DesCOTS: QM Conceptual Model

Xavier Franch, Gemma Grau, Carme Quer, Xavier Lopez-Pelegrín, Juan P. Carvallo

Universitat Politècnica de Catalunya (UPC)

c/ Jordi Girona 1-3 (Campus Nord, C6) E-08034 Barcelona (Catalunya, Spain)

email: {ggrau, franch, cquer, carvallo}@lsi.upc.edu

1. A conceptual model driven presentation of QM

The conceptual model can be divided into nine different parts that will be detailed in the rest of the paper, from sections 2 to 10:

- Arranging quality models in a taxonomy of categories and domains
- Classifying general objects
- Defining and grouping of quality entities
- Defining hierarchies of quality entities
- Defining metrics
- Assignment of metrics to quality entities
- Establishing relationships among quality entities
- Defining requirement patterns
- Defining a glossary

2. Arranging quality models in a taxonomy of categories and domains

A taxonomy is used for the organization of *Quality Models*. This taxonomy is composed of *Categories* (communication infrastructure, collaboration software, etc.) and *Domains* (workflow systems, mail servers, anti-virus tools, etc.), generalized as *Quality Scopes*. Domains are grouped into categories and categories on their turn are grouped into other categories. One domain or category may be part of more than one category. We attach quality models to quality scopes in the taxonomy, supporting model reuse by inheriting them downwards the hierarchy. We allow the construction of quality models for each scope in the taxonomy. The taxonomy just has a root, that is the *Generic* node.

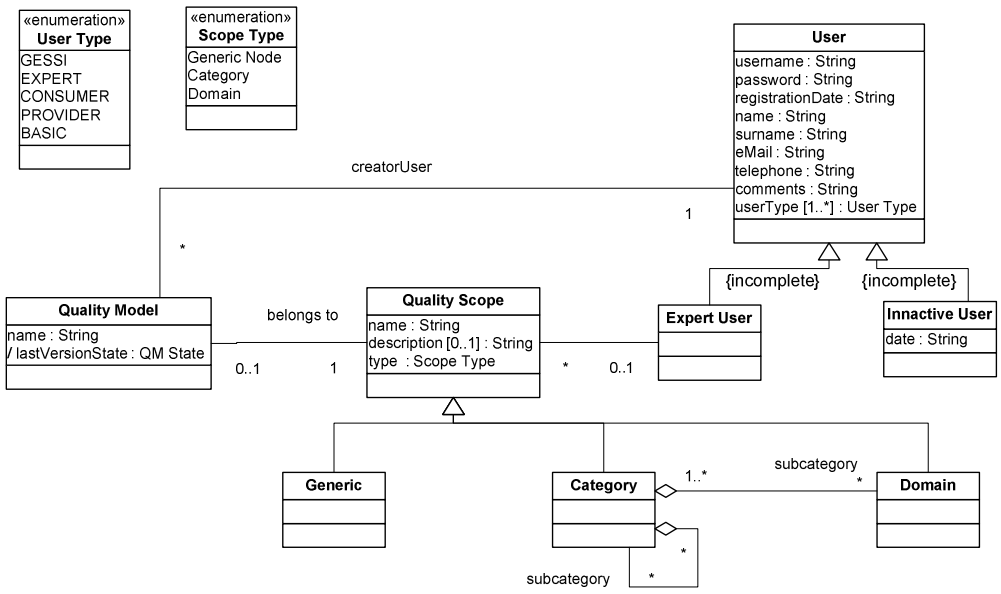

Fig. 1. Arrangement of quality models in a taxonomy.

The *User* that constructs a quality model corresponding to a quality scope, is an *Expert User* of the DesCOTS system. In case, the expert user changes we are interested in knowing the user that created it (*creatorUser*).

3. Classifying general objects

Since we want to maintain some common attributes for different elements that are managed by QM. We introduced the class *Object* in order to generalize this fact. The common attributes are two strings that correspond to a *description* and several *comments*, and the *Sources* (bibliographic source, commercial tool, tutorial, web page,…) from were the element has been seen that needs to be part of the QM repository. The elements can be elements specific of a quality model, or may be elements that are general for the construction of any quality model (for example we will have global metrics that will be shared in the construction of all quality models). For this reason we differentiate among *Global Sources* (that are sources of elements interesting for the construction of any quality model) and *Domain Sources* (that are sources of elements related with one quality model).

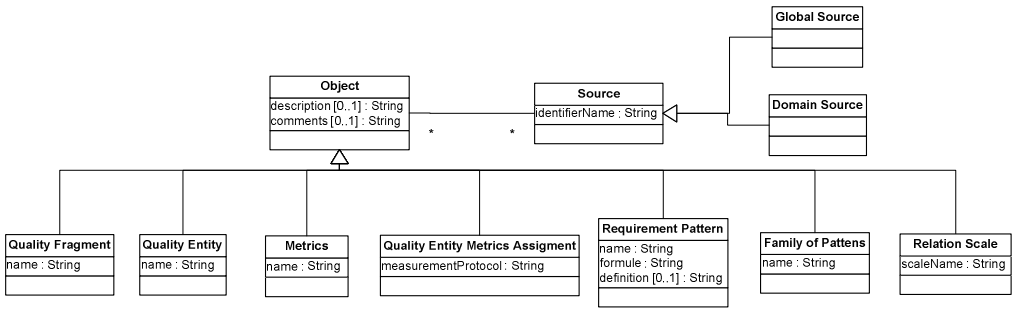

Fig. 2. Generalization of common attributes for QM elements.

General objects are *Quality Pieces, Quality Entities, Metrics, Quality Entity Metrics Assignments, Requirement Patterns, Families of Patterns and Relation Scales.*.

4. Defining and grouping of quality entities

Quality models contain quality entities customized to a particular *Quality Scope*. More than one *Quality Model Version* of a quality model may be defined. An order is necessary among the versions of a quality model (*{ordered}*). In order to define a new version the previous version has to be validated (*validationDate*), and the *lastVersionState* of the quality model has to be *VALIDATED*.

Aside of quality models, QM also deals with other *Quality Fragments. Quality entities* are grouped into quality fragments. Quality entities are related to quality fragments in two different ways. Quality entities may be copied from one fragment to another. If this is done, they are replicated, and each replica is a different quality entity that belongs to a different quality fragment. Thus, quality entities are not shared among quality fragments. This decision makes changes on quality entities to be local, making easier quality model management. One quality model version may have more than one quality entity *root*, since all quality entities in the first level of the quality model (*Characteristics*) will be roots of the quality model.

The other type of quality fragment are *Quality Patterns*. These patterns are chunks of quality entities that appear in many quality models. QM promotes reusability by allowing the construction and maintenance of a catalog of quality patterns. This catalog may be accessed by all the expert users (during the construction of quality models), and thus these users can share quality patterns as a global element of QM.

Elements that are specific of a quality model are *Quality Model Specific Metrics*, *Requirement Patterns*, *Families of Patterns* and *Domain Sources*.

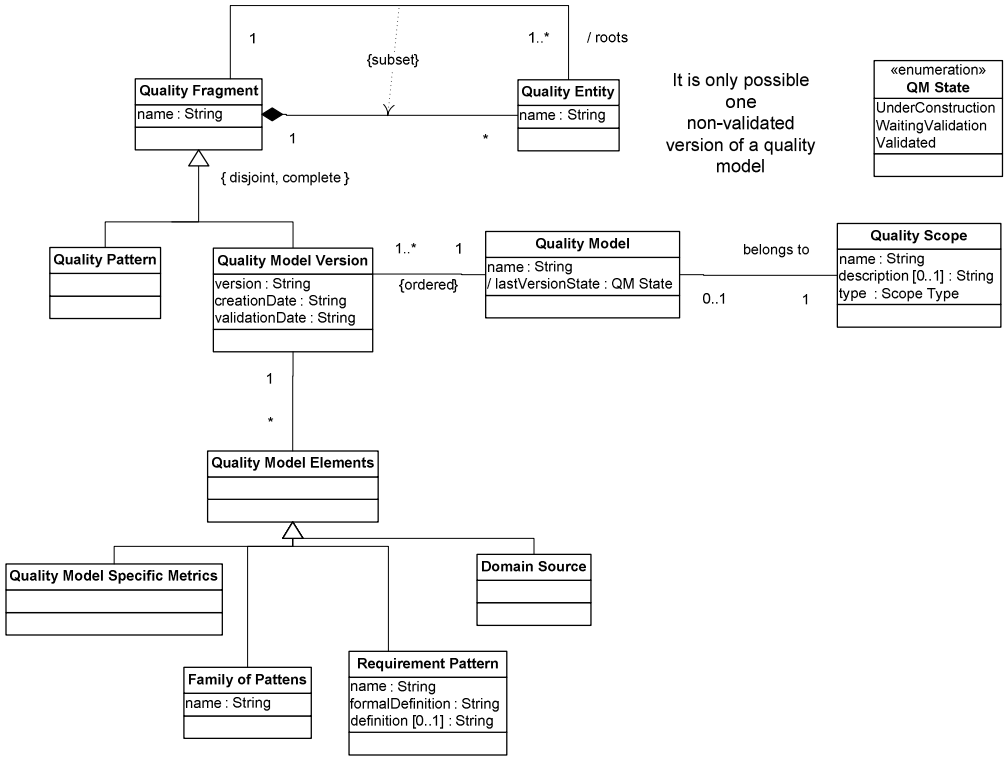

Fig. 3. Quality Fragments: quality models and quality patterns.

5. Defining hierarchies of quality entities

As the cornerstone of our quality framework, we have chosen the ISO/IEC 9126-1 standard. It is quite generic, presents a hierarchical structure and is widespread. So we have included in our conceptual model its three types of *Quality Entities*: *Characteristics*, *Subcharacteristics* and *Attributes*. The ISO/IEC 9126 standard is not precise enough in some points and, therefore, some decisions have been taken and reflected in the conceptual model. The two most important ones are:

- Hierarchies of subcharacteristics and attributes are allowed without any restriction about its number of levels.
- An attribute or subcharacteristic may be associated to several subcharacteristics, as the standard does not forbid overlapping of software entities.

We have also introduced an extra classification hierarchy for quality entities, which indicates if an entity is *Specific Quality Entity* or *Generic Quality Entity*, i.e. if it is fully defined or not. Thus, we also have specializations for each type of specific quality entity, obtaining: *Specific Characteristics*, *Specific Subcharacteristics* and *Specific Attributes*. This specialization is introduced in order to model that just specific entities may be decomposed and that metrics can be only defined for specific subcharacteristics and attributes.

One textual restriction has to be defined in order to restrict the use of specific quality entities into quality models fragments; they cannot appear into quality patterns. On the contrary, generic quality entities facilitate reusability among quality models, because they hide those details that

are bound to particular context. Since they are not bound to any particular quality model, they may be included and tailored to the specific needs of several quality models. Thus this type of entity may appear in any type of quality fragment. However, a quality model cannot be considered complete if it still contains generic quality entities.

Specific subcharateristics are named *Derived Subcharacteristics* if they are decomposed in a new level of subcharacteristics. On the contrary, if they are decomposed into attributes, they are named *Basic Suncharacteristics*. Also, specific attributes, that are decomposed in a new level of attributes are named *Derived Attributes*. And attributes that are not plus decomposed are named *Basic Attributes*.

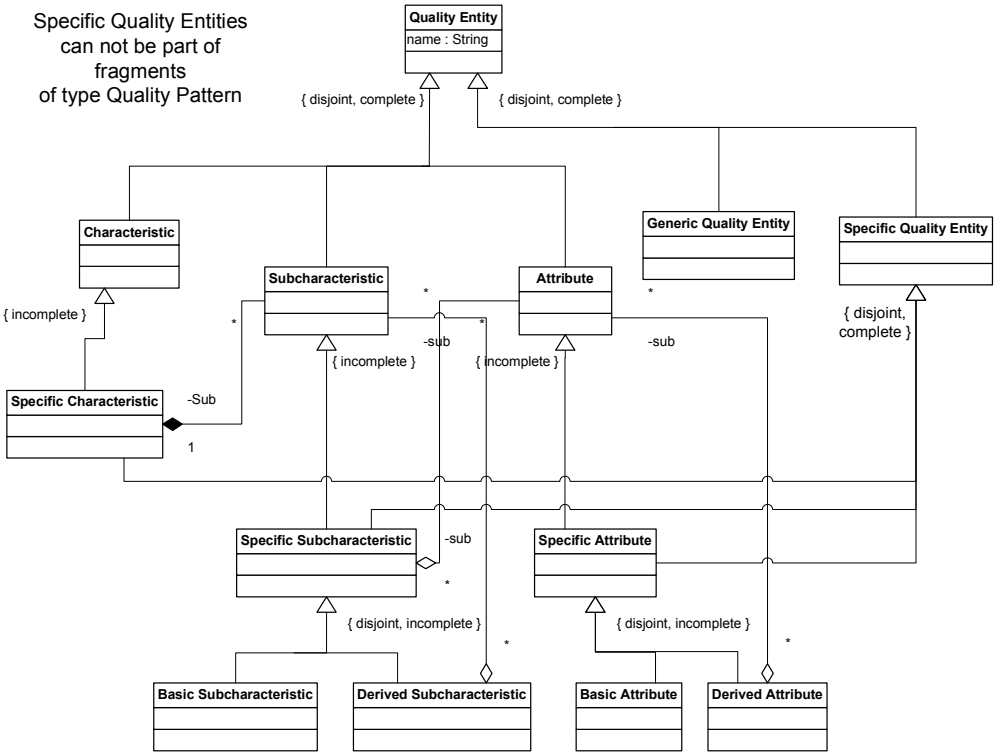

Fig. 4. Quality entities hierarchy.

6. Defining metrics

QM manages two types of *Metrics*: the *Global Metrics* that are general to all quality model constructions, and the *Quality Model Specific Metrics*, that are metrics that correspond to an specific quality model version. Metrics of each of these types may be replicated to the other type. This means that a user may decide that a metrics defined as specific, may be useful as a global metrics, or the other way round. If this is done they are just copies and may be changed independently.

We may also classify metrics as *Qualitative Metrics* and *Quantitative Metrics*. A metrics is qualitative when it is not possible to establish a precise, non-ambiguous measurement procedure to get the value of the quality entity that it evaluates, but it is possible to give an appreciative value (subjective). Otherwise, the metric is quantitative. Quantitative metrics usually catch observable quality factors of software products (Number of Retransmission Retries before

Failure,…). As a desirable property, the measurement procedure for objective metrics should be always repeatable and should give the same results.

Qualitative metrics always have a metrics of type string.

Quantitative metrics may be a *Formula* or a *Basic Metric.* A quantitative metrics is basic when its value must be assigned directly by a software quality expert. Otherwise, the metrics is a formula, defined by means of a *formalDefinition.* A formula will be based on the quality entities from which the value is calculated. Since a quality entity may be evaluated by different metrics depending on the quality entity that they are decomposing, it is necessary to relate the formula with the *Quality Entity Metrics Assigment,* that is the assignment of the metrics to the quality entity (the assignment corresponds to a quality entity used in the formula, the predecessor quality entity that it is decomposing and the metrics of the quality entity when it is decomposing the predecessor quality entity).

Basic Metrics may be *Simple*, *Set*, *Tuple* or *Function.* Simple metrics are *Boolean*, *String* (for them it is possible to establish a *defaultValue*), *Integer* (for them it is possible to establish a *max* value and a *min* value), *Real* (for them it is possible to establish a *max* value and a *min* value), or *Domain* (they may be *Ordered Domains* or not, depending on if it is possible to establish an order among the *Domain Values*). The *Set Elements* must have a simple metrics. The *Tuple Elements* may have a simple metrics (*Simple Tuple Elements*) or a set metrics (*Set Tuple Elements*). Tuple elements must have a *name* to identify them. The *Function* metrics may have more than one *Input Parameter* and just one *Output Parameters* that may have any type of simple metrics but function (see textual restriction).

Finally, in order to facilitate the definition of the quality entity metrics assignment (see the next section), we have needed to add one generalization of qualitative metrics and formulas in *Derived Metrics*, since they will be the type of metrics suitable for derived quality entities.

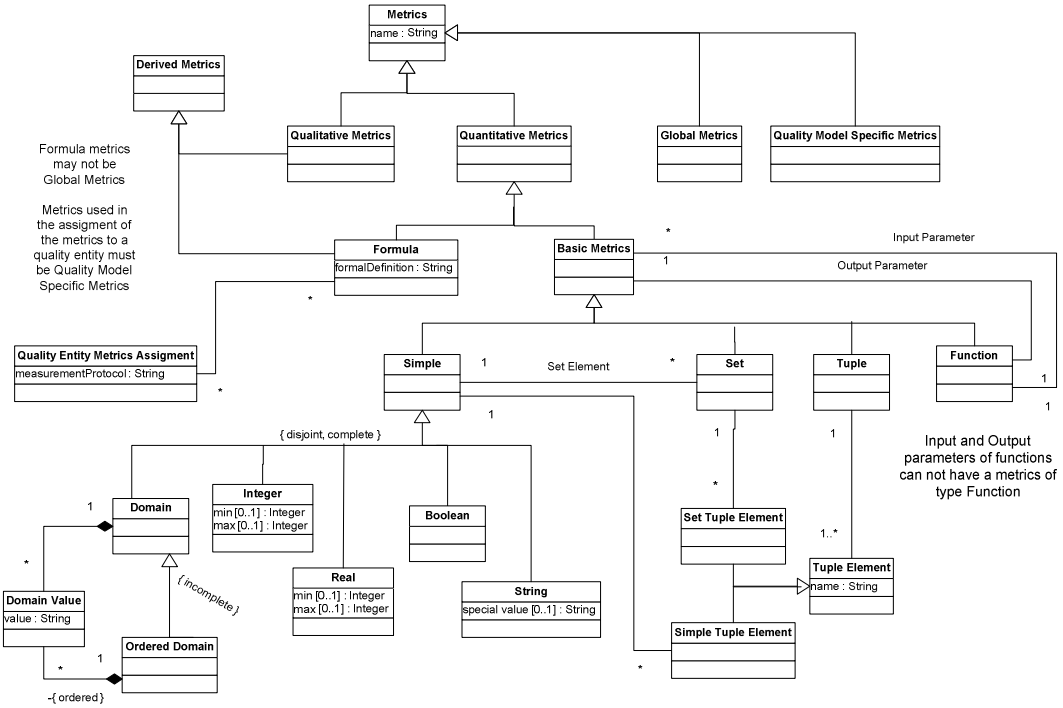

Fig. 5. Metrics classification.

7. Assignment of metrics to quality entities

To allow the evaluation of products in a domain following a quality model, we need to assign metrics to quality entities. *Quality Entity Metrics Assignment* are elements of a quality model, and the assignment of a metrics to quality entities of a quality model version must be quality model specific metrics (see the previous section).

No all the types of metrics are suitable for each type of quality entity. For this reason we have specialized the quality entity metrics assignment depending on the type of quality entity to which the assignment corresponds.

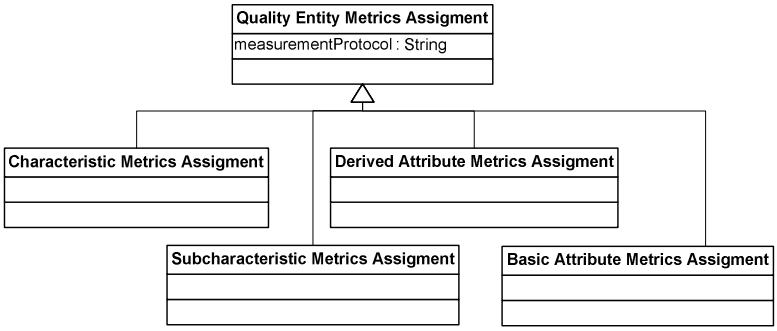

Fig. 6. Characteristic metrics assignment

Next, we describe the relationships that are involved in the assignment depending on the type of quality entity:

- Characteristics are used as a classification level which groups the different quality entities related with it. Most of times they are not evaluated, however we allow to assign them a qualitative metrics. Thus the *Characteristic Metrics Assignment* is an association class among a qualitative metrics and the characteristic to which the metrics is assigned.

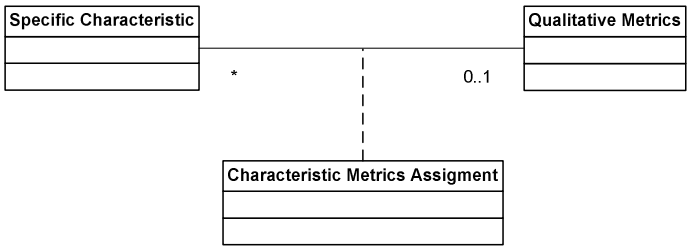

Fig. 7. Characteristic metrics assignment

- Subcharacteristics can be measured by derived metrics. Taking into account that a subcharacteristic may decompose more than one subcharacteristic (having different metrics in each decomposition), the *Subcharacteristic Metrics Assignment* is an association class among a derived metrics, the subcharacteristic to which the metrics is assigned, and the quality entity that it is decomposing in this assignation.

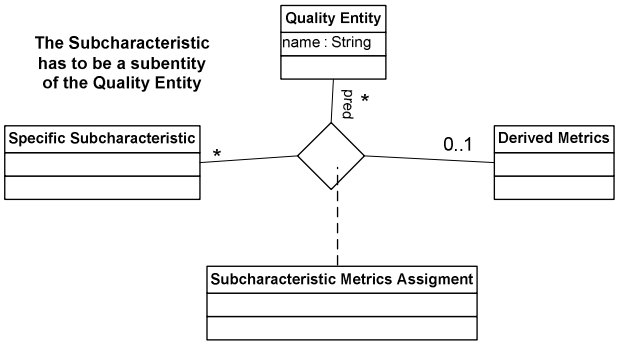

Fig. 8. Subcharacteristic metrics assignment.

- Derived attributes can be measured by derived metrics. Taking into account that an attribute may decompose more than one subcharacteristic or derived attribute (having different metrics in each decomposition), the *Derived Attribute Metrics Assignment* is an association class among a derived metrics, the attribute to which the metrics is assigned, and the quality entity that it is decomposing in this assignation.

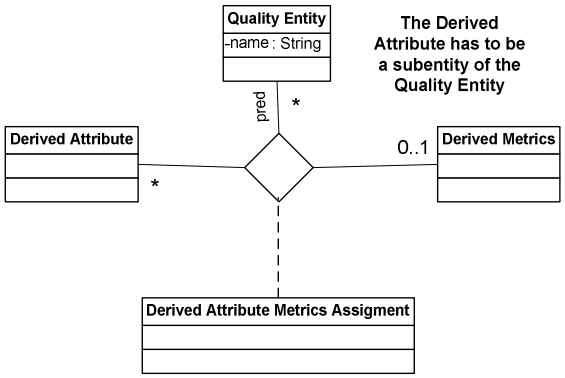

Fig. 9. Derived attributes metrics assignment.

- Basic attributes can be measured by basic metrics. Taking into account that an attribute may decompose more than one subcharacteristic or derived attribute (having different metrics in each decomposition), the *Basic Attribute Metrics Assignment* is an association class among a basic metrics, the attribute to which the metrics is assigned, and the quality entity that it is decomposing in this assignation.

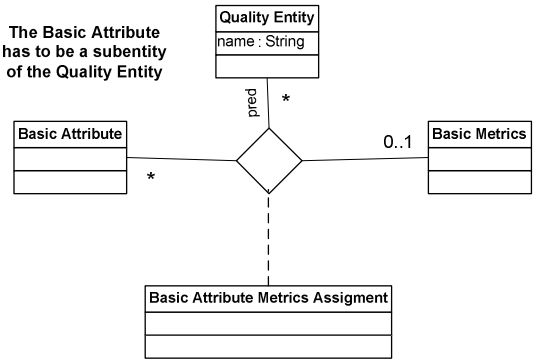

Fig. 10. Basic attributes metrics assignment.

8. Establishing relationships among quality entities

It is possible to establish relationships among *Quality Entities* of a quality model version. These relationships play a fundamental role when analysing software requirements expressed in terms of quality entities. For instance, consider the requirements "The system shall encrypt personal data" and "The system shall provide optimal response time when retrieving a personal data record". Both of them involve quality factors, Encryption Strategy Used and Response Time, respectively. A complete quality model should state that these two quality attributes are conflicting, which also makes the requirements mutually conflicting.

The *Relation Scale* consist of the set of possible values (*Scale Elements*) of the scale. Two examples of relation scales (with 3 and 5 scale elements) are:

IQMC-scale = (depends, conflicts, collaborates), from [FC03].

NFR-scale = (make, help, unknown, hurt, break), from [CNYM00].

Thus, two quality entities may be related by an scale element in a relation scale, and it is also possible to have the intensity of the relationship by means of the *intensity* attribute of the association class *QER Degree* (quality entity relationship degree) included for the relationship. The intensity is not a mandatory value for the association since it has not sense for all the relation scales (it has sense in the *IQMC-scale* above but not in the *NFR-scale*).

Finally, taking into account that it is possible to have refinement of relationships, that is, that relationships among subcharacteristics, then are refined in relationships among the attributes in which these subcharacteristics are decomposed, there is a reflexive association among quality entity relationship degrees.

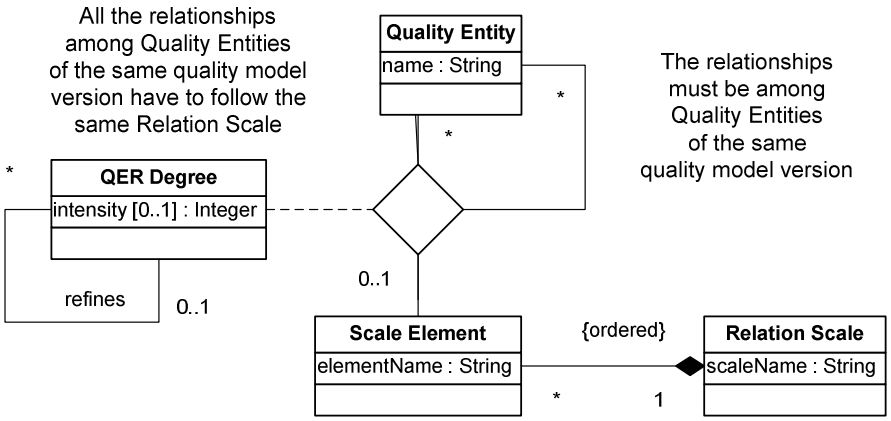

Fig. 11. Relationships among quality entities.

9. Defining requirement patterns

A *Requirement Pattern* provides a template that can be tailored to specific contexts to generate software requirements. Requirement patterns are elements of quality model versions. Thus, they are always associated to a quality model version.

Requirement patterns are classified in *Families of Patterns*, although it is possible to have requirement patterns that are not assigned to any family. It is also possible to have a hierarchy of families.

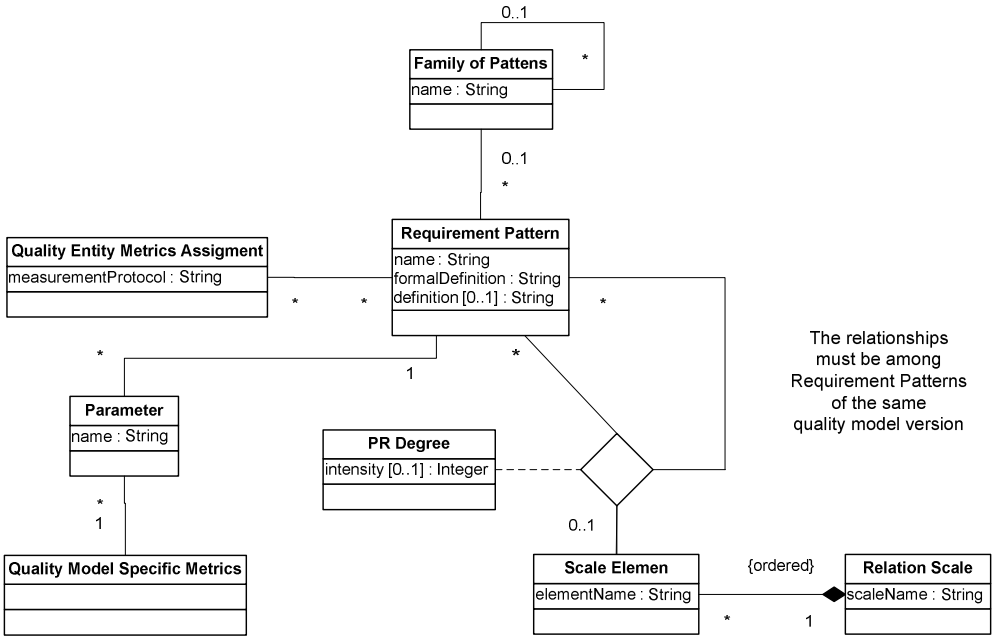

Fig. 12. Requirement patterns.

The specific information managed for requirement patterns are a *formalDefinition* and also another narrative *definition*. In both of them may appear *Parameters*, that will be filled when the requirements will be used in a selection process. In order to know which values may take these parameters, the *Quality Entity Specific Metrics*.

The formal definition is a formula that when evaluated must have a boolean value. It is based on the quality entities from which the Boolean value is calculated. Since a quality entity may be evaluated by different metrics depending on the quality entity that they are decomposing, it is necessary to relate the formula with the *Quality Entity Metrics Assigment,* that is the assignment of the metrics to the quality entity (the assignment corresponds to a quality entity used in the formula, the predecessor quality entity that it is decomposing and the metrics of the quality entity when it is decomposing the predecessor quality entity).

It is possible to establish relationships among requirement patterns of a quality model version. The relationship follow the same *Relation Scale* than the quality entities (see the previous section). Thus, two requirement patterns may be related by an *Scale Element* in a relation scale, and it is also possible to have the intensity of the relationship by means of the *intensity* attribute of the association class *PR Degree* (pattern relationship degree) included for the relationship. The intensity is not a mandatory value for the association since it has not sense for all the relation scales.

10. Defining a glossary

A glossary helps QM to have definitions of terms used in the construction of quality models. There is just one global glossary that manages common *Glossary Terms* shared by all *Users*.

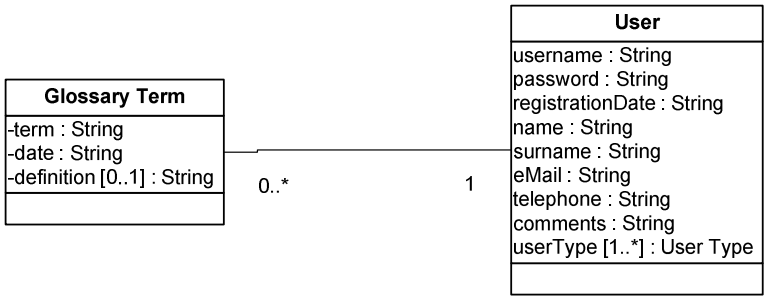

Fig. 13. Glossary