Inducing Metaassociations and Induced Relationships[†]

Xavier Burgués[1], Xavier Franch[1], Josep M. Ribó[2]

[1]Universitat Politècnica de Catalunya. J. Girona 1-3, Campus Nord. 08034 Barcelona, Spain.
[2]Universitat de Lleida. Jaume II 69. 25001 Lleida, Spain.
{diafebus, franch}@lsi.upc.edu
josepma@diei.udl.cat

Abstract. In the last years, UML has been tailored to be used as a domain-specific modelling notation in several contexts. Extending UML with this purpose entails several advantages: the integration of the domain in a standard framework; its potential usage by the software engineering community; and the existence of supporting tools. In previous work, we explored one particular issue of heavyweight extensions, namely, the definition of inducing meta-associations in metamodels as a way to induce the presence of specific relationships in their instances. Those relationships were intended by the metamodel specifier but not forced by the metamodel itself. However, our work was restricted to the case of induced associations. This paper proposes an extension to the general case in which inducing metaassociations may force the existence of arbitrary relationships at M1. To attain this goal, we provide a general definition of inducing metaassociation that covers all the possible cases. After revisiting induced associations, we show the inducement of the other relationship types defined in UML: association classes, generalization and dependencies.

Keywords: UML, MOF, Metamodels.

1. Introduction

In the last years, we may find several contexts in which UML [1, 2] has been tailored to be used as a domain-specific modeling notation. For instance, we may mention extensions to model data warehouses [3], software processes [4], real-time issues [5], etc. Extending the UML with this purpose entails several advantages: the integration of the domain in a standard framework; its potential usage by the software engineering community; and the existence of supporting tools.

Two different strategies may be adopted in order to extend UML:

- *Lightweight extensions*, which create a UML profile with the UML standard extension mechanisms provided in the profiles package (e.g., [6]).
- *Heavyweight extensions*, which enlarge the UML metamodel, creating a new metamodel specific for the target domain (e.g, [3, 4, 7]).

[†] This work has been partially supported by the Spanish project TIN2007-64753.

In this paper, we are interested in heavyweight extensions, that take place at the M2 level of the MetaObject Facility (MOF) Specification [8], where the UML metamodel is placed. Modifications to this M2 level impact on the form that UML models, located at the M1 MOF level, may take (and transitively on the possible model instances that conform to the M0 MOF level).

In [9], we explored one particular issue of heavyweight extensions, namely, the lack of expressive power that most metamodeling approaches have for building M2 metamodels that force a specific association in their model instances (at level M1). To overcome this limitation, we introduced the notions of *inducing metaassociations* and *induced associations*. In short, induced associations are those associations in a UML model whose existence is implied by a specific kind of metassociations (which are tagged as "inducing") that have been included in a UML metamodel extension. In that paper, we defined formally the notions of inducing metaassociation and induced association, we analyzed how several other UML constructs (like adornments and subsettings) were affected by these definitions, and we presented a method for introducing induced associations in a UML model based on tagging the appropriate metaassociations as inducing metaassociations. We explored the feasibility of the proposal in a complex case study for building a generic quality model as an extension of the UML metamodel (as proposed in [10]).

Once the concepts of induced associations and inducing metassociations were defined, the next natural step is to induce other kinds of UML relationships that are also needed in the process of UML metamodel extension. This need arose also in the heavyweight metamodel extensions that we have built. In this paper, we tackle the inducement of the other type of relationships defined in the UML metamodel [1, 2]: association classes, generalization relationships and dependencies. To avoid working in a case-by-case basis, we rephrase the notion of inducing metassociation in a way that it may induce all these relationship types, as well as induced associations as defined in [9], and eventually others that could arise in the future.

The rest of the paper is structured as follows. In Section 2, we revisit the problem of induced associations as explored in [9], which provides the necessary background to understand the rest of the paper. Then we present the general definition of inducing associations in Section 3 and explore its application in Section 4. Section 5 deals with the combination of induced elements and the presence of generalizations in models at M1. We finish the paper in Section 6 with the conclusions and future work.

2. Background

To illustrate the problem, let's consider the definition of a metamodel for quality aspects of software as presented in [10]. Such a *quality metamodel*, located at M2, is defined as a heavyweight extension of the UML metamodel and is responsible for defining the generic concepts that come up in the definition of a quality model and the relationships between these concepts. Each particular quality model will be defined as an instance in M1 of the quality metamodel. For example, the quality model ISO-9126 [11] could be defined as an instance of the quality metamodel.

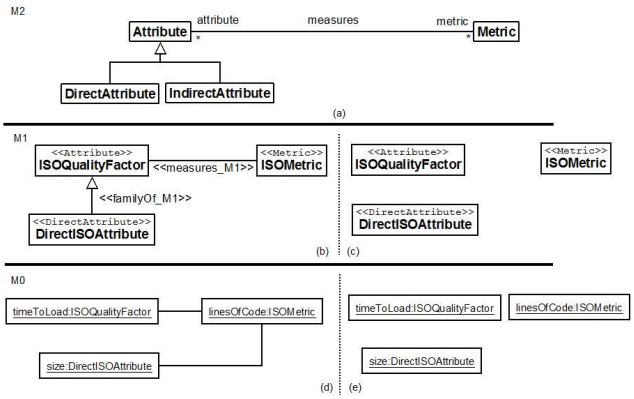

Fig. 1. The intention (b, d) and a possible non-intended result (c, e) of the instantiation of a metamodel for quality (a).

This quality metamodel contains, among others, the metaclasses *Attribute* and *Metric*. *Attribute* represents the quality aspects that are to be measured by a specific model. *Metric* represents the element used to measure an attribute. A metaassociation *measures* between both metaclasses is defined in the metamodel. Also, we consider the existence of two types of attributes, direct attributes whose value is computed from direct observation of a software artefact, and indirect attributes, computed from other attributes' values. Fig. 1(a) presents this metamodel fragment.

Because of its semantics, the metaassociation *measures* is intended to be an inducing metaassociation: when a model instance of this quality metamodel is defined (thus, at level M1), for each pair of instances of *Metric* and *Attribute* that belong to the extension of *measures*, an association should come up at M1. Fig. 1(b) shows two instances of these two metaclasses for the ISO/IEC 9126 quality model, *ISOQualityFactor* and *ISOMetric*, that belong to *measures*' extension, and the association between them. As a consequence, classes and relationships among them (in particular, associations) defined in M1 are eventually instantiated by objects and links between them when that model is instantiated (level M0). In Fig. 1(d), we show how a specific instance of *ISOMetric*, named *linesOfCode*, can be used to measure a specific instance of *ISOQualityFactor*, named *timeToLoad*.

A similar reasoning applies to the specialization relationship that comes up in the metamodel. In particular, note in Fig. 1(d) that instances of *DirectISOAttribute* like *size* may be linked to *linesOfCode*, due to the M1 induced inheritance relationship.

However, a careful analysis reveals that the metamodeller intentions are, in fact, not really represented in the metamodel. The presence of the association at level M1 (which was meant by the metamodeller) is not implied by the semantics of the metamodel and, hence, is left to the modeller skills. As a result of this limitation, the M1 model may not convey all the information that was meant by the M2 metamodel which it is an instance of, leading to incompleteness and inaccuracies. This situation is shown in Fig. 1(c), and the effects on M0 are shown in Fig. 1(e), where no links between the instances appear. Furthermore, even if the association was correctly

added, traceability is seriously damaged since no explicit link is established with the metaassociation at M2. This is what may happen if the issue is not taken into account or if, as is done in the Unified Process [12], stereotypes are attached to associations at layer M1 with no connection with the metamodel at M2.

In [9] we addressed this limitation by providing a technical solution for the case of inducing metaassociations. As shown in Fig. 2(a), metaassociations may be forced to be inducing by attaching an appropriate OCL constraint that connects it with a new class, declared as heir of the *Association* metaclass. As a consequence, now at M1 induced associations appear stereotyped with the metaclass name (Fig. 2(b)). Therefore, instances at M0 may be connected again as intended (Fig. 2(c)).

Whilst Fig. 2 shows our solution to the case of inducing metaassociations as given in [9], it also reveals the limitations of the approach: no other inducing metaelements have been considered. Therefore, the specialization declared at level M2 that is also intended to be inducing, is not covered by our solution, then it is not possible to force M1 models to contain those needed inheritance relationships, being thus the non-intended situation illustrated at Fig. 2(b) possible to occur. As a consequence, it may not be possible to link instances of *DirectISOAttribute* with instances of *ISOMetric* (i.e., to establish metrics for direct attributes) which of course was not intended by the metamodeller. The same would happen for any other type of metamodel element except metaassociations. The purpose of this paper is then to further refine the proposal given in [9] for avoiding situations like this in Fig. 2(b, c).

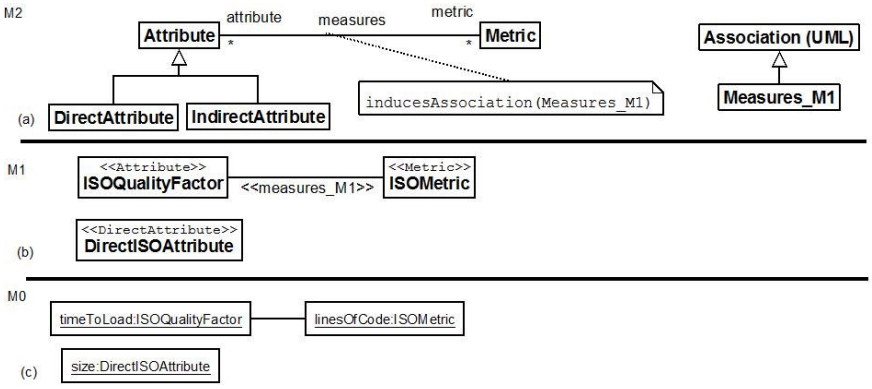

Fig. 2. Declaring inducing metaassociations at M2 and the consequences in the lower levels.

3. Relationship-inducing Metaassociations

In this section, we generalize the idea of inducing metaassociation for induced associations presented in [9] to allow inducing metassociations to induce all type of relationships at M1. We call them *relationship-inducing metaassociations*.

Let *EM* be an extension of the UML metamodel (at layer M2) containing three metaclasses *MC1*, *MC2* and *MR* and a metaassociation *M2A* between *MC1* and *MC2*.

In order to make the pair (*M2A*, *MR*) induce relationships (i.e., associations, association classes, generalizations or dependencies) at layer M1, the next procedure shall be followed (see Fig. 3):

a) Add to *EM* a generalization relationship from *MR* to one heir of the UML *Relationship* metaclass: *Association*, *AssociationClass*, *Generalization* or *Dependency*.
   - The instances of *MR* will constitute *relationships induced* by (*M2A, MR*).
   - If *MR* is a subclass of the *DirectedRelationship* UML metaclass (i.e., *MR* is *Generalization* or *Dependency*), *M2A* should be a unidirectional metaassociation and the *source* and *target* of the induced directed relationship are denoted by the navigability sense of *M2A*.
   - (M2A, MR) constitutes a relationship-inducing metaassociation pair, where "relationship" refers to the specific relationship type induced. For short, M2A or MR can also be referred to as inducing metaassociation resp. relationship.

b) Add to *EM* a constraint attached to *MR* establishing that for all instances *C1* of *MC1* and *C2* of *MC2* s.t. <*C1, C2*> is in *M2A*'s extension, there is an instance of the relationship *MR* connecting *C1* and *C2* (and vice versa). The sense of the connection in the case of *DirectedRelationships* is given by the navigability sense of *M2A*.

c) Make *MR* subclass of *InducedRelationship*.

In order to generate a more structured metamodel, we have introduced a new *InducedRelationship* metaclass that roots the hierarchy of M1-relationships that are induced by M2 metaassociations. Hence, each subclass of *InducedRelationship* is also an heir of the appropriate subclass of *Relationship*, according to its type. The following constraint holds: "for each subclass *MR* of *InducedRelationship* there is a metaassociation *M* such that all the *M1*-relationships which are instances of *MR* will be induced by *M"*. This idea can be expressed by means of the OCL-helper *InducesRelationship(anyMA: Association)* which is defined in the context of *InducedRelationship* as shown in Fig. 4. It states the following:

(a) There will be an instance of the inducing relationship *self* binding each pair of classes that are linked by the extension of the metaassociation *anyMA*. This is stated by the part 1 in the case of directed relationships (the undirected case is similar to the directed and not included here for reasons of space).

(b) All instances of *self* are meant to be relationships induced by the extension of the metaassociation to which *self* is bound (*anyMA*). This is shown by the part 2 in the case of directed relationships.

Notice that, since all kinds of relationships are heirs of the *Relationship* class in the UML metamodel and the *inducesRelationship()* specification deals only with the features of (Directed)Relationship (in the UML metamodel), this operation covers the induction of any kind of such relationships (associations, association classes, generalizations and dependencies) from inducing metaassociations. If, in the future, some new relationship were added to the UML metamodel (or to a UML extension), this new type of relationship could be handled in the same way than the others.

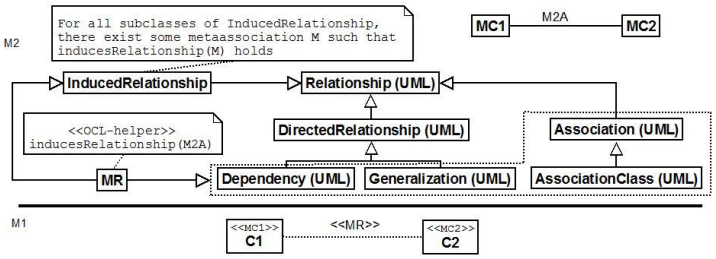

Fig. 3. Definition of relationship-inducing metaassociations

```
context InducedRelationship
def: inducesRelationship(anyMA: Association): Boolean =
      let MC1: Class = anyMA.memberEnd->at(1).class,
          MC2: Class = anyMA.memberEnd->at(2).class in
      self.oclIsKindOf(Relationship) and
              (MC1.allInstances()->forAll(c1 |
                    MC2.allInstances()->forAll(c2 | c2.mc1 = c1
                          implies if (anyMA.navigableOwnedEnd->size() = 2)
                                       inducesNonDirectedRelationship(anyMA)
                                  else inducesDirectedRelationship(anyMA) endif)))

context InducedRelationship
def: inducesDirectedRelationship(anyMA: Association): Boolean
      let TargetEnd: Class = anyMA.navigableOwnedEnd->at(1),
          SourceEnd: Class = if (anyMA.memberEnd->at(1) = TargetEnd)
                                anyMA.memberEnd->at(2)
                             else anyMA.memberEnd->at(1) in
      self.oclIsKindOf(DirectedRelationship)
                and  -- PART 1
                   TargetEnd.allInstances()->forAll(c1 |
                   SourceEnd.allInstances()->forAll(c2 | c2.targetend = c1
                           implies self.allInstances()->exists(r |
                                       r.target = c1 and r.source = c2)))
                and -- PART 2
                   self.allInstances()->forAll(r |
                         r.source->size() = 1 and r.target->size() = 1 and
                         r.source->at(1).oclIsKindOf(SourceEnd) and
                         r.target->at(1).oclIsKindOf(TargetEnd) and
                         r.source->at(1).targetend = r.target->at(1))
```

Fig. 4. OCL representation of the *inducesRelationship* OCL-helper

4. Induced relationships

In this section we define four types of M1 induced UML relationships.

4.1 Induced associations

This is the case in which the relationship induced at level M1 by an M2-inducing metaassociation is an *association.* In this case, *MR* is a subclass of the UML

Association metaclass. As opposed to [9] and explained in detail in the previous section, now the heir of *Association* declares the inducement by calling the *inducesRelationship* operation with the inducing metaassociation as parameter.

Sect. 1 presented an example of a situation requiring an inducing meta-association. Fig. 5 shows the modelling of that inducing metaassociation together with the corresponding induced associations following the definition proposed in Sect. 3.

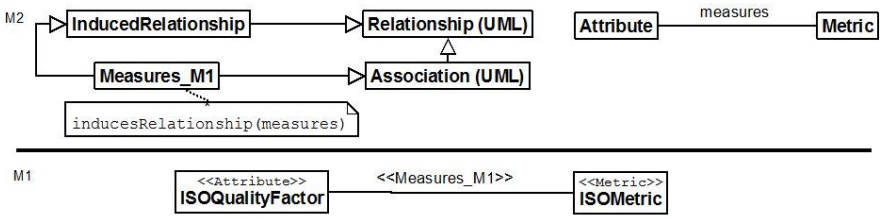

Fig. 5. Inducing metaassociations and induced associations

4.2 Induced association classes

Another particular case of relationship-inducing metaassociation takes place when the metaclass *MR* is, actually, a subclass of the UML *AssociationClass* metaclass (which, in turn, is a subclass of *Association*). In this case, the pair (*M2A, MR*) induces association classes at layer M1.

For induced association classes, the subclass *MR* of the UML metaclass *AssociationClass*, usually comes up as a metaclass that modelizes a (meta-)domain concept, while in the case of induced associations, *MR* usually models an association between (meta-)domain concepts.

Induced association classes constitute a common need in metamodeling situations. Consider, for instance, in the context of the quality metamodel, the metaclass *QualityModel* (which has been defined as a subclass of AssociationClass). The ontology proposed in [10] stated that quality models apply for a given software domain (e.g., the domain of business applications, or the software categories identified in a IT consulting company, …) and a given environment (e.g., public administration, SME, …). In that paper, this situation was modelled in M1 with an association class as shown in Fig. 6, where also some M0 instances are represented.

From the metamodelling perspective, this association class must be defined as induced, because it does not show up from scratch but from some metamodel concepts. Specifically, the quality metamodel includes the metaclasses *Domain* and *Environment*, and also a metaclass *QualityModel* for the association class itself. Finally, a metaassociation *usedIn* between *Domain* and *Environment* is introduced, and to make it association-class-inducing, *QualityModel* is declared as heir of *AssociationClass* with the usual constraint about inducement. As a result, the metamodeller has established that there is a different quality model associated to each specific (*Environment, Domain*) pair (see Fig. 6).

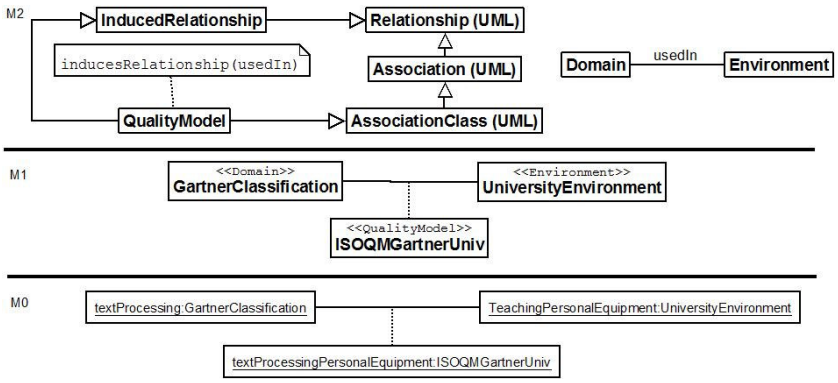

Fig. 6. Inducing metaassociations and induced association classes

4.3 Induced generalizations

This is the case in which the relationship induced at level M1 by an M2-inducing metaassociation is a generalization. In this case, *MR* is a subclass of the UML *Generalization* metaclass.

It is worth noting that *Generalization* is a subclass of the UML *DirectedRelationship*, which defines a non-symmetrical relationship from a source class to a target class. As it has been stated in the general definition, when a metaassociation induces a *DirectedRelationship,* it should be a unidirectional metaassociation directed from the metaclass whose instance acts as source (in generalizations, *subclass*) to the metaclass whose instance acts as target (in generalizations, *superclass*).

Although in our experience, induced generalizations are not as common as induced associations or induced association classes, there still exist situations in which it is interesting that the metamodel forces specific generalizations between the classes that are meant to instantiate it. One of those typical situations occurs if various groups of elements, each one belonging to a different family, are expected at M1. The metamodeller may want that the M1 instances of the metamodel make clear the separation between the different families and hence, force several (induced) generalizations.

Fig. 7 shows an example coming from the quality metamodel already outlined in Sect. 2. In this metamodel excerpt, three metaclasses come up which model the notions of *Attribute* (an element whose quality has to be measured), *DirectAttribute* (an attribute that can be measured directly) and *IndirectAttribute* (an attribute whose measure is obtained from that of other attributes). The inheritance relationships at M2 come from the fact that both direct and indirect attributes are themselves attributes and, hence, inherit its features. Obviously, this pair of generalization relationships does not imply any generalization relationship among their instances. With the generalization-inducing metaassociation *familyOf,* the metamodeller is stating that different families of attributes (instances of *Attribute*) may come up at M1. Each family will be composed of a group of specific attribute classes (instances either of

DirectAttribute or *IndirectAttribute*). The classes that model direct and indirect attributes corresponding to the same family will be linked by an induced generalization to the class (instance of *Attribute*) that represents that family. Two families are shown in Fig. 7: The ISO-9126 family and the SEI family. Notice that the extension of the *familyOf* metaassociation determines which are the induced generalizations (i.e., the attribute classes belonging to the same family).

As usual, the metaclasses bound by the *familyOf* associations are themselves linked by means of a generalization relationship at M2. Although this is the normal case, it is not a compulsory requirement. In some occasions, the metamodeller is not interested in bringing up that generalization relationship (or one of the involved metaclasses) because either it requires to model an artificial element or it does not provide any relevant information to the metamodel.

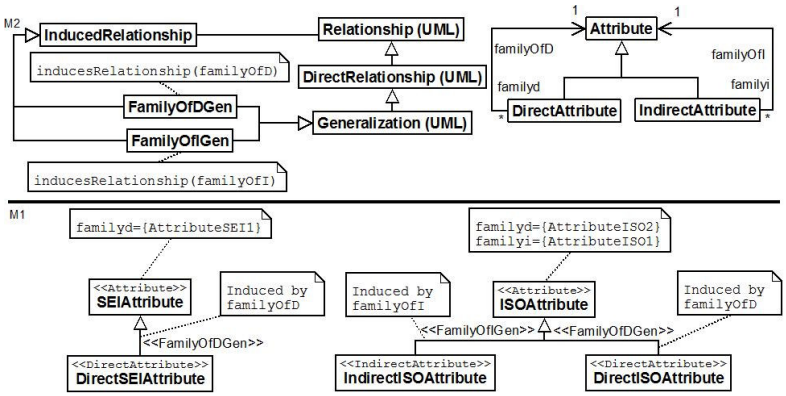

Fig. 7. Generalization-inducing metaassociations

4.4 Induced dependencies

When the relationship induced at level M1 by an M2-inducing metaassociation is a UML dependency, *MR* is a subclass of the UML *Dependency* metaclass. Induced dependencies are, in our experience, not as frequent as the other kinds of induced relationships. In particular, no need for induced dependencies has been encountered in the quality metamodel that we are mentioning through this paper (although we needed them in other metamodeling experiences). However, the metamodeller could have eventually been interested in capturing the following situation: all the instances of a specific metaclass (e.g., *Metric*) should behave according to a specific M1 interface in every single model that is an instance of the quality metamodel, e.g., they should offer the operation *assessMetric(art:Artifact)*. This interface would be shared by all the models instance of the quality metamodel. This requirement can be modelled as shown in Fig. 8. In fact, *ImplementsMetric_M1* is a heir of *InterfaceRealization* which is an indirect heir of *Dependency*. We haven't depicted the entire path of the generalization hierarchy to avoid messing the figure up. Instances of *Realization* connect classes with *Interface* as the realizations depicted in the M1 level of Fig. 8 do.

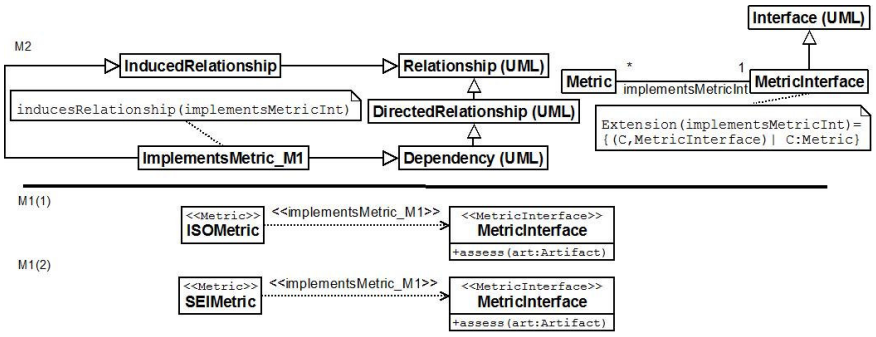

Fig. 8. Dependency-inducing metaassociation.

5. Induced relationships and inheritance

We claimed above that a new M1-relationship would be induced for each pair of classes in the extension of each inducing metaassociation. However, when some of the classes in such extension are connected by inheritance relationships at M1, this may result in redefinitions in the induced relationships. This section deals with this issue.

5.1 Induced associations with inheritance

When some of the classes in the extension of an association-inducing metaassociation are connected by inheritance relationships, some of the induced M1-associations can be considered as *redefinitions* of other more general induced M1-associations.

UML does not consider the notion of *redefinition* applied to associations (i.e., they are not *RedefinableElements*). Next, we define a notion of association redefinition which is appropriate for the purposes of this article. Let *C1, C2, S1* and *S2* be classes such that *S1* conforms to *C1* and *S2* to *C2* (i.e., *S1* is *C1* or one of its descendants, and the same for *S2*). We say that an association *R* between classes *S1* and *S2* is a redefinition of another association *A* between *C1* and *C2* if:

a) *R* is derived from *A* by specialization [13] with the specialization condition: given a pair (*c1, c2*) of *A*'s extension, *c1* is an instance of *S1* and *c2* is an instance of *S2*.

b) Each association-end of *R* redefines its respective association end of *A*.

Intuitively, this idea corresponds to the fact that the association *R* is the same as *A* for the particular case in which instances of *S1* and *S2* are involved.

Notice in the definition above that neither b) implies a) nor the other way around. In particular, *R* could be derived from *A* by the specialization stated in a) but the extension of *A* could include a pair (*c1, c2*) where *c1* is an instance of *S1* and *c2* is not an instance of *S2* (thus, the *S2* end of *R* would not be a redefinition of the *C2* end of *A*). On the other hand, it could happen that each association end of *R* was a redefinition of the corresponding association end of *A* but certain attributes of *A* were

not shared by *R*. For instance, *A* could have the metafeature *UML::Property::isReadOnly* corresponding to one of its ends defined as true, while *R* did not. In this case, *A* would not be a generalization of *R*.

As example, Fig. 9 presents a fragment of the ISO-9126 quality model, expressed as an instance of the quality metamodel (see Fig. 2 (a)). According to ISO-9126, this fragment splits the quality factors (the concept captured by the *Attribute* metaclass) into three categories: characteristics (*ISOCharacteristics*), subcharacteristics (*ISOSubcharacteristics*) and attributes (*ISOAttribute*). On the other hand, the metamodel states that attributes can be direct (*DirectAttribute* metaclass, when they can be measured by observation) and indirect (*IndirectAttribute* metaclass, whose measure depends on that of other attributes). In the ISO framework, characteristics and subcharacteristics are indirect while attributes (*ISOAttributes*) may be of both kinds. Finally, we decide to classify our metrics into observation metrics (*ObservISOMetric*) and calculated metrics (*CalculatedISOMetric*). The following extension of the *measures* metaassociation makes the appropriate assignment of metrics to quality factors and induces M1-associations (as depicted in Fig. 9):

```
Ext(measures) = {(ISOQualityFactor, ISOMetric),
                 (ISOCharacteristic, CalculatedISOMetric),
                 (ISOSubcharacteristic, CalculatedISOMetric),
                 (ISOAttribute, ISOMetric),
                 (IndirectISOAttribute, CalculatedISOMetric),
                 (DirectISOAttribute, ObservISOMetric)}
```

Some of the associations of the above figure may be seen as *redefinitions* of others. For example, the association between *DirectISOAttribute* and *ObservedISOMetric* (named *measuresDirAttr* in Fig. 9) is a redefinition of the association between *ISOAttribute* and *ISOMetric* (named *measuresAttr* in the figure), which, in turn, is a redefinition of the association between *ISOQualityFactor* and *ISOMetric*(*measuresQF*). The meaning of this redefinition is the following: when an instance of the class *DirectISOAttribute* is linked to some instance (say, *m*) of the

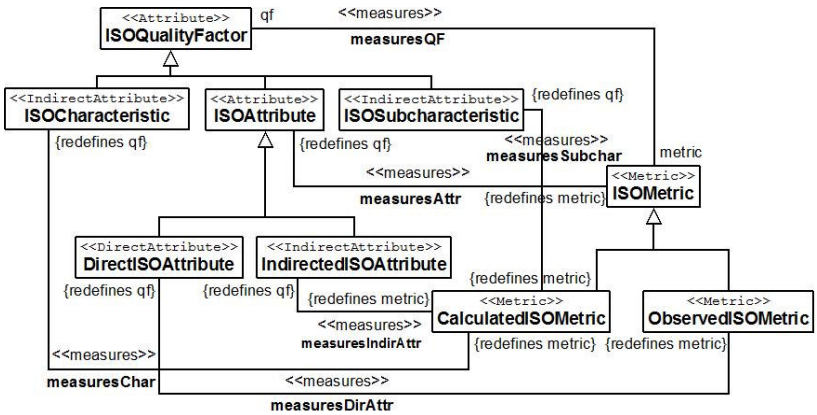

Fig. 9. An instantiation of the metamodel with redefined associations

class *ISOMetric*, *m* will be, actually, an *ObservedISOMetric* (and vice versa). In other words, the association *measuresDirAttr* is the same as the associations *measuresAttr* and *measuresQF* for the particular case in which instances of *DirectISOAttribute* or *ObservedISOMetric* are involved. Incidentally, notice that this forbids the existence of a link in the extension of the association *measuresAttr* between a *DirectISOAttribute* and a *CalculatedISOMetric*, among other similar cases.

In this way, all the associations in the above figure can be considered as redefinitions and just one of them is a non-redefining one: *measuresQF*. We think that this vision is the closest to the specifier's intention and, hence, we have adopted it. This kind of situation is repeated in many other modelling examples (e.g. [9]).

As a final remark, note that this redefinition notion applies whenever generalization relationships exist between classes connected by induced associations, not depending on the source of the generalizations, which may be induced by metaassociations (like those between quality factors in the example) or additionally stated by the modeller (like those between metrics in the example). In this last case, care should be taken not to introduce generalization relationships that are not compatible with induced ones leading to an incorrect instantiation of the metamodel. This would be the case if we pretend to state that an instance of *DirectAttribute* is a generalization of an instance of *Attribute*. This is left as future work (see Sect. 6).

5.2 Induced association classes with inheritance

In a similar way as happened with induced associations, we may have an induced association class (*R*) whose ends are subclasses of the ends of some other induced association class (*A*). In this case, as before, *R* will be a redefinition of *A*.

In order to show an example, Fig. 10 presents an instantiation of a fragment of the quality metamodel (see Fig. 6, M2 level), together with the association classes that would be induced in the case that the extension of *usedIn* was the following:

$$Ext(usedIn) = \{(GartnerClassification,\ UniversityEnvironment),\ (OfficeSuiteApp,\ AcademicEnvironment)\}$$

In the model we instantiate *Domain* with *GartnerClassification* because we want to structure the software domains according to this catalogue of domains. We also define a specialization, *OfficeSuiteApp*, to handle more specific software domains. In a similar way, we define two instances of *Environment.* Two quality models come up as induced association classes: *ISOQMGartnerUniv* (those quality models that result from applying the ISO-9126 to domains of the Gartner classification in University environments) and *ISOQMOfficeAcademic*, which applies the ISO-9126 principles to a specific subclass of domains (office suites) and to specific subclass of *university environment* (*academic environment*, as opposed to *administrative environment*, which is also a part of university environment but has different requirements).

The pair (*GartnerClassification, UniversityEnvironment*) induces the association class called *ISOQMGartnerUniv*, which is an instance of <<QualityModel>>. On the other hand, the pair (*OfficeSuiteApp*, *AcademicEnvironment*) induces the association class named *ISOQMOfficeAcademic*, also instance of <<QualityModel>>.

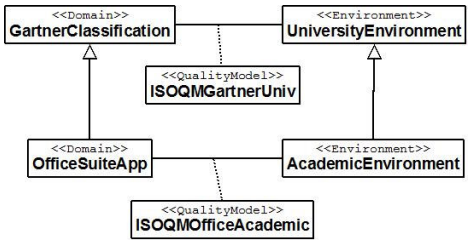

Fig. 10. An instantiation of the quality metamodel with redefined association-classes

The association class *ISOQMOfficeAcademic* may be seen as a redefinition of *ISOQMGartnerUniv*. The meaning of this redefinition is the following: when an instance of the class *OfficeSuiteApp* is linked to some instance (say, *uenv*) of the class *UniversityEnvironment*, *uenv* will be, actually, an *AcademicEnvironment* (and vice versa). In other words, the association class *ISOQMOfficeAcademic* is the same as the association class *ISOQMGartnerUniv* for the particular case in which instances of *OfficeSuiteApp* and *AcademicEnv* are involved.

In order to make the formalization of this idea easier, we introduce, in next section, the notion of directed graph associated to the extension of a metaassociation.

5.3 Formal definition

Let *M2A* be an association/association-class inducing metaassociation between the metaclasses *MC1* and *MC2*. Let *MA* be a heir of *Association*. Hence, (*M2A,MA*) is the pair that induces associations/association-classes at layer M1. Let *Ext(M2A)* be an extension of *M2A*. This extension is constituted by a set of pairs (*C1, C2*), where *C1* is an instance of *MC1* and *C2* is an instance of *MC2*. According to the declaration of *M2A* as inducing, each pair in *Ext(M2A)* will have an (induced) distinct instance of *MA* (i.e., an association/association class) connecting them.

We define the *directed graph associated to Ext(M2A,MA)* in a particular metamodel instantiation (denoted as *DGExt(M2A,MA)*) as a directed graph such that:

- The set of vertices of *DGExt(M2A,MA)* is the set of instances of *MA* that connect the pairs in *Ext(M2A,MA)*.
- Given two distinct vertices *A* and *A'* of *DGExt(M2A,MA)*, being *A*, *A'* associations or association-classes between *C1* and *C2*, and *C1'* and *C2'* respectively: there is an edge from *A* to *A'* iff *C1* conforms to *C1'* and *C2* conforms to *C2'*.

The idea conveyed by this graph is that an induced association/association class *A* between *C1* and *C2* is a redefinition (in the sense of the previous sections) of another induced association/association class *A'* between *C1'* and *C2'* if and only if there is a path from *A* to *A'*. For example, the directed graph *DGExt(measures, measures_M1)* corresponding to the example shown in Fig. 10 is shown in Fig. 11. Notice that the construction of this directed graph is straightforward from its definition.

Two issues can be easily drawn from the definition of *DGExt*:

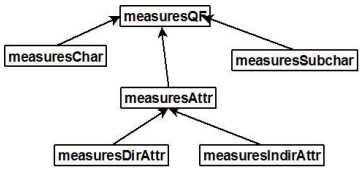

Fig. 11. *DGExt(measures,measures_M1)* corresponding to the model of Fig. 9

1) *DGExt(M2A,MA)* is acyclic as long as generalization hierarchies are also acyclic.

2) The relation $(V, <)$ is a partial order, where:

- V is the set of vertices of *DGExt(M2A, MA)* and
- For any A, A' of V: $A < A'$ iff *DGExt(M2A, MAC)* contains a path from A to A'.

The notion of directed graph associated to a metaassociation extension, together with the partial order that can be drawn from such definition, allows an easy formalisation of the idea presented in the previous sections concerning which induced M1-associations/association classes are, actually, redefinitions of which other. The idea is the following: given a metaassociation *M2A* and a metaclass *MA* (heir of *Association*) conceived as a pair that induces associations at level M1 and given also a specific model *mod*, an M1-association/association-class is induced in *mod* for each tuple in the *Ext(M2A,MA)*. In this context, the non-redefining M1-association/association classes are those corresponding to the maximal elements of $(V, <)$, that is, the vertices of *DGExt(M2A,MA)* which have no successor (i.e., from which no edge is issued). In the case of the example, there is one of such vertices: *measuresQF*.

On the other hand, the redefining associations correspond to those vertices which are not maximal. In particular, a specific association/association-class represented by a vertex v of *DGExt(M2A,MAC)* redefines the associations/association classes that correspond to the vertices which are successors of v. For example, the association *measuresDirAttr* is a redefinition of *measuresQF*.

5.4 Inheritance in other relationships

Consider the situation in which an inducing metaassociation *MA* leads to an induced association *a1* between classes *A* and *B* and to another induced association *sa* between classes *SA* and *SB*, which are subclasses of *A* and *B* respectively. In such case, as it has been discussed above, the issue of redefined associations comes up in a natural way since: (a) both *a* and *sa* are induced from the same metaassociation, and (b) the extension of *sa* is constituted by pairs of instances of *SA* and *SB* and the extension of *a*, by pairs of instances of *A* and *B*. However, by definition of generalization, instances of *SA* (*SB*) are also instances of *A* (*B*). Hence, it makes sense that the pairs linked by *sa* are, in fact, a subset of those linked by *a* and, hence, the association *sa* can be seen as a redefinition of *a*. In the case of induced generalizations or dependencies, item (b) does not occur. Therefore, the notion of redefinition of either induced generalizations or dependencies has a less natural sense.

6. Conclusions and future work

We have presented a general notion of inducing metaassociation to be used in UML heavyweight extensions that extends our previous work in [9] by inducing not just associations but all type of UML relationships at M1. As a necessary complement to our work, we have also extended our analysis on how the presence of generalizations in M1 models (either induced or directly defined) affects the induced elements.

We believe that our proposal supplies more expressiveness and accuracy in the definition of a heavyweight extension of the UML metamodel while keeping MOF-compliance and providing strict metamodelling. Remarkably, the ability to declare metaelements as inducing is a powerful conceptual tool for metamodelers since the intended semantics of the UML extension can be more accurately defined. The effort required is just to declare a new metaclass (heir of *Relationship* and the new *InducedRelationship* metaclass) for every inducing metaassociation, but even this declaration may be considered positive from the comprehensibility point of view, since the inducing nature of metaassociations is made explicit in the metamodel.

The induced relationships that have been presented in this article are binary relationships. In general, this is not a limitation since the vast majority of relationships that come up naturally in a model are (or can be decomposed into) binary relationships. However, the convenience of n-ary induced relationships cannot be excluded in the future. For this reason, it could be interesting to define inducing relationships for the case of n-ary relationships (specially, n-ary associations), which should be considered carefully due to the absence of ternary metaassociations.

As more future work, we are considering to complement our inducing mechanism providing the transformation of the heavyweight UML extension generated by our approach into an UML profile, following the ideas presented in [14]. To make our inducing mechanism even more useful and easy to apply we are also working on an accurate definition of correct instantiation of a metamodel. This definition should state which conditions must hold to guarantee the soundness of the models obtained as instantiations of a metamodel taking also into account the induced elements.

The problem faced in this article has also drawn the attention of other researchers, who have identified it as an important challenge [15, 16, 17]. We analyzed these approaches in [8] and found out that none of them was compliant with the MOF 2.0 architecture (thus being non-standard approaches) and some of them suffer from other drawbacks. A newest approach, [18], is based on the same principles than those cited above and, hence, suffers from the same difficulties.

It is worth mentioning that, if we look at metamodeling environments other than MOF/UML, we find that some of them induce relationships in a natural way because the instantiation of a relationship leads to another relationship in the next level, in the same way as the instantiation of an entity leads to another entity (metaclass to class in the MOF framework). This is the case of Telos [19], which defines individuals (to represent entities) and attributes (binary relationships between individuals) and a classification dimension (instantiation hierarchy) for both elements. Another metamodeling example with symmetrical treatment of entities and relationships is the MetaEdit+ Workbench tool [20]; it allows the user defining a relationship in a level

and instantiating it in a lower level to obtain relationships in this last level. Induction of relationships proves to be a convenient option in those frameworks lacking this symmetry, as happens in the MOF-related metamodels. For instance, in [21] and other works around OWL [22], an ongoing research effort is adding metamodeling expressiveness taking into account the computational problems that may arise. Last but not least, we would like to remark that our proposal allows inducing not just associations but also association classes, generalizations and dependencies. These cases are not covered by the other approaches because the instantiation of the concept equivalent to that of association in UML cannot generate anything else than another association. Our future work includes a more detailed assessment of these facts.

References

1. *UML 2.0 Infrastructure*. OMG doc. formal/07-05-05. Available at http://www.omg.org/.
2. *UML 2.0 Superstructure*. OMG doc. formal/07-05-04. Available at http://www.omg.org/.
3. *Common Warehouse Metamodel Specification*. OMG doc. formal/2003-03-02. Available at www.omg.org
4. *Software Process Engineering Metamodel Specification (SPEM)*. OMG doc. formal/2005-01-06. Available at www.omg.org.
5. *UML profile for CORBA*. OMG doc. formal/02-04-01. Available at www.omg.org
6. *UML 2.0 testing profile*. OMG doc. formal/05-07-07. Available at www.omg.org.
7. Knapp, A., Koch, N., Moser F., Zhang, G. "ArgoUWE: A CASE Tool for Web Applications". In *Procs. EMSISE'03*, 2003.
8. *MOF 2.0 Core Final Adopted Specification*. OMG doc. formal/06-01-01. Available at http://www.omg.org/spec/MOF/2.0/.
9. Burgués, X., Franch, X., Ribó, J.M. "Improving the Accuracy of UML Metamodel Extensions by Introducing Induced Associations". *SoSyM* 7(1), Springer-Verlag, Feb. 2008.
10. Burgués, X., Franch, X., Ribó, J.M. "A MOF-Compliant Approach to Software Quality Modeling". In *Procs. ER'05*, LNCS 3716, Springer-Verlag, 2005.
11. *ISO/IEC Standard 9126-1. Software Engineering – Product Quality – Part 1*, 2001.
12. Kruchten, P. *The Rational Unified Process. An Introduction*. Addison-Wesley, 2000.
13. Olivé, A. *Conceptual Modeling of Information Systems*. Springer-Verlag, 2007.
14. Ribó, J.M. *PROMENADE: A UML-based Approach to Software Process Modelling*. PhD. Thesis, UPC, 2002.
15. Atkinson, C., Kühne, T. "Rearchitecting the UML Infrastructure". *ACM TOMACS*, 12(4), Oct. 2002.
16. Álvarez, J., Evans, A., Sammut, P. "MML and the Metamodel Architecture". WTUML 2001.
17. Henderson-Sellers, B., Gonzalez-Perez, C. "The Rationale of Powertype-based Metamodelling to Underpin Software Development Methodologies". In *Procs. APCCM'05*, 2005.
18. Gutheil, M., Kennel, B., Atkinson, C. "A Systematic Approach to Connectors in a Multi-level Environment". In *Procs. MoDELS'08*, LNCS 5301, Springer-Verlag, 2008.
19. Mylopoulos, J., Borgida, A., Jarke, M., Koubarakis, M. "Telos: Representing Knowledge About Information Systems". *ACM TOIS* 8(4), Oct. 1990.
20. The MetaEdit tool. Available at http://www.metacase.com.
21. Motik, B. "On the Properties of Metamodeling in OWL". *JOLC* 17(4), Oxford Univ. Press, Aug. 2007.
22. OWL web page. Available at http://www.w3.org/2007/OWL/wiki/OWL_Working_Group.