measured goal. To that end, we can use a sentence and/or a question to be answered by the application of the measure;
- Measurement function – ISO/IEC 25022 (2012) defines as an equation showing how the quality measure elements (base measures) are combined to produce the quality measure (derived measure). ISO/IEC15939 (2007) states that it can also be an algorithm or calculation performed to combine two or more base measures.
- Interpretation – a description to support the interpretation of the result for decision making. We propose two general practices. The first one is to normalize the value of the measure within 0.0 to 1.0 and that consider the interpretation as the closer to 1.0 is better (as suggested by in ISO/IEC 25022 (2012)) or close to

0.0 is better. The second practice is to apply the baselines defined in the abstraction sheets to define the initial values for thresholds. Those values can be revised after empirical evaluations (see next section).

- evaluation method – "procedure describing actions to be performed by the evaluator in order to obtain results for the specified measurement applied to the specified product components or on the product as a whole" (ISO/IEC 25000, 2007). In practice, we used three kinds of evaluation methods, as follows:
  - Questionnaires – classical way to collect data for the subjective measure from users' opinion. In this case, the Likert scale is commonly used. We have also used a continuous scale named VAS (Visual Analogue Scale) that allows the application of a wider range of statistical methods to the measurements.
  - Third-party observation – a third person that observe users interacting with the system during an evaluation session. The third person takes notes for further analysis. Forms with the data to be collected during the observation should be provided. The evaluation sessions can be face-to-face or remote.
  - Interaction log – use of automated data collection tools (usually specifically implemented for the system to be evaluated). It collects a trace of the execution of the system with specific data previously defined. The data to be collected is defined based on the defined measures.

Moreover, specific functions can be coded to collect measures from code (e.g.: complexity of the code). Several plug-ins and open-source tools are also available to this purpose[2].

Table 1 shows two examples of measure description. The first one was defined based on the abstraction sheet presented in Figure 4. Several examples of indicators description following ISO/IEC 15939 (2007) structure can be found in (Assila *et al.*, 2016; Monteiro and Oliveira, 2011).

3.3. Practices for Measure Evaluation

In order to guarantee the validity of a defined measure, we should assure its theoretical correction and apply it in several applications. The ideal scenario would be to have a historical database of the measures collected and use it to refine the thresholds defined to support the interpretation. This scenario requires a long-term research. In any case, a common sense in literature is that two kinds of validation are necessary for measurements (Srinivasan and Devi, 2014): theoretical and empirical.

Theoretical validation aims to confirm that the measurement does not violate any necessary properties of the elements of measurement (Srinivasan and Devi, 2014). To that end, a theory of definition of measures must be applied. We have used the theory of measures proposed in (Kitchenham *et al.*, 1995), (Fenton and Pfleeger, 1997) and

[2] For instance, for code in java (https://www.spinellis.gr/ sw/ckjm/; https://github.com/mauricioaniche/ck; https://www.sourcemeter.com); in C/C++, C# and Python (https://www.sourcemeter.com)).

(ISO/IEC 25000, 2007). In general, the main theoretical items from all these theories are the following: the entity to be measured, the attribute (also called "property to quantify" in (ISO/IEC 25000 series)), the scale type, and a measurement unit. Kitchenham *et al.* (1995) defend that it is also important to consider the adequation of the instrument and the adoption of a measurement protocol. (Srinivasan and Devi, 2014) also shows that mathematical properties may be assured. In any way, the main idea is to assure that we followed a theory of measurement while defining the measures. Table 2 shows a simple way of presenting the theoretical validation, by explicitly defining all elements of the measures (examples used in this paper). In the same way, Cheikhi *et al.* (2014) presents a theoretical validation of traditional well-known measure for object-orient systems.

Table 1. Example of specification of measurements

Name	Description	Measurement function	Interpretation	Evaluation Method
Adaptation Correctness (Carvalho *et al.*, 2018)	Does the adaptation occur correctly in the current context of the user?	$X = \frac{\left(\sum_{j=1}^{N} \frac{Aj}{Bj}\right) * 100}{N}$ N=Number of different adaptations Aj =Number of correctly performed adaptations j Bj =Number of performed adaptations j	The closer to 100%, the better.	Interaction log and Third party observation
Overall density (for graphical user interface) (Adapted from (Assila *et al.*, 2016))	It measures the percentage of display used to present all information.	X= Used space / Total space of an interface	0.0 < X < 1.0 The closer to 0 the better. **or** Acceptable: for values ϵ]0%,X] Unacceptable: for values ϵ]X;1] X in literature is usually 0.25-0.3	Coded function[3]

The empirical validation aims to confirm that a measure has the desired predictive power for predicting or evaluating the variable of interest (Antinyan *et al.*, 2016). According to (Srinivasan and Devi, 2014), three types of empirical validations are: surveys, case studies and experiments. Preferably, case studies, and experiments should be performed in a variety of application domains.

[3] This function in general calculate the total area of all the graphical components displayed in the interface against the overall area of the interface.

Table 2. Example of theoretical elements of a measurement

Measure	Base measure	Entity	Attribute	Scale	Unit
Adap-tation Correct-ness	Number of different adaptations	Running system	Adaptation Types (*e.g.*, adaptation occurred by battery, adaptation occurred by location)	Ratio	Adap-tation
	Number of correctly performed adaptations	Running system	Correctly performed adaptations	Ratio	Adap-tation
	Number of performed adaptations	Running system	Performed adaptations	Ratio	Adap-tation
Overall density	Used space of an interface	GUI compone nts (labels, textfields, images, etc.)	components' height components' width	Ratio	Pixel
	Total space of an interface	GUI	height width	Ration	Pixel

We have been using the action research method proposed by Antinyan *et al.* (2016) (Figure 5). In general, the measures are selected, calculated, evaluated, and redefined based on the evaluation until they are perceived to be good measures. To that end, the designer of the measurement (who defined the measures) and a reference group (a group of practitioners who work closely with the artifacts that are to be measured) work together during the cycle of validation.

The empirical evaluation should start with the definition of the research protocol (for a case study or experiment) to be applied. Classical elements such as subjects, time and environment for the study, and data collection methods should be clearly described. After the execution of the case study/experiment and the data collection, the designer presents the results for a reference group, and they brainstorm together to understand how effectively the selected measure can assess the variable of interest (Antinyan *et al.*, 2016). The aim is to evaluate if a given measure is effective or not and to check with the reference group whether the measurement results match the current state of the application (that means, the results correspond to what they know about the entity been evaluated). As consequence, either the reference group agrees with the results achieved or disagrees, indicating possible changes to be made. All agreements, disagreements, and reasons should be registered in a document that will be used for the measurement improvement. Then, a new cycle of evaluation is performed. From our experience, typical suggestions from the reference group are the

following: the modification of thresholds, modification of interpretation procedures and improvement of collect procedures (for instance, redefinition of questionnaires, and inclusion of new facilities in the automated tools).

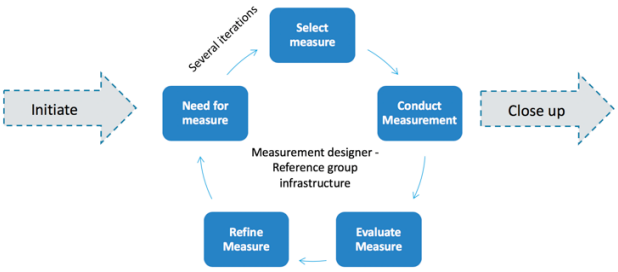

Figure 5. Action research cycle for validation of measures (Antinyan et al., 2016).

From our experience, we observed that after some action research cycles, we can obtain more stable thresholds for future interpretations. To deal with the complexity of definition of experimental protocols, we suggest the application of short cycles of evaluation with simple protocols, that can be evolved after each cycle.

4. Conclusion

This paper presented a set of practices to be applied while defining measures. These practices can be applied with any measurement approach while defining the measurement goal and measures, and proceeding the measure evaluation.

We can summarize the defined practices drawing the following principles: (i) follow a measurement approach; (ii) use a clear template for the measurement goal definition; (iii) consider the entity to be measured as the base element for measurement definition (we recall that we measure attributes of entities); (iv) look for measures already defined in literature even if not formalize is better than start from scratch; (v) include experts (developers of the kind of applications, managers, technical team, etc.) in the definition and the validation of the measures; (vi) formalize measurements by defining the measure name, description, measurement function, method, unit of measurement, etc.; (vii) automatize as much as possible (to collect the data for the base measures); (viii) apply and validate measurements.

We consider that the practices presented in this work can be directly applied to new projects and we hope that they can motivate the measurement definition and dissemination for new kinds of software system.

Acknowledgements

We strongly thank all co-authors of papers listed in this article.

References

Antinyan V., Staron M., Sandberg A. (2016). Validating Software Measures Using Action Research - A Method and Industrial Experiences, *17th International Conference on Enterprise Information Systems*, vol. 2, p. 15–27.

Assila A., Oliveira K., Ezzedine H. (2016). Integration of Subjective and Objective Usability Evaluation based on ISO/IEC 15939: a Case Study for Traffic Supervision Systems. *International Journal of Human-Computer Interaction*, 32 (12), p. 931-955.

Basili, V., Rombach, H. (1994). Goal Question Metric Paradigm, Encyclopedia of Software Engineering. Encyclopedia of Software Engineering – 2.

Bellini C.G., Pereira R.D.C.D.F., Becker J.L. (2008). Measurement in software engineering from the roadmap to the crossroads. *International Journal of Software Engineering and Knowledge*, 18(1), p. 37–64.

Briand L.C, Morasca S., Basili V. (2002). An Operational Process for Goal-Driven Definition of Measures. IEEE Transactions on Software Engineering, vol. 28, no. 12, p. 1106-1125.

Carvalho R., Andrade R., Oliveira K. (2018). AQUArIUM - A Suite of Software Measures for HCI Quality Evaluation of Ubiquitous Mobile Applications. *Journal of Systems and Software*, vol. 136, p. 101-136.

Carvalho R., Andrade R., Oliveira K., Santos I., Bezerra C. (2017). Quality characteristics and measures for human-computer interaction evaluation in ubiquitous systems. *Software Quality Journal*, 25(3), p. 743-795

Cheikhi, L., Al-Qutaish, R.E., Idri. A., Sellami, A. (2014) Chidamber and Kemerer Object-Oriented Measures: Analysis of their Design from the Metrology Perspective, International Journal of Software Engineering and Its Applications, vol.8, no.2, p. 359-374.

Dupuy-Chessa S., Oliveira K., Si-Said cherfi S. (2014). Qualité de Modèles : retour d'expérience. XXXIIème INFORSID, p. 363-378.

Evers V., Cramer H., Van Someren M., Wielinga, B. (2010). Interacting with adaptive systems. *Interactive collaborative information systems*, p. 299–325.

Fenton, N., Pfleeger, S. *Software Metrics A Rigorous & Practical Approach*, 2nd. Ed., PWS Publishing Company, 1997.

Gabillon Y., Lepreux S., Oliveira K. (2013) Towards ergonomic User Interface composition: a study about information density criterion. *15th International Conference on Human-Computer Interaction*, p. 211-220.

Gómez O., Oktaba H., Piattini M., García, F. (2008). A systematic review measurement in software engineering: state-of-the-art in measures. ICSOFT, LNCS 5007, p. 165–176.

Hall T., Beecham S., Bowes D., Gray D., Counsell S. (2011) A Systematic Literature Review on Fault Prediction Performance in Software Engineering. *IEEE Transactions on Software Engineering*, 38(6), p. 1276-1304.

ISO/IEC 15939. *System and Software Engineering – Measurement Process*, 2nd edition, 2007.

ISO/IEC 25000. (2014) *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE)*.

ISO/IEC 25010. (2011) -*SQuaRE – System and Software Quality Models*.

ISO/IEC 25022. *SQuaRE) — Measurement of quality in use*, July 2012.

ISO/IEC 9126. *Software engineering - Product quality - Part 1: Quality model*, 2001.

Jafari S., Mtenzi F., O'Driscoll C., Fitzpatrick R., O'Shea, B. (2011). Measuring privacy in ubiquitous computing applications. *International Journal of Digital Society*, 2(3), p. 547–550.

Kitchenham, B., Pfleeger, S.L., Fenton, N., 1995. Towards a framework for software measurement validation, *IEEE Transactions on Software Engineering*, pp. 929–944.

Lee, J., & Yun, M. H. (2012). Usability assessment for ubiquitous services: Quantification of the interactivity in inter-personal services. *IEEE international conference on management of innovation & technology*.

Lima S., Lima,F., Oliveira K. M. (2009) Evaluating the Accessibility of Websites to Define Indicators in Service Level Agreements, *11th International Conference on Enterprise Information Systems*, p. 858-869.

McGarry J, Card D, Jones C, Layman B, Clark E, Dean J, Hall F. (2002) *Practical Software Measurement: objective information for decision makers*. 1st ed. Addison-Wesley: Boston.

Monteiro L., Oliveira K. (2010). Defining a catalog of indicators to support process performance analysis. *Journal of Software Maintenance and Evolution: Research and Practice*, 23 (6), p. 395-422.

Nuñez-Varela, A.S., Pérez-Gonzalez, H.G., Martínez-Perez, F.E., Soubervielle-Montalvo C. (2017) Source code metrics: A systematic mapping study, *Information and Software Technology*, vol. 128, p. 164-197.

Oliveira K., Thion V., Dupuy-Chessa S., Gervais M.-P., Si-Said cherfi S., Kolski C. (2012). Limites de l'évaluation d'un système d'information : une analyse fondée sur l'expérience pratique. Actes XXXème Congrès INFORSID, p. 411-427.

Park R.E., Goethert W.B. e Florac W.A. (1996). *Goal Driven Software Measurement – a Guidebook*, CMU/SEI-96-BH-002, Software Engineering Institute.

Paschou M., Sakkopoulos E., Sourla E., Tsakalidis, A.(2013). Health Internet of Things: Metrics and methods for efficient data transfer. *Information and Software Technology*, 34, p. 189-199.

Petersen, K., Vakkalanka, S., & Kuzniarz, L. (2015). *Guidelines for conducting systematic mapping studies in software engineering: An update. Information and Software Technology*, vol. 64, p. 1-18.

Ramos C. S., Oliveira K. M., Anquetil, N. (2004). Legacy Software Evaluation Model for Outsourced Maintainer. *8th IEEE European Conference on Software Maintenance and Reengineering*, p. 48-57.

Solingen, R. van, Berghout, E. (1999). *The Goal/Question/Metric Method: A practical guide for quality improvement of software development*. McGraw-Hill.

Srinivasan, K.P., Devi, T. (2014). Software Metrics Validation Methodologies in Software Engineering. *Journal of Software Engineering and Applications*, vol. 5, p. 87–102.

Tahir T., Rasoola G., Gencelb C., (2016). A systematic literature review on software measurement programs. *Information and Software Technology*, 73, p. 101–121.

Wu, C. L., & Fu, L. C. (2012). Design and realization of a framework for human–system interaction in smart homes. *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, 42(1), p. 15–31.