Towards a Theoretical Framework for Measuring Software Attributes

Sandro Morasca
Dipartimento di Elettronica e Informazione
Politecnico di Milano
Piazza Leonardo da Vinci 32,
I-20133 Milano, Italy
e-mail: morasca@elet.polimi.it
http://www.elet.polimi.it/~morasca

Lionel C. Briand
Fraunhofer Institute for Experimental Software
Engineering (FhG IESE)
Sauerwiesen 6
D-676761, Kaiserslautern, Germany
e-mail: briand@iese.fhg.de

Abstract

Several attributes (e.g., size, complexity, cohesion, coupling) are commonly used in Software Engineering to refer to software product properties. A large number of measures have been proposed in the literature to measure these attributes. However, since software attributes are often defined in fuzzy and ambiguous ways, it is sometimes unclear whether the proposed measures are adequate for the software attributes they purport to measure (i.e., their construct validity). In recent years, a few approaches have been proposed to lay theoretical foundations for defining measures for software attributes, but no widespread agreement has been reached on a rigorous, unambiguous definition of software attributes. In this paper, we first extend previous work carried out on axiomatic approaches for the definition of measures for software attributes. Second, we show how a hierarchical axiomatic framework can be constructed to support the definition of consistent measures for a given attribute at different levels of measurement. This paper shows how axiomatic approaches can be combined with the theory of measurement scales so that, depending on the level of sophistication of our empirical understanding of the attribute, we can select an appropriate level of measurement and a suitable axiomatic framework.

1. Introduction

Rigorous measurement of software attributes can provide substantial help in the evaluation and improvement of software products and processes. However, little agreement exists about the meaning of software attributes such as size, complexity, cohesion, and coupling, which are commonly used by researchers and practitioners to

- characterize internal software quality (e.g., "module A is more complex than module B"),
- predict external attributes of interest, (e.g., "longer modules are likely to have more faults than shorter ones").

One reason for this is that the software measurement field is a young one, if compared to other measurement fields, and the phenomena it addresses (e.g., the impact of loop complexity on the presence of defects in software modules) are less well understood than the phenomena of other older, more consolidated disciplines (e.g., Physics, Chemistry). Reaching a clearer understanding and a wider consensus about the meaning of software attributes is a necessary precondition to the progress of the software measurement field towards the solution of the practical problems it addresses.

As a response to this need for precise and unambiguous definition of measures for software attributes, a few proposal have been proposed in recent years, based on

- Measurement Theory [2, 10, 3],
- Axiomatic approaches [9, 5, 1].

Measurement Theory specifies the general framework in which measures should be defined. First, an Empirical Relation System should be specified, to define the relations among the entities as far as the studied attribute is concerned. Then, a Numerical Relation System is defined, to provide values for the measures of the attribute and relations among these values. A measure is the link between the Empirical Relation System and the Numerical Relation System, that maps entities into values and empirical relations among entities into formal relations among values. Axiomatic approaches formally define desirable properties of the measures for a given software attribute. These properties are properties of the Numerical Relation System of measures. However, they indirectly affect the Empirical Relation System. For instance, if a measure for an attribute does not satisfy axioms that have been stated for that attribute, then not only does one need has to review the Numerical Relation System, but he or she should go back to the Empirical Relation System and check if it really captures his or her own intuition. Therefore, axioms can be used as guidelines for the definition of a measure. (See also [1] Section 3.6 for a more detailed explanation.)

0-8186-8093-8/97 $10.00

A few proposals have been introduced by using either Measurement Theory or axiomatic approaches. For instance, Zuse, in [10], mainly focus on complexity of flowgraphs. He introduces the notion of atomic modification (a change in the entity being measured) and partial property (relation between the value of the metric before and after the modification). In [9], Weyuker introduces a set of software complexity axioms. Briand et al. [1] define a more general framework, with a number of sets of axioms to differentiate the measures for several software attributes (size, length, complexity, cohesion, coupling).

In this paper, we provide a contribution towards a framework for the unambiguous definition of measures for software attributes. First, we generalize previous work we carried out on axiomatic approaches. We define an axiomatic approach for measures of attributes of software systems in which the relationships among elements may not be only binary (as is the case in [1]). We show that new properties may be defined for some software attributes in this more general case, since this generalization offers one more degree of freedom in the modeling of software attributes.

Second, we outline a hierarchical axiomatic framework for the definition of measures of software attributes at different levels of measurement (e.g., nominal, ordinal, interval, ratio [2]). In this hierarchical framework, a software attribute is not associated with a single set of axioms, but different sets of axioms can be chosen, based on the measurement level of the measures one wants to define. This addresses in part the criticisms frequently made when assessing axiomatic approaches from a measurement theory perspective, e.g., discussions on [9] in [10] and [3] (see also [6, 4]). In order for the axioms in the hierarchy to be consistent, the axioms associated with different measurement levels need to satisfy consistency relationships. This shows how axiomatic approaches can be combined with the theory of measurement scales so that, depending on the level of sophistication of our empirical understanding of the attribute, we can select an appropriate level of measurement and a suitable set of axioms. Also, this framework allows for comparisons between different axiomatic approaches, to identify commonalities and differences. We want to highlight that, in this paper, we use the axioms we defined for in [1] to exemplify our approach, but other sets of axioms can be used as well.

The paper is organized as follows. In Section 2, we introduce the generalization of the axiomatic approach of [1]. In Section 3, we describe the hierarchical axiomatic framework. Conclusions and perspectives on future work follow in Section 4.

2. An axiomatic approach to the definition of measures for software attributes

2.1. Basic definitions

A software attribute can be associated with various kinds of artifacts. For instance, one can speak about size and complexity of software code, design, specification, for which different languages and formalisms may be used. However, the idea of an attribute is independent of the specific artifact and language chosen. Therefore, we define our axiomatic approach based on systems, i.e., one can speak about the size of a system, the complexity of a system, or the coupling among parts of a system. The definition of systems and modules we provide below generalizes that of [1]. The definition of elements and relationships can be tailored to needs for each specific artifact of interest.

System. A *system* S is represented as a pair S = <E,R>, where

- E represents the set of elements of S,
- R represents the set of relationships between S's elements; R is a set of tuples $<e_1, e_2, \ldots, e_i, \ldots, e_n>$ whose elements e_i belong to E. The arity (i.e., the number of elements) of different tuples in R may be different. For instance, some tuples may represent a relationship between two elements, others a relationship among three elements.

The need for this generalization arises from the fact that, in general, the relationships among the elements of the system are not only binary (as in [1]), but more than two elements can participate in a relationship, when modeling a system. As a first example of a system with n-ary relationships ($n \geq 2$), consider a software program in which variables are the system elements and a relationship among among variables exists if they are used in the same statement (e.g., an assignment, a function call). Since n variables may appear in the same statement, it is natural to use an n-ary relationship. Also, a different number of variables may appear in different statements. This circumstance can be represented by our definition of system, in which the arity of relationships is not fixed. As another example, consider a software procedure in which formal parameters and global variables are used. The call of that procedure implies a multi-way interaction among the actual parameters and global variables.

Relationships among n system elements are also commonly found in artifacts different from software code. Consider a Petri net [8] specification of a concurrent software system. Petri nets are a graph-based notation (with a formal mathematical foundation) in which nodes are partitioned into places (depicted as circles), used to represent the states of the processes that compose the system, and transitions (depicted as boxes), used to represent the changes of state of such processes. A fragment of a Petri net is in Figure 1. The arcs of a Petri net may only connect places to transitions and transitions to places (but not places to places or transitions to

transitions). In general, a transition may have more than one input place and more than one output place (see Figure 1(a)). The number of input places of a transition may be different from the number of output places of that transition. When the number of input places is greater (respectively, less) than the number of output places, the transition also represents destruction (respectively, creation) of processes. The fact that a process has reached a given state is represented by the presence of a token in the place representing that state of the process (in Figure 1(a) all the input places of transition t contain one token). The firing rule of a transition (i.e., the rule that shows how a system may evolve from one state to another) requires that at least one token[1] be present in each input place of the transition. The firing of a transition removes one token from each input place and creates one token in each output place (Figure 1(b) shows the effect of the firing of transition t). A transition therefore represents a synchronization among several processes, i.e., a relationship among the states of these processes before and after the transition. This synchronization relationship is intrinsically n-ary. Representing it by a set of binary relationships between pairs of processes may not be adequate, since the interaction part among the various processes would not be completely represented. The number of input and output places of different transitions may be different, which requires the ability to have relationships with different arities. Even if one describes the relationship between the states of the processes represented by the input places of a transition and the state of one process represented by an output place, an n-ary relationship is needed.

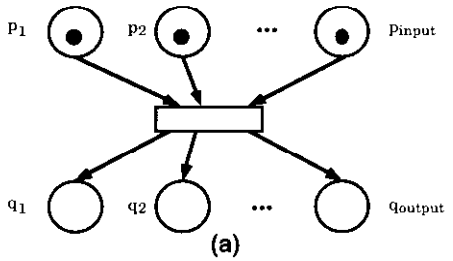

Figure 1. The firing of a Petri net transition.

In what follows, $E^{*\geq 2}$ will denote the set of all possible relationships whose arity is at least 2 built on the set E of elements. (We use this notation since this set is isomorphic to the set of all possible strings whose length is at least 2 that can be generated from the alphabet E.)

The definitions below are mostly unchanged from [1]. We report them here to make the paper self-contained.

Module. Given a system $S = \langle E,R \rangle$, the system $m = \langle E_m,R_m \rangle$ is a module of S if and only if $E_m \subseteq E$, $R_m \subseteq E_m{}^{*\geq 2}$, and $R_m \subseteq R$.

Outer Relationships. The elements of a module m are connected to the elements of the rest of the system S by the relationships in the set OuterR(m)

$OuterR(m) = \{\langle e_1, e_2, \ldots, e_i, \ldots, e_n \rangle \in R \mid \exists\, i, j\ (e_i \in E_m$ **and** $e_j \in E - E_m)\}$

Inclusion. Module $m_i = \langle E_{mi},R_{mi} \rangle$ is said to be included in module $m_j = \langle E_{mj},R_{mj} \rangle$ (notation: $m_i \subseteq m_j$) if $E_{mi} \subseteq E_{mj}$ and $R_{mi} \subseteq R_{mj}$.

Union. The union of modules $m_i = \langle E_{mi},R_{mi} \rangle$ and $m_j = \langle E_{mj},R_{mj} \rangle$ (notation: $m_i \cup m_j$) is the module $\langle E_{mi} \cup E_{mj},R_{mi} \cup R_{mj} \rangle$.

Intersection. The intersection of modules $m_i = \langle E_{mi},R_{mi} \rangle$ and $m_j = \langle E_{m\,j},\ R_{mj} \rangle$ (notation: $m_i \cap m_j$) is the module $\langle E_{mi} \cap E_{mj},R_{mi} \cap R_{mj} \rangle$.

Empty module. Module $\langle \varnothing,\varnothing \rangle$ (denoted by $\varnothing$) is the empty module.

Disjoint modules. Modules m_i and m_j are said to be disjoint if $m_i \cap m_j = \varnothing$.

Attributes such as cohesion and coupling require that a system be structured into modules, i.e., they are applicable to systems that are structured into modules or to modules of such systems.

Modular System. $MS = \langle E,R,M \rangle$ represents a modular system if $S = \langle E,R \rangle$ is a system and M is a collection of modules of S such that

$\forall\, e \in E\ (\exists\, m \in M\ (m = \langle E_m,R_m \rangle$ **and** $e \in E_m)\ )$ **and**
$\forall\, m_i,m_j \in M\ (m_i = \langle E_{mi},R_{mi} \rangle$ **and** $m_j = \langle E_{mj},R_{mj} \rangle$ **and** $E_{mi} \cap E_{mj} = \varnothing)$

i.e, the set of elements E of MS is partitioned into the sets of elements of the modules.

[1] For simplicity, we refer to Petri nets in which all arcs have weight 1.

Modules may be defined in different ways in different software artifacts and the studied attribute. For instance, C functions, Ada packages, C++ classes, or Ada tasks may be considered modules.

2.2. Axioms for measures of software attributes

Here, we show how the extension in the representation of systems outlined above affects the definition of axioms for the measures for software attributes. We take two of the software attributes for which axioms were provided in [1] as examples: size and coupling. The rationale behind this choice is that size refers to systems, while coupling refers to modular systems. Moreover, the axioms for size are unchanged with respect to those shown in [1], whereas one new axiom has been introduced among the axioms for coupling. The axioms for size will also be used to exemplify the building of hierarchies of properties (Section 3).

The meaning of the attributes below is to some extent subjective. One of the final goals of this line of research is to provide these attributes with a precise and unambigous semantics by means of an axiomatic approach, i.e., we associate a set of properties with the measures for each of these attributes. These axioms should be regarded as necessary properties, i.e., there might be measures that satisfy these properties but turn out to be practically useless. At any rate, it would not be possible—at least at this state of knowledge—to define axioms that are sufficient properties, i.e., satisfied only by practically sensible measures. It is not even clear whether sufficient properties will eventually be provided for attributes. For a comparison, take the concept of distance between two points of a set, which is formalized by means of three well-known axioms. One can define measures that satisfy these three axioms but are practically useless. Our axioms (or, in general, the axioms of any axiomatic approach) can be used to rule out measures that cannot be considered legitimate measures for specific software attributes they purport to measure.

For each attribute below, we provide a short explanation of the rationale behind the axioms we introduce. A more comprehensive discussion can be found in [1]. For short, it will be implicitly understood that all free variables in the logical formulae defining the axioms are universally quantified. In addition, it will be implicitly understood that all of the axioms belonging to a set of axioms for a software attribute are assumed to hold at the same time, i.e., they are parts of a single logical conjunction.

The sets of axioms below are meaningful only for ratio scale measures, i.e., they do not apply to measures whose scale is not ratio. Section 3 shows how one can build axiom sets that are both consistent with the axiom sets below and meaningful for measures defined on other measurement levels.

Axioms for Size

Size is, with complexity, the most often used software attribute. Axioms Size.I - Size.III below characterize measures that purport to measure the size of a system. In our approach, size depends on the elements of a system, not its relationships.

Size cannot be negative

$$Size(S) \geq 0 \qquad \text{(Size.I)}$$

The size of a system that does not contain any element is null

$$E = \varnothing \Rightarrow Size(S) = 0 \qquad \text{(Size.II)}$$

When modules do not have elements in common, size is additive

$$(m_1 \subseteq S \textbf{ and } m_2 \subseteq S \textbf{ and } E = E_{m1} \cup E_{m2} \textbf{ and } E_{m1} \cap E_{m2} = \varnothing) \Rightarrow Size(S) = Size(m_1) + Size(m_2) \qquad \text{(Size.III)}$$

The above axioms Size.I-Size.III are the same as the ones contained in [1], i.e., they are not affected by the generalization of Section 2.1.

Axioms for Coupling

Coupling is used with reference to modules or modular systems, and captures the amount of relationships among the elements belonging to different modules of a system. Axioms Coupling.I-Coupling.VI below characterize measures that purport to measure the coupling of a module $m = \langle E_m, R_m \rangle$ of a modular system.

Coupling cannot be negative

$$Coupling(m) \geq 0 \qquad \text{(Coupling.I)}$$

The coupling of module m is null if m has no relationships with the other modules

$$OuterR(m) = \varnothing \Rightarrow Coupling(m) = 0 \qquad \text{(Coupling.II)}$$

Adding inter-module relationships does not decrease coupling

$$OuterR(m') \subseteq OuterR(m'') \Rightarrow Coupling(m') \leq Coupling(m'') \qquad \text{(Coupling.III)}$$

Merging two modules yields a module whose coupling is not greater than the sum of the couplings of the two original modules, since the two modules may have inter-module relationships from one to the other which are intra-module relationships in the new module.

Given the modular systems $MS' = \langle E,R,M' \rangle$ and $MS'' = \langle E,R,M'' \rangle$, let $M''=M'-\{m'_1,m'_2\}\cup\{m''\}$ with $m'_1 \in M'$, $m'_2 \in M'$, and $m'' \notin M'$. If $E_{m''} = E_{m'1} \cup E_{m'2}$ and $R_{m''} = \{\langle e_1,e_2,\ldots,e_i,\ldots,e_n \rangle \in (E_{m''})^* | \langle e_1,e_2,\ldots,e_i,\ldots,e_n \rangle \in R\}$

$Coupling(m'_1) + Coupling(m'_2) \geq Coupling(m'')$
(Coupling.IV)

The coupling of disjoint modules is additive. Given the modular systems MS' = <E,R,M'> and MS'' = <E,R,M''>, let $M'' = M' - \{m'_1, m'_2\} \cup \{m''\}$ with $m'_1 \in M'$, $m'_2 \in M'$, and $m'' \notin M'$, and $m'' = m'_1 \cup m'_2$. If no relationships among the elements belonging to m'_1 and m'_2

$Coupling(m'_1) + Coupling(m'_2) = Coupling(m'')$
(Coupling.V)

The coupling of a module $S_1 = <E,R_1>$ is not greater than the coupling of a module $S_2 = <E,R_2>$ where one of the relationships in $OuterR_1$ $<e_1, e_2, \ldots, e_i, \ldots, e_n>$ is replaced in $OuterR_2$ by a relationship $<e_1, e_2, \ldots, e_i, \ldots, e_j, \ldots, e_{n+1}>$ that "covers" it (i.e., tuple $<e_1, e_2, \ldots, e_i, \ldots, e_j, \ldots, e_{n+1}>$ contains the same elements as relationship $<e_1, e_2, \ldots, e_i, \ldots, e_n>$ in the same order and one more element in some position j)

$(m_1 = <E_m,R_{m1}>$ **and** $m_2 = <E_m,R_{m2}>$ **and**
$\{<e_1, e_2, \ldots, e_i, \ldots, e_n>\} \in OuterR_1$ **and**
$\{<e_1, e_2, \ldots, e_i, \ldots, e_n>\} \notin OuterR_2$ **and**
$\{<e_1, e_2, \ldots, e_i, \ldots, e_j, \ldots, e_{n+1}>\} \notin OuterR_1$ **and**
$\{<e_1, e_2, \ldots, e_i, \ldots, e_j, \ldots, e_{n+1}>\} \in OuterR_2)$
$\Rightarrow Coupling(m_1) \leq Coupling(m_2)$ (Coupling.VI)

The above axioms for size and coupling provide an example of how the presence of n-ary relationships affects the axioms for measures of software attributes. In the case of size axioms, whose definition is based on the elements of a system, no further axioms have been introduced. On the other hand, one more axiom (Coupling.VI) has been introduced for coupling, whose definition is based on the relationships across modules. Other axioms may be added to this set of axioms. For instance, one might argue that the coupling of a module m is the same as the coupling of a module m' obtained by adding a relationship "covered" (according to the notion of "covering" provided above) by at least one relationship in OuterR(m). As another example, one might argue that the coupling of a module m does not change if all of its outer relationships are reversed: starting from a system S that includes module m, one can build a system S' with a module m' which coincides with module m and whose outer relationships are such that for each relationship $<e_1, e_2, \ldots, e_i, \ldots, e_n>$, $<e_1, e_2, \ldots, e_i, \ldots, e_n> \in OuterR(m)$ if and only if $<e_n, \ldots, e_i, \ldots, e_2, e_1> \in OuterR(m')$. We do not think these are suitable axioms for coupling, but somebody else's intuitition may disagree with ours. It is also true that one may not accept the new axiom we provided for coupling. At any rate, the generalization described in Section 2.1 makes it possible to investigate several new possibilities for axioms, to better define the software attribute. However, when building axiom sets for software attributes, it seems advisable to start with axioms for systems with only binary relationships. This allows one to obtain a firm preliminary understanding of the attribute, in an important particular case. One may relax the constraint of having only binary relationships later.

3. Hierarchies of axioms

The axioms shown above for size and coupling can be applied only to measures at the ratio level of measurement[2]. In this section, we show how hierarchies of sets of axioms for a software attribute can be built to support the definition and selection of measures at different levels of measurement. To this end, we take the axioms of size as an on-going example.

The axioms for size of Section 2.2 have consequences, i.e., other axioms implied by Size.I-Size.III. For instance, the following axiom is a size axiom too, implied by Size.I-Size.III.

Adding elements to a system cannot decrease its size (*size monotonicity axioms*)
$(S' = <E',R'>$ **and** $S'' = <E'',R''>$ **and** $E' \subseteq E'')$
$\Rightarrow Size(S') \leq Size(S'')$ (Size.IV)

This axiom, if taken in isolation (i.e., independent of the three axioms Size.I-Size.III), is relevant for measures whose level of measurement may not be ratio. For instance, it is easy to show that axiom Size.IV is satisfied by measures whose level of measurement is at least ordinal (i.e., by showing that the property holds under the admissible transformations of the ordinal level of measurement). Therefore, one may accept the above axiom Size.IV as an axiom that can be used to define measures for size whose level of measurement is ordinal or higher. Axiom Size.IV is consistent with the three axioms Size.I-Size.III, i.e., all the measures that satisfy the set of axioms Size.I-Size.III also satisfy Size.IV.

Another axiom implied by axioms Size.I-Size.III is the following.

Systems with the same set of elements have the same size, regardless of their set of relationships
$(S' = <E,R'>$ **and** $S'' = <E,R''>)$
$\Rightarrow Size(S') = Size(S'')$ (Size.V)

This axiom is relevant for measures that may not even be at an ordinal level of measurement, i.e., it is relevant for measures whose level of measurement is nominal. Axiom Size.V is also implied by axiom Size.IV, in that

$((S' = <E',R'>$ **and** $S'' = <E'',R''>$ **and** $E' \subseteq E'')$

[2]This does not make these properties invalid as properties for their associated attributes, as implied in [2]. Simply, they are not applicable to other measures than ratio scale measures.

$\Rightarrow$ Size(S') $\leq$ Size(S") **and**
(S' = <E',R'> **and** S" = <E",R"> **and** E" $\subseteq$ E')
$\Rightarrow$ Size(S') $\geq$ Size(S"))
$\Rightarrow$ Size(S') = Size(S")

Therefore, one obtains three sets of axioms, as follows

Ax1: The set of axioms for size of Section 2.2
Size(S) $\geq$ 0 (Size.I)
E = $\emptyset$ $\Rightarrow$ Size(S) = 0 (Size.II)
($m_1 \subseteq S$ **and** $m_2 \subseteq S$ **and**
$E = E_{m1} \cup E_{m2}$ **and** $E_{m1} \cap E_{m2} = \emptyset$)
$\Rightarrow$ Size(S) = Size(m_1) + Size(m_2) (Size.III)

Ax2: The following set of axioms (with only one axiom) is implied by Ax1 and relevant for measures at the ordinal level of measurement
(S' = <E',R'> **and** S" = <E",R"> **and** E' $\subseteq$ E")
$\Rightarrow$ Size(S') $\leq$ Size(S") (Size.IV)

Ax3: The following set of axioms (with only one axiom) is implied by Ax2 and relevant for measures at the nominal level of measurement
(S' = <E,R'> **and** S" = <E,R">)
$\Rightarrow$ Size(S') = Size(S") (Size.V)

Other sets of axioms for size may be conceived, consistent with the axiom set Ax1, two of which are listed below. For instance, axiom set Ax4 of is implied by the three axioms of Ax1 and is relevant for measures at the ratio level of measurement. Therefore, this set represents a weaker form of Ax1.

Ax4: The following set of axioms is implied by Ax1 and is relevant for measures at the ratio level of measurement
Size(S) $\geq$ Size($\emptyset$) (Size.VI)
($m_1 \subseteq S$ **and** $m_2 \subseteq S$ **and** $E = E_{m1} \cup E_{m2}$
$\Rightarrow$ Size(S) $\leq$ Size(m_1) + Size(m_2) (Size.VII)

Ax5: The following set of axioms (with only one axiom) is implied by Ax4 and is relevant for measures at the ordinal level of measurement
Size(S) $\geq$ Size($\emptyset$) (Size.VI)

These five axiom sets can be arranged in a hierarchy, as depicted in Figure 2.

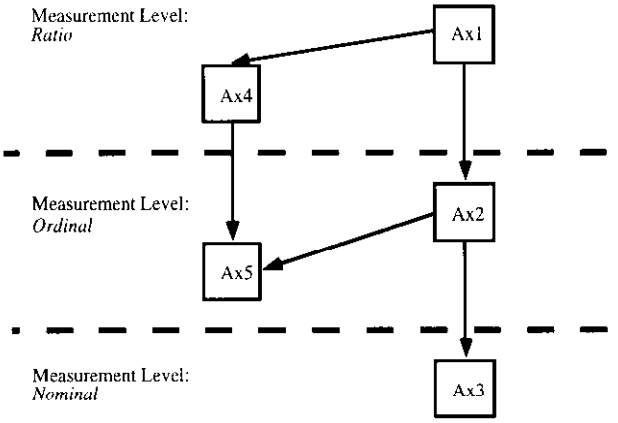

Figure 2. Hierarchy of axiom sets for size.

In general, given a set of axioms Ax for a software attribute, a hierarchy can be built, in which

- the nodes represent sets of axioms,
- node Ax has no incoming arcs.

For simplicity, in these hierarchies we do not show all the implications among the sets of axioms, but only the direct ones. For instance, if $Ax_i \Rightarrow Ax_j \Rightarrow Ax_k$, then $Ax_i \Rightarrow Ax_k$, since logical implication is transitive. For instance, in Figure 2, the axiom set Ax_1 implies all other axiom sets. Also (see Figure 2), a hierarchy is built in the general case, not necessarily a tree.

This hierarchical mechanism can be used in a top-down fashion. One starts from a set of properties that are defined on a specific measurement scale and derives—by means of logical implications—other sets of properties. This serves two purposes:

- on the one hand, one is able to realize the implications of one's own understanding and modeling of an attribute across all levels of measurement,
- on the other hand, one can explore several alternatives for the definition of sets of axioms for measures at each measurement level.

As the hierarchy in Figure 2 shows, different axiom sets at the same measurement level may exist in the same hierarchy. At any rate, when defining the axiom sets for a software attribute, one may build a whole hierarchy but then has to select a path, i.e., linearly ordered part of the hierarchy, in which all of the axiom sets are related to each other by implication relationships. This the nodes of this path will be used to characterize and assess measures at each level of measurement for which there is an axiom set. In addition, one should choose axioms at different measurement levels. For instance, one may choose the following linear orders in the axiom sets in the hierarchy of Figure 2

A: Ax_1, Ax_2, Ax_3

B: Ax_1, Ax_5

C: Ax_4, Ax_5

D: Ax_1, Ax_3

E: Ax_2, Ax_3

As a counterexample, Ax_4, Ax_2, Ax_3, cannot be chosen, since Ax_4 does not imply Ax_2. As these linear orders show, it is not mandatory to include any specific axiom set (for instance, it is not mandatory to include the axiom set Ax1, as linear orders B and C show), nor to have axiom sets at all measurement levels. In principle, the choice of the axiom sets and the linear order is a subjective matter. However, failing to include an axiom set for a measurement level may entail failure to be able to discriminate measures at that measurement level.

In addition, despite this freedom in choosing the sets of axioms for the measures of a software attribute, it is largely advisable that, at each measurement level, one choose the strictest set of axioms that formalize his or her own knowledge about the software attribute. For instance, in the hierarchy of Figure 2, one should choose the axiom set Ax_1 at the ratio measurement level, Ax_2 at the ordinal measurement level, and Ax_3 at the nominal measurement level. For instance, choosing the axiom set Ax_4 at the ratio measurement level would cause

- a loss of information about one's understanding of the attribute: one's own understanding of the size attribute as described by the axiom set Ax_1 is more precise than that embodied in the axiom set Ax_4;
- a loss of power to reject measures for an attribute at a given measurement level: based on the axiom set Ax_4, one may accept size measures at the ratio level of measurement that would be rejected at the ratio level of measurement if one's own understanding of the size attribute was modeled by the axiom set Ax_1.

Whenever possible, one should also include axiom sets at the higher levels of measurement in the sets of axioms chosen. In general, axioms that are satisfied by measures at lower levels of measurement (say, nominal or ordinal) are weaker than axioms that are satisfied by measures at higher levels of measurement (say, interval or ratio). As a consequence, axioms at lower levels of measurement do not provide as much contribution to the modeling of an attribute, and they are less interesting and useful. This is the reason why researchers have defined sets of axioms that are satisfied only by measures at the ratio level of measurement [9, 1].

At any rate, the hierarchical mechanism can also be used in a "bottom-up" fashion, i.e., starting from axiom sets for measures whose measurement level is nominal and strengthening them to build axiom sets for measures whose measurement level is ordinal, interval, and ratio. This process was used in several disciplines in the past. For instance, as Stevens points out (see [7]), the progress in the understanding of the temperature attribute of objects started from realizing that there are objects with the same temperature and objects with different temperatures, which implies measures for temperature with a nominal measurement level. Then, temperatures were ordered, i.e., it was possible to ascertain whether an object was more or less warm than another object, which implies measures whose measurement level is ordinal. Later, it was possible to measure distances between temperatures (i.e., with Celsius or Fahrenheit degrees), which implies measures on an interval level of measurement. Last, progress of Thermodynamics made it possible to obtain ratio level measures for which proportions were meaningful (i.e., Kelvin degrees). This process can be used to start with an axiom set upon which a wide consensus exists (for instance, Ax3 in Figure 2). New axiom sets are then added over time (e.g., Ax5 in Figure 2) and/or the existing properties are refined, so a new axiom set is obtained (e.g., Ax2 in Figure 2). The refinement must be carried out in such a way that the new axiom set defines a subset of the measures defined by the original axiom set, i.e., the new axiom set implies the old one. This new axiom set provides a better, more refined understanding of the attribute. However, the introduction of the new axiom set does not eliminate the starting axiom set of properties. The two sets are useful for different purposes, i.e., to define measures on different measurement scales.

This hierarchical axiomatic approach to the definition of measures for software attributes can also be used to identify the measurement level at which one is willing to accept a measure for a software attribute. For instance, suppose that one chooses the above linear order A: Ax_1, Ax_2, Ax_3, as representative of his or her own understanding of the size attribute. Suppose that a measure satisfies the axiom set Ax_2 but not the axiom set Ax_1. Based on the linear order A, one can accept that measure as a size measure at the ordinal level of measurement.

4. Conclusions

This paper provides a contribution towards the generalization of axiomatic approaches for the definition of measures for software attributes in two ways:

- It generalizes the framework in [1] by considering n-ary relationships between system and module elements. This provides a new degree of freedom, which can be used to better refine and formalize the knowledge about a software attribute.
- More importantly, the proposal of the paper reconciles, in a clear and practical manner, the theory of measurement levels and axiomatic approaches to software measurement by proposing a hierarchical axiomatic framework where hierarchical levels map to levels of measurement.

Based on this, at each level of measurement, one may select an optimal set of axioms to both define measurement attributes and evaluate candidate measures for each attribute. If one questions some of the axioms at a given level, it is always possible to resort to lower levels of measurement and adopt their weaker axiom sets.

In future work, we will investigate how the use of relationships with more than two elements affects other software attributes, e.g., complexity, cohesion, by allowing the definition of new axioms. We will also investigate the possibility of introducing new axioms for size and coupling, if needed. This will make it possible to better refine our knowledge about software attributes. This step will allow us to define hierarchies of sets of axioms for software attributes that are based on sets of axioms that are more refined, and therefore more informative, than those where only binary relationships are possible.

Acknowledgments

This work was supported in part by CNR, MURST, and Fraunhofer Gesellschaft.

References

[1] L. Briand, S. Morasca, and V. Basili, "Property Based Software Engineering Measurement," IEEE Transactions on Software Engineering, vol. 22, no. 1, pp. 68-86, Jan. 1996.

[2] N. Fenton, "Software Metrics, A Rigorous Approach," Chapman&Hall, 1991.

[3] B. Kitchenham, S. Pfleeger, and N. Fenton, "Towards a Framework for Software Validation Measures," IEEE Transactions on Software Engineering, vol. 21, no. 12, pp. 929-944, Dec. 1995.

[4] B. Kitchenham, S. Pfleeger, and N. Fenton, "Reply to: Comments on "Towards a Framework for Software Validation Measures,"" IEEE Transactions on Software Engineering, vol. 23, no. 3, pp. 189, Mar. 1997.

[5] K. B. Lakshmanan, S. Jayaprakash, and P. K. Sinha, "Properties of Control-Flow Complexity Measures," IEEE Transactions on Software Engineering, vol. 17, no. 12, pp. 1289-1295, Dec. 1991.

[6] S. Morasca, L. Briand, V. Basili, E. Weyuker, and M. Zelkowitz, "Comments on "Towards a Framework for Software Validation Measures,"" IEEE Transactions on Software Engineering, vol. 23, no. 3, pp. 187-188, Mar. 1997.

[7] F. Roberts, "Measurement Theory, with Applications to Decisionmaking, Utility, and the Social Sciences," Addison-Wesley, 1979.

[8] W. Reisig, "Petri Nets: an Introduction," ETACS Monographs on Theoretical Computer Science, Springer Verlag, 1985.

[9] E. Weyuker: "Evaluating Software Complexity Measures," IEEE Transactions on Software Engineering, vol. 14, no. 9, pp. 1357-1365, Sept. 1988.

[10] H. Zuse, "Software Complexity: Measures and Methods," De Gruyter, 1991.