6. An Ontology for Software Measurement

Manuel F. Bertoa and Antonio Vallecillo

University of Malaga, Dept. Lenguajes y Ciencias de la Computación, Málaga, Spain,
bertoa@lcc.uma.es, av@lcc.uma.es

Félix García

ALARCOS Research Group. Dept. of Information Technologies and Systems, Escuela Superior de Informática, University of Castilla-La Mancha, Spain,
Felix.Garcia@uclm.es

6.1 Introduction

Software measurement has evolved in such a way that it is no longer a marginal or atypical activity within the software development process and has become a key activity for software project managers. All successful software organizations use measurement as part of their day-to-day management and technical activities. Measurement provides organizations with the objective information they need to make informed decisions that positively impact their business and engineering performance [17]. As a matter of fact, CMMI (Capability Maturity Model Integration) includes software measurement as one of its requisites for reaching higher maturity levels and it helps organizations to institutionalize their measurement and analysis activities, rather than addressing measurement as a secondary function. Other initiatives such as ISO/IEC 15504 [11], SW-CMM (Capability Maturity Model for Software) and the ISO/IEC 90003:2004 standard [12] also consider measurement to be an important element in the management and quality of software. In all these initiatives measurement plays a fundamental role as a means for assessing and institutionalizing software process improvement programs.

However, as with any relatively young discipline, software measurement has some problems. When we approach software measurement and compare diverse proposals or international standards, it becomes apparent that the terminology used is not always the same or the same term may refer to different concepts. Terms such as “metrics”, “attribute”, or “measure” need to have a single definition accepted by all the researchers and practitioners who work in software measurement. The most serious point is when inconsistencies appear between different measurement proposals or standards.

Standards provide organizations with agreed and well-recognized practices and technologies, which assist them to interoperate and to work using engineering methods, reinforcing software engineering as an “engineering” discipline, instead of a “craft”. Furthermore, the Internet is changing how business is done nowadays, promoting cooperation and interoperation among individual organizations, which need to compete in a global market and economy, and share information and resources. Standardization is one of the driving forces to achieve this interoperability, with the provision of agreed domain conventions, terminologies and practices. However, there is no single standard which embraces the whole area of software measurement in its totality, but rather there are diverse standards orientated towards specific areas such as the measurement process or function points. Without an overall reference framework managing these standards, inconsistencies arise in the measurement terminology. This issue has been recognized by ISO/IEC, which has created a work group for the harmonization of systems engineering standards within its Joint Technical Committee 1 (JTC1: “Information Technology”, www.jtc1.org), and is trying to explicitly include in its directives the procedures which guarantee consistency and coherency among its standards. Furthermore, there has been an agreement in place since the year 2002 between the IEEE Computer Society and ISOJTC1-SC7 to harmonize their standards, which includes the terminology on measurement.

In spite of these efforts, the problem of terminology harmonization still needs to be resolved in our opinion. The objective of this chapter is to present a coherent software measurement terminology which has been agreed upon by consensus, i.e., without contradictions or disparities in the definitions, and a terminology which is widely accepted. The terminology presented in this chapter has been obtained as a result of an exhaustive analysis of the concepts and terms used in the field of software measurement. First of all, similarities, discrepancies, shortcomings and weaknesses in the terminology used in the main standards and proposals have been identified, including ISO International Vocabulary of Basic and General Terms in Metrology (VIM) [13] in the comparison [5]. The result has been a

software measurement ontology that provides a set of coherent concepts, with the relations between these concepts well defined, and which we hope helps to create a unified framework of software measurement terminology.

This chapter is organized as follows. After this introduction, Sect. 6.2 guies a brief analysis of the current situation. Section 6.3 presents the Software Measurement Ontology proposal; the concepts of the ontology and relationships among them are presented in detail grouped according to the sub-ontology to which they belong. A running example based on a real case study is used to illustrate the ontology. Finally, Sect. 6.4 draws some conclusions, proposes some suggestions for harmonization, and identifies future research work.

6.2 Previous Analysis

We selected sources from the existing international standards and research proposals that deal with software measurement concepts and terminology. From IEEE we took IEEE Std. 610.12: "Standard Glossary of Software Engineering Terminology" [7] and IEEE Std. 1061-1998: "IEEE Standard for a Software Quality Metrics Methodology" [8]. From ISO and IEC we selected the ISO/IEC 14598 series "Software engineering – Product evaluation" [9], the ISO VIM: "International Vocabulary of Basic and General Terms in Metrology" [13] and the International Standard ISO/IEC 15939: "Software engineering – Software measurement process" [10]. We also included other relevant research proposals related to software measurement, such as the ones by Lionel Briand et al. [3] and by Barbara Kitchenham et al. [16]. The general enterprise ontology proposed by Henry Kim [15] was also considered in the analysis, since it contains a sub-ontology for measurement concepts and terms. Other proposals that make use of measurement terminology (sometimes adapted to their particular domains) were also analyzed, although they were not included in the comparative study because they were either too specific, or clearly influenced by other major proposals already considered.

Once the sources were identified the next step was to collect from them all the definitions of terms related to software measurement. As a result of this, the first thing we realized was that the different standards and proposals could be basically organized around three main groups, depending on the particular measurement topics they focused on: software measures, measurement processes, and targets-and-goals. The first group of concepts, **software measures**, deals with the main elements involved in the

definition of software measures, including terms such as measure, scale, unit of measurement, etc. The second group, **processes**, is related with the actions of measuring software products and processes, including the definition of terms like measurement, measurement result, measurement method, etc. Finally, the third group, **target-and-goals**, gathers the concepts required to establish the scope and objectives of the software measurement process, e.g., quality model, measurable entity, attribute, information need, etc. It is worth noting that no single proposal from the set of analyzed sources covers all three groups. Moreover, the set of concepts covered by each source is not homogeneous, even for those sources focusing on the same group. There is a tendency in the sources, however, to converge around these three topic groups as they evolve over time.

However, once the ontology was created we discovered that it was not fully aligned with the VIM and with the new harmonization efforts taking place at ISO. Therefore, it was decided to adapt it in order to make it converge with these efforts, and the ontology presented here was subsequently created. The resulting software measurement ontology is therefore based mainly on the ISO VIM and ISO/IEC 15939 standards. It also includes some terms which are missing from these two documents (e.g., "quality model") that we think are essential in software measurement, and presents some discrepancies with ISO/IEC 15939, e.g., the treatment of indicators.

6.3 A Running Example

To illustrate the ontology, let us use an example based on a real case of software measurement which uses all of the concepts and terms of the ontology. It occurs in the context of a component-based development process of an industrial application, which needs to select a set of commercial off-the-shelf (COTS) software components to be integrated into the system.

More precisely, the software architect has decided not to develop a software component to provide the "print and preview" facilities of the application, but to obtain it from an external source, i.e., go to a software component repository (e.g., ComponentSource) and buy or license a commercial product. There seem to be some candidate components in the repository that provide similar functionality, from different vendors, and that could be used. Of course, the software architect wants to select the "best" component, i.e., the one that best suits the requirements and preferences. Therefore, he/she needs to evaluate the candidate components, i.e., measure them in order to rank them according to such requirements. To

simplify the example, let us suppose that the software architect is only interested in evaluating the **Usability** of the candidate components.

In this example we will also suppose that the organization counts on a set of analysis tools to facilitate the selection of COTS software. The major problem encountered when COTS software is assessed is the lack of source code. COTS components are developed and licensed by a third company, so their evaluation must be done without access to their code and internals. The organization has developed some tools to asses the component from two standpoints: its documentation (manuals, demos, marketing information, etc.), and its design. For the first, the organization uses an analysis tool for manuals in electronic format (as they are commonly provided). The software design is assessed by a tool that uses reflection techniques to interrogate and evaluate the COTS software. Thus, the tool can load a Java .jar file and then count the number of classes, methods, arguments, fields, etc., and also get their names and types.

This is the setting that we will use to illustrate the concepts of the ontology presented here.

6.4 The Proposal of Software Measurement Ontology

In this section we present the Software Measurement Ontology (SMO) proposal which we have developed to facilitate harmonization efforts in software measurement terminology. This ontology is based on an initial proposal [4], which had been created to address the lack of consensus on Spanish software measurement terms, based on the most representative measurement standards and proposals. Once the Spanish ontology was defined, it was translated into English. Finding the correct translation of each Spanish term became a rather difficult task and was done by comparing the different proposals again, and selecting the most appropriate terms in each case.

6.4.1 The SMO

With our comparison analysis we pursued the following goals: to locate and identify synonyms, homonyms, gaps and conflicts; to generalize the different approaches to measuring attributes; and to provide a smooth integration of the concepts from the three groups, so that measurement processes can be built using clearly defined measures, while quality models identify the targets and goals of the measurement processes.

A natural approach to achieving these goals was to use a common software measurement ontology, able to identify all concepts, provide precise definitions for all the terms, and clarify the relationships between them. Such an ontology also served as the basis for comparing the different standards and proposals, thus helping to achieve the required harmonization and convergence process for all of them. Another important requirement for the SMO was that its terms should try to conform to general terminology accepted in other fields, including measurement—which is a quite mature field with a very rich set of terms.

The SMO was developed with these goals in mind. The main features and characteristics of the SMO (shown in Fig. 6.1) are the following:

- It uses the term “measure" instead of “metric". This issue is one of the most controversial ones amongst software measurement experts nowadays. Although the term metric is widely used and accepted by many practitioners and researchers, this term has many detractors who argue the following reasons against its use. First, formally speaking a metric is a function that measures the distance between two entities—and therefore it is defined with the precise mathematical properties of a distance. Secondly, the definition of metric provided by both general and technical dictionaries does not reflect the meaning with which it is informally used in software measurement. Furthermore, metric is a term that is not present in the measurement terminology of any other engineering disciplines, at least with the meaning commonly used in software measurement. Therefore, the use of the term “software metric” seem to be imprecise, while the term “software measure” seems to be more appropriate to represent this concept. As a matter of fact, all new harmonization efforts at ISO/IEC and IEEE are trying to avoid the use of the term metric in order to fall into line with the other measurement disciplines, which normally use the vocabulary defined in metrology. In our proposal we finally decided to avoid the use of the term metric, using the term “measure” instead.
- It differentiates between “measure”, “measurement” and “measurement result”. These terms are used with different meanings in the different proposals (one of the reasons is that “measure” can be used as both a noun and a verb, and therefore it can be used to name both an action (to measure) and the result of the action). In our proposal the action is called “measurement”; the result is called “measurement result”; while the term “measure” defines the measurement approach that needs to be used to perform the measurement, and the scale in which the result is expressed.

- It distinguishes between base measures, derived measures and indicators, but considers them all as measures, generalizing their respective measurement approaches (measurement method, measurement function and analysis model).
- It integrates the software measures with the quality model that defines the information needs that drive the measurement process.

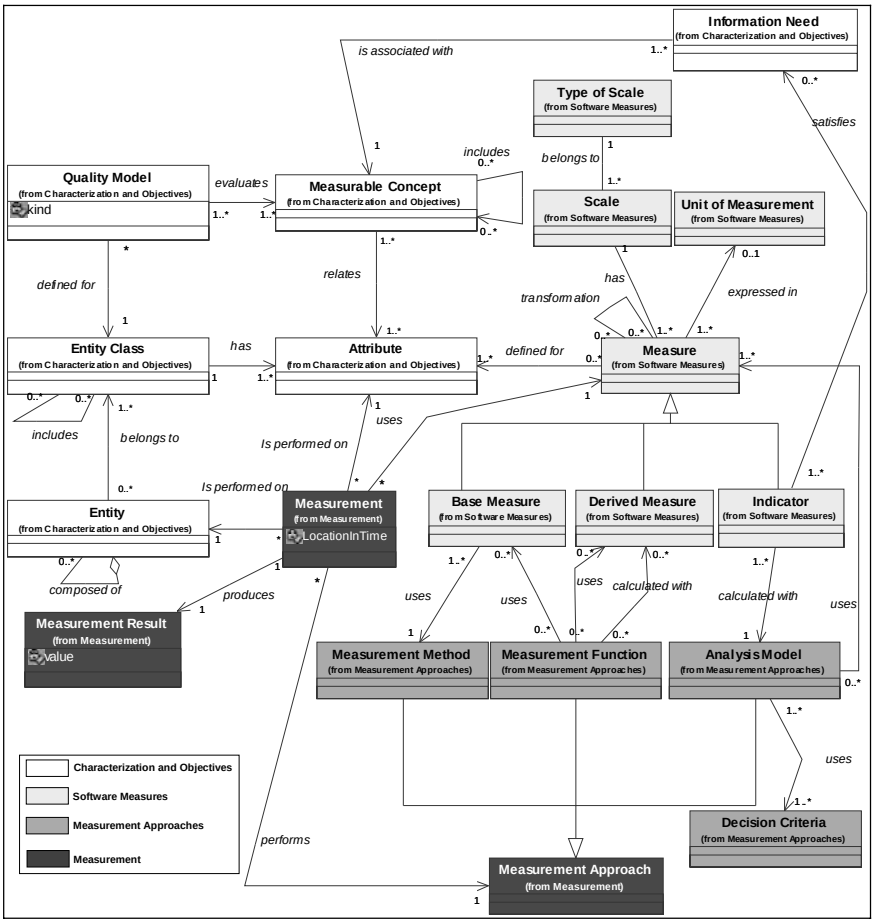

Fig. 6.1. The SMO

Figure 6.1 shows the terms of the SMO and their relationships, using the UML (Unified Modeling Language) notation. As can seen, the SMO has been organized around four main sub-ontologies: **Software Measurement Characterization and Objectives**, to establish the context and goals of the measurement; **Software Measures**, to clarify the terminology in-

volved in the measures definition; **Measurement Approaches**, to describe the different ways of obtaining the measurement results for the measures; and **Measurement**, which includes the concepts related to performing the measurement process. These four sub-ontologies are closely related to the three main groups of concepts identified above. Thus, the first sub-ontology corresponds to the target-and-goals group. The software measures sub-ontology corresponds to the measures group. The last two sub-ontologies together cover the measurement process group.

To represent the SMO we have chosen REFSENO (Representation Formalism for Software Engineering Ontologies) [18]. REFSENO provides constructs to describe concepts (each concept represents a class of experience items), their attributes and relationships. Three tables are used to represent these elements: one with the glossary of concepts, one table of attributes, and one table with the relationships. REFSENO also allows the description of similarity-based retrievals, and incorporates integrity rules such as cardinalities and value ranges for attributes, and assertions and preconditions on the elements' instances. Several main reasons moved us to use REFSENO for defining our ontology. First, REFSENO was specifically designed for software engineering, and allows several representations for software engineering knowledge whilst other approaches, e.g. [6, 19, 20], only allow representations which are less intuitive for people not familiar with first-order predicate (or similar) logics. In addition, REFSENO has a clear terminology, differentiating between conceptual and context-specific knowledge, and thus enabling the management of knowledge from different contexts. REFSENO also helps the building of consistent ontologies thanks to the use of consistency criteria. Unlike other approaches, REFSENO uses constructs known from case-based reasoning (CBR). Finally, REFSENO stores experience in the form of documents, and not as codified knowledge. This results in an important reduction of the learning effort required, something typically associated with knowledge-based systems [1].

The SMO was defined following the process suggested by REFSENO. More precisely, we used the following steps:

1. Define the concept glossary from the knowledge sources mentioned above.
2. Define the semantic relationships among the concepts by representing them in the UML notation, and create the relationship class tables.
3. Analyze the concepts which have some kind of relationship in order to identify the commonalities among two or more concepts. Then, we need to decide whether these commonalities are concepts (inserted

for modeling reasons) and, if so, to include them in the glossary of concepts.
4. Identify the terminal attributes of all the concepts and include them in the UML diagrams. Each time a new attribute type is identified, it must be included in the table of types.
5. Complete the attributes concept tables by including the non-terminal attributes.
6. Check the completeness of all the attribute tables.

The REFSENO representation of the SMO is presented in the following subsections. For simplicity, we describe only the terms and relationships for each sub-ontology.

6.4.1.1 "Software Measurement Characterization and Objectives" Subontology

The "Software Measurement Characterization and Objectives" sub-ontology includes the concepts required to establish the scope and objectives of the software measurement process. The main goal of a software measurement process is to satisfy certain information needs by identifying the entities (which belong to entity classes) and the attributes of these entities (which are the focus of the measurement process). Attributes and information needs are related through measurable concepts (which belong to a quality model). Figure 6.2 shows the concepts and relationships of the sub-ontology "Software Measurement Characterization and Objectives" expressed in a UML diagram. The terms of this sub-ontology are given in Table 6.1. The first two columns show the term being described and its super-concept in the ontology, respectively. The third column contains the definition of the term in the SMO. The final column shows the source (standard or proposal) where the term has been adopted from. Possible values in the fifth column can be:

- a reference to a source (e.g., 15939, VIM, 14598), meaning that the term and its definition have been adopted from that source without any changes;
- "Adapted from (source)", if the term has been borrowed from a source, but its definition has been slightly changed for completeness or consistency reasons;
- "Adapted from (source) (other term)", if the definition of the term has been borrowed from a source, but that term is known differently in the source; or

- *new*, if the term has been coined for the SMO, or has a new meaning in this proposal.

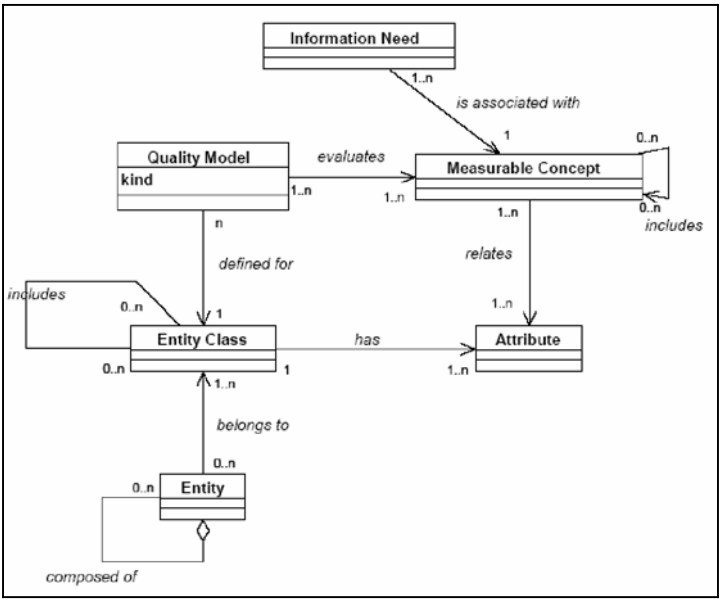

Fig. 6.2. "Software Measurement Characterization and Objectives" Sub-Ontology

Table 6.1. Concepts table of the sub-ontology characterization and objectives

Term	Super-concept	Definition	Source
Information Need	Concept	Insight necessary to manage objectives, goals, risks and problems	15939
Measurable Concept	Concept	Abstract relationship between attributes of entities and information needs	15939
Entity	Concept	Object that is to be characterized by measuring its attributes	15939
Entity Class	Concept	The collection of all entities that satisfy a given predicate	New
Attribute	Concept	A measurable physical or abstract property of an entity that is shared by all the entities of an entity class	Adapted from 14598
Quality Model	Concept	The set of measurable concepts and the relationships between them which provide the basis for specifying quality requirements and evaluating the quality of the entities of a given entity class	Adapted from 14598

Table 6.2 describes the relationships defined in the sub-ontology.

Table 6.2. Relationships table of the sub-ontology characterization and objectives

Name	Concepts	Description
Includes	Entity Class –Entity Class	An entity class may **include** several other entity classes An entity class may be included in several other entity classes
Defined for	Quality Model–Entity Class	A quality model is **defined for** a certain entity class. An entity class may have several quality models associated
Evaluates	Quality Model–Measurable Concept	A quality model **evaluates** one or more measurable concepts. A measurable concept is evaluated by one or more quality models
Belongs to	Entity–Entity Class	An entity belongs to one or more entity classes. An entity class may characterize several entities
Relates	Measurable Concept–Attribute	A measurable concept **relates** one or more attributes
Is associated with	Measurable Concept–Information Need	A measurable concept **is associated with** one or more information needs. An information need is related to one measurable concept
Includes	Measurable Concept–Measurable Concept	A measurable concept may **include** several measurable concepts. A measurable concept may be included in several other measurable concepts
Composed of	Entity–Entity	An entity may be **composed** of several other entities
Has	Entity Class–Attribute	An entity class **has** one or more attributes. An attribute can only belong to one entity class

6.4.1.2 Examples

In our example, the *Entity Class* is the "COTS components which provide services of print and preview", and an *Entity* is the component "C005 ElegantJ Printer V1.1 developed by Elegant MicroWeb". We use a *Quality Model* which is the one proposed in the norm ISO/IEC 9126 or we can adapt this generic model to the DSBC context and use our own quality model (for instance, we could use one specific quality model defined for software components, such as the COTS-QM quality model [2]).

Quality software is a complex and broad topic so we focus on only one quality characteristic, the **Usability**. We will try to assess COTS Usability measuring three sub-characteristics: Understandability, Learnability and Operability. Our goal will be to look for indicators for them.

Therefore, our *Information Need* is "to evaluate the Usability of a set of 'print and preview' COTS components that are candidates to be integrated

into a software application, in order to select the best among them". These sub-characteristics which we wish to measure are related, to a greater or lesser degree, to two *Measurable Concepts*: "Quality of Documentation" and "Complexity of Design".

Each measurable concept is related to one or more *Attributes*, so the Quality of Documentation is related to the attribute "Manual Size" and the Complexity of the Design is related to the "Customizability" among other attributes

6.4.1.3 "Software Measures" Sub-ontology

This sub-ontology aims at establishing and clarifying the key elements in the definition of a software measure. A measure relates a defined measurement approach and a measurement scale (which belongs to a type of scale). Most measures may or may not be expressed in a unit of measurement (e.g., nominal measures cannot be expressed in units of measurement), and can be defined for more than one attribute. Three kinds of measures are distinguished: base measures, derived measures and indicators. Figure 6.3 shows the concepts and relationships of the "Software Measures" sub-ontology.

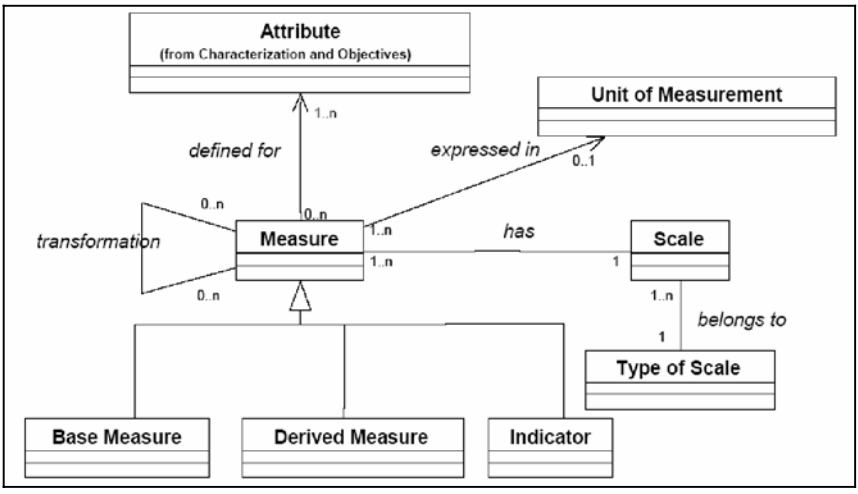

Fig. 6.3. "Software Measures" sub-ontology

Tables 6.3 and 6.4 show the terms and relationships of this sub-ontology, respectively.

Table 6.3. Concepts table of the sub-ontology software measures

Term	Super-concept	Definition	Source
Measure	Concept	The defined measurement approach and the measurement scale. (A measurement approach is a measurement method, a measurement function or an analysis model)	Adapted from 14598 "metric"
Scale	Concept	A set of values with defined properties	14598
Type of Scale	Concept	The nature of the relationship between values on the scale	
Unit of Measurement	Concept	Particular quantity, defined and adopted by convention, with which other quantities of the same kind are compared in order to express their magnitude relative to that quantity	VIM
Base Measure	Measure	A measure of an attribute that does not depend upon any other measure, and whose measurement approach is a measurement method	Adapted from 14598 "direct metric"
Derived Measure	Measure	A measure that is derived from other base or derived measures, using a measurement function as measurement approach	Adapted from 14598 "indirect metric"
Indicator	Measure	A measure that is derived from other measures using an analysis model as measurement approach	New

Table 6.4. Relationships table of the sub-ontology software measures

Name	Concepts	Description
Belongs to	Scale–Type of Scale	Every scale **belongs** to a type of scale. A type of scale may characterize several scales
Defined for	Measure–Attribute	A measure is **defined for** one or more attributes. An attribute may have several associated measures
Transformation	Measure–Measure	Two measures can be related by a **transformation** function; the kind of function will depend on the scale types of the scales
Expressed in	Measure–Unit of Measurement	A measure can be **expressed in** one unit of measurement (only for measures whose type is interval or ratio). A unit of measurement is used to express one or more measures of interval or ratio types
Has	Measure–Scale	Every measure **has** a scale. A scale may serve to define more than one measure

6.4.1.4 Examples

Let us define *measures* to measure each attribute. These measures are more complex at each step. The first step is to define a set of *Base Measures*, then *Derived Measures* and, if possible, *Indicators*. Each *Measure*, according to its type, has its corresponding *Measuring Approach* and *Scale.*

Thus, a *Base Measure* could be the "Number of Words of Manuals" (related to quality of documentation) or the "Number of Public Methods" provided by the component (related to complexity of design). For the first measure, its *scale* is "integers between 0 and +∞", its *type of scale* is "Ratio" and its units are "Kilo-words". The *scale* of the second measure is "integers between 1 and +∞", its *type of scale* is "Ratio" and its units are "Methods". Other base measures that we need to use to calculate the indicator presented below are "Number of Classes", "Number of Configurable Parameters" and "Number of Methods without Arguments".

We obtain *derived measures* using one or more base measures. Now, we want to calculate the *Derived Measures* "Ratio of Words per Functional Element". We have designated functional element (FE) to the aggregation of Classes, Methods and Configurable Parameters. Therefore, to calculate this derived measure we must know and calculate the base measures "Number of Words of Manuals", "Number of Classes", "Number of Methods" and "Number of Configurable Parameters". Its *Scale* is "real numbers from 0 to +∞" and its *units* are "Kilo-words/FE" with a ratio *type of scale*.

We wish to calculate the "percentage of methods without arguments" which is another example of a *Derived Measure.* We need to know the "Number of Methods without Arguments" and the "Number of Methods". In this case, the *scale* is "real numbers between 0 and 1" without *units* being a relative value (percentage) and with a "Ratio" *type of scale*.

Now, we could define an Indicator using some derived (or base) measures and defining an analysis model. For instance, we want to asseshs the Understandability inside the proposed quality model. We have an *indicator* named C_UND whose analysis model uses the ratio of words per FE (WpFE) and the percentage of methods without arguments (MwoA), by aggregating these two derived measures. After using its analysis model (i.e., its aggregation function) to calculate the indicator, the result can be given as a numerical value (e.g., 1.5) or a category value (e.g., Acceptable). In this example, the understandability indicator (C_UND) *type of scale* is "Ordinal" and it takes the values Acceptable, Marginal or Unacceptable, where Acceptable is better than Marginal and this is better than Unacceptable.

6.4.1.5 "Measurement Approaches" Sub-ontology

The "Measurement Approaches" sub-ontology introduces the concept of measurement approach to generalize the different "approaches" used by the three kinds of measures for obtaining their respective measurement results. A base measure applies a measurement method. A derived measure

uses a measurement function (which rests upon other base and/or derived measures). Finally, an indicator uses an analysis model (based on a decision criterion) to obtain a measurement result that satisfies an information need. Figure 6.4 shows the concepts and relationships of this sub-ontology, and Tables 6.5 and 6.6 show the terms and relationships of this sub-ontology.

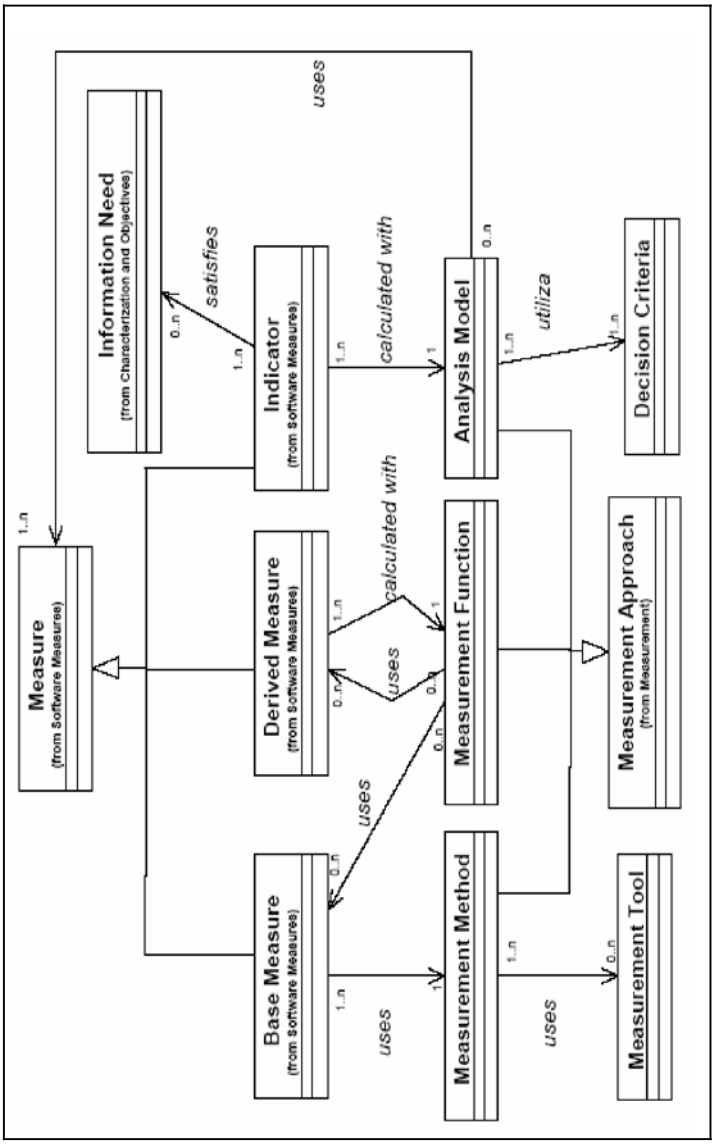

Fig. 6.4. "Measurement Approaches" sub-ontology

Table 6.5. Concepts table of the sub-ontology measurement approaches

Term	Super-concept	Definition	Source
Measurement Method	Measurement Approach	Logical sequence of operations, described generically, used in quantifying an attribute with respect to a specified scale. (A measurement method is the measurement approach that defines a base measure)	Adapted from ISO 15939
Measurement Function	Measurement Approach	An algorithm or calculation performed to combine two or more base or derived measures. (A measurement function is the measurement approach that defines a derived measure)	Adapted from ISO 15939
Analysis Model	Measurement Approach	Algorithm or calculation combining one or more measures with associated decision criteria. (An analysis model is the measurement approach that defines an indicator)	Adapted from ISO 15939
Decision Criteria	Concept	Thresholds, targets or patterns used to determine the need for action or further investigation, or to describe the level of confidence in a given result	15939

Table 6.6. Relationships table of the sub-ontology measurement approaches

Name	Concepts	Description
Uses	Base Measure–Measurement Method	Every base measure **uses** one measurement method Every measurement method defines one or more base measures
Calculated With	Indicator–Analysis Model	Every indicator is **calculated with** one analysis model. Every analysis model may define one or more indicators
Calculated With	Derived Measure–Measurement Function	Every derived measure is **calculated with** one measurement function. Every measurement function may define one or more derived measures
Satisfies	Indicator–Information Need	An indicator may **satisfy** several information needs. Every information need is satisfied by one or more indicators
Uses	Measurement Function–Base Measure	A measurement function may **use** several base measures. A base measure may be used in several measurement functions
Uses	Measurement Function–Derived Measure	A measurement function may **use** several derived measures. A derived measure may be used in several measurement functions
Uses	Analysis Model–Measure	An analysis model **uses** one or more measures. A measure may be used in several analysis models
Uses	Analysis Model–Decision Criteria	An analysis model **uses** one or more decision criteria. Every decision criterion is used in one or more analysis models

6.4.1.6 Examples

Let us look at some examples of measurement approaches for the measures proposed in previous sections. We have base measures, derived

measures and indicators. All of them have their own measurement approach but, depending on the type of measure, they have a measurement method, measurement function or analysis model, respectively.

"Number of Words of Manuals" is a base measure and, therefore, its measurement approach is a *Measurement Method*. In this case, it is composed of the following steps:

1. Run the software application for automatic evaluation of electronic manuals.
2. Load the executable component files (e.g., C005.jar).
3. Load the files of the component manuals
    a. If it is a HTML manual, indicating the path of the index file (index.htm).
    b. If it is a ".chm" file, selecting the file or files which compose the manual.
4. Select the function which counts words from the manual (e.g., select the "manual" drop-down flap and click on the "manual info" button).
5. Finally, read the obtained result from the screen or save it in a text file for later use.

The rest of the base measures have a similar measurement method describing the steps for calculating them using other options or different software tools if available.

The derived measure "Ratio of words per FE (WpFE)" has a measurement approach which is a *Measurement Function*. This *Measurement Function* uses some (previously calculated) base measure and its formula is the following:

$$\frac{NumberOfWords/1000}{NumberOfClasses + NumberOfMethods + NumberOfConfigParam}$$

The other derived measure "Percentage of Methods without Arguments (MwoA)" has the following *Measurement Function*:

$$\frac{NumberOfMethodsWithoutArguments}{NumberOfMethods} * 100$$

We use these two derived measures to evaluate the understandability indicator C_UND. We have a small *Analysis Model* that gives us a numerical value using a formula. Using this numerical value, we have *Deci-*

sion Criteria to obtain a final result. Therefore, its *Analysis Model* includes the following formula:

$$C\_UND = 0.2 \cdot WpFE - 1.4 \cdot MwoA + 1.6$$

Subsequently, we analyze the resulting values using the following *Decision Criteria*:

If $C\_UND > 1.2$ then the component is ACCEPTABLE;
If $C\_UND < 0.8$ then the component is UNACCEPTABLE;
Otherwise, the component is MARGINAL

6.4.1.7 Sub-ontology "Measurement"

This sub-ontology establishes the terminology related to the act of measuring software. A measurement (which is an action) is a set of operations having the object of determining the value of a measurement result, for a given attribute of an entity, using a measurement approach. Measurement results are obtained as the result of performing measurements (actions). Figure 6.5 shows the concepts and relationships of the sub-ontology.

Tables 6.7 and 6.8 show the terms and relationships of this sub-ontology.

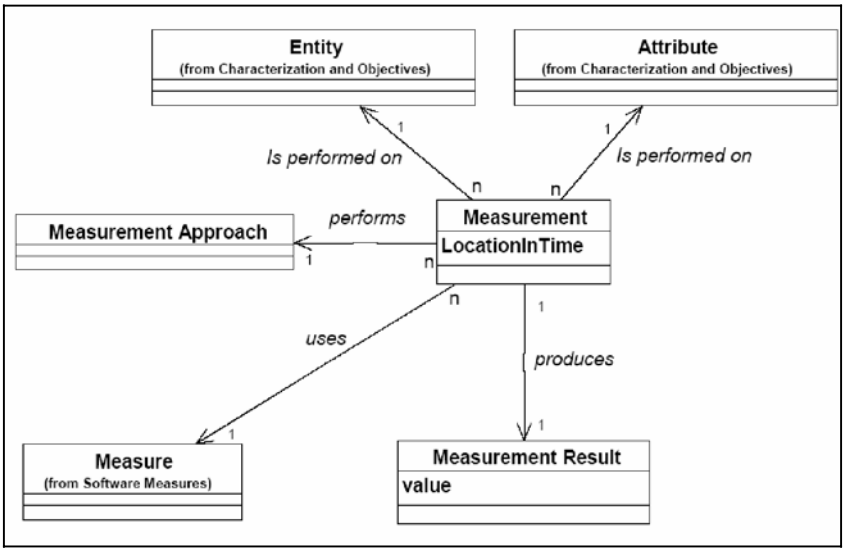

Fig. 6.5. "Measurement" sub-ontology

Table 6.7. Concepts table of the sub-ontology measurement

Term	Super-concept.	Definition	Source
Measurement Approach	Concept	Sequence of operations aimed at determining the value of a measurement result. (A measurement approach is a measurement method, a measurement function or an analysis model)	New
Measurement	Concept	A set of operations having the object of determining the value of a measurement result, for a given attribute of an entity, using a measurement approach	Adapted from VIM
Measurement Result	Concept	The number or category assigned to an attribute of an entity by making a measurement	Adapted from ISO 14598 "Measure"

Table 6.8. Relationships table of the sub-ontology measurement

Name	Concepts	Description
Performs	Measurement–Measurement Approach	A measurement is the action of **performing** a measurement approach (the kind of measurement approach will be dictated by the kind of measure used for performing the measurement). A measurement approach may be used for performing several measurements
Produces	Measurement–Measurement Result	Every measurement **produces** one measurement result. Every measurement result is the result of one measurement
Is Performed on	Measurement–Attribute	Every measurement **is performed** on one attribute of an entity (the attribute should be defined for the entity class of the entity)
Is Performed on	Measurement–Entity	Every measurement **is performed** on an entity, through one of its attributes (the attribute should be defined for the entity class of the entity)
Uses	Measurement–Measure	Every measurement **uses** one measure. One measure may be used in several measurements

6.4.1.8 Examples

A *Measurement* of the component understandability incorporates all the operations mentioned in previous points, using software tools and calculating base and derived measures and indicators. In the end, we obtain as a *Measurement Result* that the understandability of the component is "Acceptable" since its C_UND value is 1.5. Another *Measurement* for the component C012 could give us "Unacceptable" as a measurement result because C_UND has a value of 0.6.

Examples of *Measurement Results* are the following: "135,423 words", "243 methods", "34 classes", "22 Configurable Parameters", "0.41 kilowords/FE", "73%", or "Acceptable".

6.5 Conclusions

In the current (and not mature enough) software measurement field, the lack of a common terminology and inconsistencies between the different standards may seriously jeopardize the usefulness and potential benefits of these standardization efforts. Measurement terms have often been defined in unique and inconsistent ways from organization to organization. This has led to confusion and difficulty in widespread measurement implementation. In many cases, decision makers were unsure of what the measurement results actually represented [14].

In this chapter, a software measurement ontology has been presented, which aims to contribute to the harmonization of the different software measurement proposals and standards, by providing a coherent set of common concepts used in software measurement. The ontology is aligned with the most representative standards and proposals in the references and also with the metrology vocabulary used in other more mature measurement engineering disciplines. The common vocabulary provided by this common ontology has been used to resolve the problems of completeness and consistency identified in several international standards and research proposals and the ontology has been used as the basis for a comparative analysis of the terminology related to measurement [5].

The definition of the measurement terms to an adequate level of detail provides everyone with a common understanding of what is being measured, and how this relates to the measurement goals or information needs. Most of the problems in collecting data on a measurement process are mainly due to a poor definition of the software measures being applied. Therefore, it is important not only to gather the values pertaining to the measurement process, but also to appropriately represent the metadata associated to this data [16]. In this sense, the ontology provides the companies with the necessary conceptual basis for carrying out the measurement process and storing their results in an integrated and consistent way based on a rigorously defined set of measurement concepts and their relationships. Based on the ontology, companies can develop metamodels and languages to define their models for process and product measurement in a homogeneous and consistent way which can facilitate the integration and communication of their measurement process-related data and metadata. Consequently, a consistent software measurement terminology may also provide an important communication vehicle to companies when interoperating with others.

On the other hand, the proposed ontology can serve as a basis for discussion from where the software measurement community can start pav-

ing the way to future agreements. Without these agreements, all the standardization and research efforts may be wasted, and the potential benefits that they may bring to all users (software developers, ICT suppliers, tools vendors, etc.) may never materialize.

As a result, this proposal tries to address the needs of two main audiences: first, software measurement practitioners, who may be confused by the terminology differences and conflicts in the existing standards and proposals; and, second, software measurement researchers and standards developers (e.g., international standardization bodies and committees), who do not currently have at their disposal a cohesive core set of concepts and terms over which their existing standards could be integrated, or new ones built.

Our future plans include the extension of the ontology to account for most of the concepts in the forthcoming version of the VIM, in order to provide a complete ontology for software measurement, and fully aligned with the VIM beyond the core concepts contemplated in this proposal.

References

1. Althoff, K., Birk, A., Hartkopf, S. and Müller, W. (1999). Managing software engineering experience for comprehensive reuse. In Procedings of ICSE'99, Kaiserslautern, Germany,
2. Bertoa, M.F. and Vallecillo, A.(2002). Quality Attributes for COTS Components. I+D Computación, 1(2):128–144.
3. Briand, L., Morasca, S. and Basili, V. (2002). An operational process for goal-driven definition of measures. IEEE Transactions on Software Engineering, 28(12):1106–1125.
4. García, F., Ruiz, F., Bertoa, M., Calero, C., Genero, M., Olsina, L.A., Martín, M.A., Quer, C., Condori, N., Abrahao, S., Vallecillo, A. and Piattini M. (2004). An ontology for software measurement. Technical Report UCLM DIAB-04-02-2, Computer Science Department, University of Castilla-La Mancha, Spain, (in Spanish)
http://www.info-ab.uclm.es/descargas/thecnicalreports/DIAB-04-02-2/UCLMDIAB-04-02-2.pdf
5. García, F., Bertoa, M., Calero, C., Vallecillo, A., Ruiz, F., Piattini, M. and Genero, M. (2006). Towards a consistent terminology for software measurement. Information and Software Technology, 48(8),. Elsevier 631–644.
6. Gómez-Pérez, A. (1998). Knowledge Sharing and Reuse. CRC Press, Boca Raton, FL.
7. IEEE (1990). STD 610.12-1990. Standard Glossary of Software Engineering Terminology.
http://standars.ieee.org/reading/ieee/std_public/description/se/610.12-1990_desc.html

8. IEEE (1998). IEEE Std 1061-1998 IEEE Standard for a Software Quality Metrics Methodology – Description. http://standards.ieee.org/reading/ieee/std_public/description/se/1061-1998_desc.html
9. ISO/IEC (1999). ISO 14598: 1999-2001. Information Technology – Software Product Evaluation – Parts 1–6.
10. ISO/IEC (2002). ISO 15939: Software Engineering – Software Measurement Process.
11. ISO/IEC (2004a). ISO/IEC 15504-1:2003, Information technology – Process assessment – Part 1: Concepts and vocabulary.
12. ISO/IEC (2004b). ISO/IEC 90003, Software and Systems Engineering – Guidelines for the Application of ISO/IEC 9001:2000 to Computer Software.
13. ISO (1993). International Vocabulary of Basic and General Terms in Metrology (ISO VIM). International Organization for Standardization, Geneva, Switzerland, 2nd edition.
14. Jones, C. (2003). Making Measurement Work. CROSSTALK The Journal of Defense Software Engineering, pp. 15-19.
15. Kim, H. (1999). Representing and Reasoning about Quality using Enterprise Models. PhD thesis, Dept. Mechanical and Industrial Engineering, University of Toronto, Canada.
16. Kitchenham, B., Hughes, R. and Linkman, S. (2001). Modeling software measurement data. IEEE Transactions on Software Engineering, 27(9):788–804.
17. McGarry, J., Card, D., Jones, C., Layman, B., Clark, E., Dean, J. and Hall, F. (2002). Practical Software Measurement. Objective Information for Decision Makers. Addison-Wesley, Reading, MA.
18. Tautz, C. and Von Wangenheim, C. (1998). REFSENO: A representation formalism for software engineering ontologies. Technical Report No. 015.98/E, version 1.1, Fraunhofer IESE, October.
19. Staab, S., Schnurr, H. and Sure, Y. (2001). Knowledge processes and ontologies. IEEE Intelligent Systems, 16(1):26–34.
20. Uschold, M. and Gruninger, M. (1996). Ontologies: Principles, methods, and applications. The Knowledge Engineering Review, 11(2):93–196.