A Radical Revision of UML's Role Concept*

Friedrich Steimann

Institut für Technische Informatik
Rechnergestützte Wissensverarbeitung
Universität Hannover, Appelstraße 4, D-30167 Hannover
steimann@acm.org

Abstract. UML's current definition of the role concept comes with many problems, not the least being that it is difficult to understand and communicate. This paper proposes a revised UML metamodel building on a much simpler role definition. Moreover, it replaces the rather unusual notions of association role and association end role as well as the rarely used association generalization with the more popular concept of overloading, thereby leading to a considerable reduction in the number of modelling concepts. Despite the rather radical nature of the proposed alterations, no changes in UML notation become necessary. However, a notable change in modelling style including in particular a clearer separation of structure and interaction diagrams are among the likely effects of the proposed revision.

1 Introduction

UML's current version [OMG 1999] has three different definitions of the role concept:

- roles as names of association ends,
- roles as slots in collaborations, and
- roles as dynamic classes.

The first two are heritage from the entity-relationship diagram and object-oriented methods such as OORAM [Reenskaug et al. 1996], respectively, and, as will be seen, are mostly independent of each other. The third is only mentioned in passing [§ 3.27.1, p. 3-46] and is presumably a tribute to a view of roles commonly found in the literature [Steimann 1999].

Of course there are various other uses of the term spread over the UML specification, for instance the roles of an actor in different use case diagrams or the roles in a state chart, but these uses are either instances of one of the three definitions listed above or merely a way of speaking. After all, role is one of the most elementary terms not only in modelling (cf. sect. 5), and it is difficult, if not impossible, to get along without it.

The rationale of the first definition of the role concept is simple: to uniquely identify it, every end of an association (like every place of a relationship in the entity-relationship diagram) can be assigned a rolename, and this end is then referred to as a role [§ 3.42.2]. The second definition, the definition of roles as slots of a collaboration, involves more:

* in: E Evans, S Kent, B Selic *UML 2000: Proceedings of the 3rd International Conference* (Springer, 2000) 194–209.

> "*... while a classifier is a complete description of instances, a classifier role is a description of the features required in a particular collaboration, i.e. a classifier role is a projection of, or a view of, a classifier. The classifier so represented is referred to as the base classifier of that particular classifier role.*" [§ 2.10.4, p. 2-112]

and

> "*Each association role represents the usage of an association in the collaboration, and it is defined between the classifier roles that represent the associated classifiers. The represented association is called the base association of the association role. As the association roles specify a particular usage of an association in a specific collaboration, all constraints expressed by the association ends are not necessarily required to be fulfilled in the specified usage. The multiplicity of the association end may be reduced in the collaboration, i.e. the upper and the lower bounds of the association end roles may be lower than those of the corresponding base association end, as it might be that only a subset of the associated instances participate in the collaboration instance.*" [ibid.]

These definitions become more transparent when looking at the corresponding excerpt of UML's metamodel, as compiled from figures 2-5, 2-6, and 2-17 of the original specification (shown in figure 1). For the sake of conciseness, classifiers other than *Class* and *Interface* have been omitted; the complete list can be found in [Rumbaugh et al. 1999]. It must be noted, however, that much of UML's complexity is due to this generalization and the resultant genericity, and that the correctness of the following and all other argumentation critically depends on what is comprised under the classifier term. Also notice that *AssociationClass*, a common subclass of *Association* and *Class*, has been omitted; although a handy modelling concept, it entails certain consistency problems that are not dealt with here.

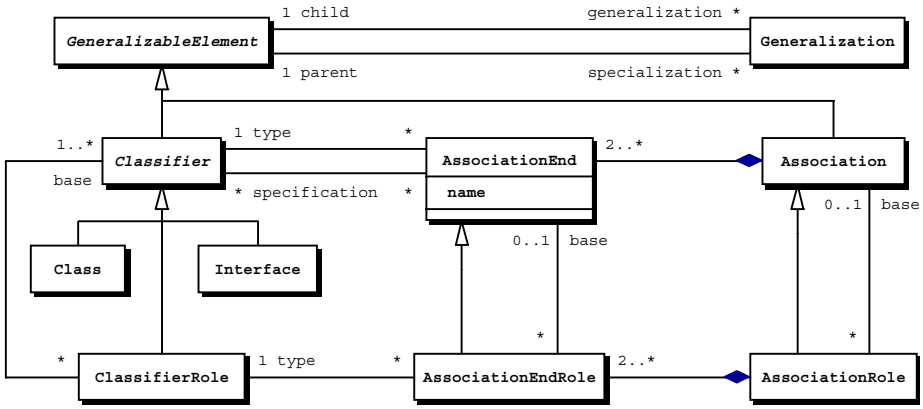

Figure 1: An excerpt of the abstract syntax that forms the structural part of the UML metamodel. More than one base for one classifier role may only be specified if classification is multiple.

Closer inspection of the original specification reveals that there are many problems in association with roles. Some of them are obvious and easily avoided. For instance, a classifier role must not specify a classifier role as its base, and association ends and association end roles must not be mixed in one association.[1] Other things are subtler and become apparent only when working through the textual specification. Some of these will be discussed next.

2 Problems

2.1 Base vs. Projection

The use of the term base in connection with collaboration roles (as reflected in the metamodel of figure 1) suggests that the roles of a collaboration are derived from their bases. And indeed, one well-formedness rule for association end roles states that the base classifier of the classifier role connected to the association end role must be the same as or a specialization[2] of the classifier connected to the association end role's base [§ 2.10.3, p. 2-108]:

```
self.type.base = self.base.type
or
self.type.base.allParents->includes (self.base.type)
```

But this observation is superficial. The rule only requires that the base itself, if not the same classifier, is a specialization, not that the classifier role is a specialization of its base. Quite the contrary: a role repeats only parts of the features of its base (the *availableFeature*s [§2.10.2, p. 2-106]) and in case of the classifier role, lets instances of any classifier engage in the association as long as it conforms to whatever is required by this role. Therefore, the extent of the role is bigger than that of its base,[3] and use of the term base (reminiscent of base type/derived type) is misleading.[4]

But if the classifier role is more general than its base, why is this base allowed to be a specialization of the original type? Why should one first restrict the extent of a classifier role (by selecting a specialization as its base) and then widen it (by reducing the number of available features)? And doesn't widening the extent conflict with the view that "only a subset of the associated instances participate in the collaboration instance"? Admittedly, it does make sense to let instances of arbitrary type participate in a collaboration as long as they do what is required by the role they play, but then this liberty should not be limited to collaborations, since a general association specification may be equally permissive.[5]

---

[1] Problems of this kind can easily be (and some actually are) fixed through well-formedness rules.

[2] Throughout this text, specialization is defined as the inverse of generalization.

[3] Note that the semantics of the base relation differs depending on what is being related, because the same does not hold for association and association end roles: their extents are samller.

[4] The roles are actually also referred to as *projections* or *views* of their bases (although *partial specifications* would seem more adequate).

[5] In fact, several extensions of the entity-relationship model allow ad hoc disjunctions (which are not generalizations!) of entity types to occur in the places of relationship types [Steimann 1999]. Interfaces may serve as such ad hoc disjunctions [Steimann 2001], but this is not discussed in the UML specification.

2.2 Interface Specifiers vs. Classifier Roles

In UML, each association end can specify one or more classifiers as *interface specifiers* (mapped to the *specification* pseudo-attribute, see figure 1) which are to restrict the access to the instances of the classifier across the association. In a way, these interface specifiers parallel the specification of base classifiers with a classifier role, but despite the symmetric structure of the metamodel this parallelism is unapparent. Instead, the UML specification offers various nebulous explanations, for instance

> "*A role may be static (e.g., an association end) or dynamic (e.g., a collaboration role)*"

in the glossary or

> "*The use of a rolename and interface specifier are equivalent to creating a small collaboration that includes just an association and two roles, whose structure is defined by the rolename and attached classifier on the original association. Therefore, the original association and classifiers are a use of the collaboration. The original classifier must be compatible with the interface specifier (which can be an interface or a type, among other kinds of classifiers)*"

in the notation guide [§ 3.42.2, p. 3-66]. Undoubtedly, both interface specifiers of an association end and base classifiers of a collaboration role serve to specify what is required from the instances engaging in an association (or collaboration). However, *AssociationEnd* generalizes *AssociationEndRole*, not *ClassifierRole*, and it lets the former inherit the rolename attribute. But what happened to the *specification* pseudo-attribute that should also be inherited from *AssociationEnd*? Has it been dropped? If an association end role in a collaboration specifies an association end as its base and this base lists interface specifiers, must the classifier role of such an association end role conform to these interface specifiers? And while the interface specifiers only serve to restrict access, not to constrain the type of the instances (this is the task of the connected type classifier), the classifier role does both: it provides a partial specification of the classifiers whose instances can play the role, and it restricts access to the features of this partial specification. Last but not least, while the base classifier of a classifier role must connect to the base association of its association role, no such constraint is imposed on the interface specifiers. Thus, the alleged symmetry is really an asymmetry (and a very confusing one at that).

2.3 Static Structure Diagrams vs. Collaboration Diagrams

The duality of concepts just criticized is paralleled by a second one: that of static structure and collaborations. For an example of this, look at the diagram of figure 2a. According to the UML specification it is a perfect collaboration diagram, albeit one without interaction information. But where is the collaboration? Is the structural information given in figure 2a less generally relevant than that of figure 2b, which is a class diagram? Or is the existence of (classifier) roles the true (and only) reason to declare one as a collaboration diagram and not the other? Indeed, one might prefer to model *Teacher* and *Student* as subclasses (rather than roles) of *Person*[6] — would this

[6] admittedly a poor preference: roles are no subtypes [Firesmith & Henderson-Sellers 1998; Steimann 1999]

simple modification turn the collaboration diagram into an ordinary class diagram? Hardly.

Note well, the problem here is not that collaboration diagrams express structure, but that roles (other than the rolenames of association ends) are excluded from class diagrams. Many kinds of constraints on associations (including multiplicities, exclusive ors etc.) have been devised, but it is impossible to express in a class diagram that of all persons only teachers can be members of a staff and give courses, even though this may very well be a fundamental structural property of a modelling problem. In such cases, the introduction of an extra collaboration diagram to express structure (but lacking an actual interaction) must appear artificial, as is the case for figure 2.

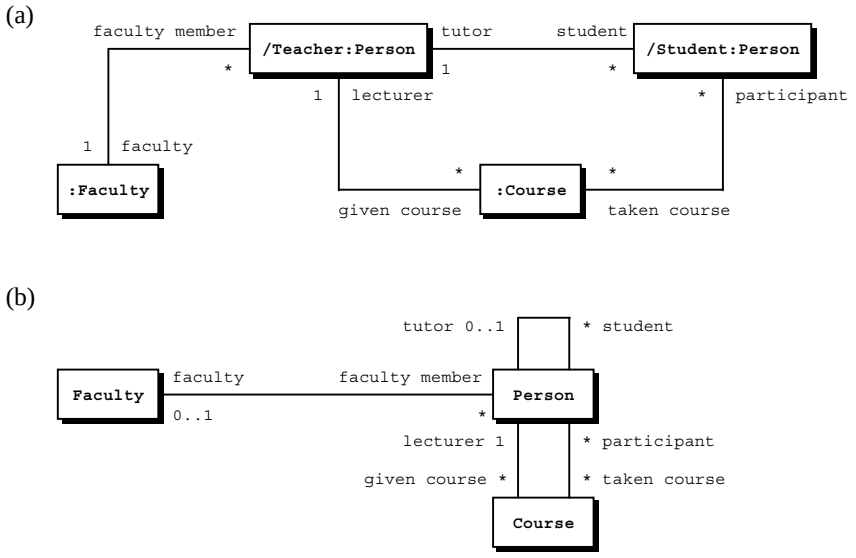

Figure 2: (a) Collaboration diagram without interaction information (from [OMG 1999])
(b) corresponding class diagram (adapted from [Övergaard 1999])

2.4 Specification vs. Instance Level

In UML, interaction diagrams are offered at two levels: the specification (or classifier) and the instance level [§ 2.10.4, p. 2-112]. However, interactions (and collaborations) always take place between actual objects, not their roles or classes, and it seems that the specification level is needed only insofar as interactions require or impose additional structure, structure that, because of its generality, should indeed be expressed on the classifier level. The specification of interaction itself however must be able to distinguish different objects even if they play the same role and, more importantly, must be able to express that the same object plays different roles in one interaction. Such is not possible on the classifier level, and it would appear that the instance level is the natural domain for interaction specification. The definition of

collaborations with generic interaction patterns on the classifier level tends to blur this distinction, and is rather difficult to communicate.

2.5 Association Roles vs. Association Generalization

As quoted above, association roles "specify a particular usage of an association in a specific collaboration" in which the "constraints expressed by the association ends are not necessarily required to be fulfilled in the specified usage" [OMG 1999, § 2.10.4, p. 2-112]. For instance:

> "*The multiplicity of the association end may be reduced in the collaboration, i.e. the upper and the lower bounds of the association end roles may be lower than those of the corresponding base association end, as it might be that only a subset of the associated instances participate in the collaboration instance.*"

In other words: the extent of an association role is a subset of the extent of its base association. And the UML specification continues:

> "*Similarly, an association may be traversed in some, but perhaps not all, of the allowed directions in the specific collaboration, i.e. the isNavigable property of an association end role may be false even if that property of the base association end is true.* [...] *The changeability and ordering of an association end may be strengthened in an association-end role, i.e. in a particular usage the end is used in a more restricted way than is defined by the association.*" [§ 2.10.4, p. 2-112]

But association roles are not the only way to restrict the use of an association in a UML model: as indicated by the metamodel in figure 1, associations can also be generalized. Whereas the current OMG specification is very brief about association generalization ("Generalization may be applied to associations as well as classes, although the notation may be messy because of the multiple lines. An association can be shown as an association class for the purpose of attaching generalization arrows." [OMG 1999, § 3.49.2, p. 3-80] is the only textual mention I found), other sources go into more detail:

> "*As with any generalization relationship, the child element* [of an association generalization] *must add to the intent (defining rules) of the parent and must subset the extent (set of instances) of the parent. Adding to the intent means adding additional constraints. A child association is more constrained than its parent.*" [Rumbaugh et al. 1999, p. 163].

But isn't disallowing navigation more constrained than leaving navigability open (that is, allowing it), and isn't being sorted more constrained than not being sorted (it is implied by an additional condition, that of the elements being ordered)? All in all, it seems that the semantics of association roles is largely covered by the specialization of their base associations, but this duality of concepts (yet another one!) is never mentioned.

2.6 Interfaces vs. Classes

For some unapparent reason, UML doesn't make much use of interfaces. In fact, even though there appears to be a basic awareness of the general importance of inter-

faces, the metamodel has no dedicated use for it (as it has, e.g., for collaboration roles). Instead, the more general *Classifier* (comprising classes and interfaces) is consistently (ab)used to specify interfaces where the use of the *Interface* concept were in place. For instance, the interface specifiers of an association end may be classes, and classifier roles are general classifiers although their purpose is clearly that of a partial specification and hence that of an interface[7].

3 Revision

It should be clear from the above that UML's current role concept is problematic. The alleged symmetry of association end and collaboration role is only seeming, and particularly the introduction of association roles and association end roles which share only superficial properties with classifier roles (such as having bases and being slots in a collaboration) appears to be a peculiarity of UML that is difficult to transport. As a matter of fact, the description of the connection between associations and association roles remains rather sketchy, and it may be speculated that any attempt to pin it down precisely would lead to more problems.

Fortunately, there is a simple solution to all this. Roles are no specialty of collaborations and the interactions they enable, they are a core structural element (although not necessarily a static one). Roles are bridges between instantiable classifiers and their associations, be it in the scope of a collaboration or outside the context of any interaction. But roles are no association ends (or the names thereof) — a role (more precisely, a *role type*) is a classifier like a class, only that it does not have instances of its own. Instead, a role recruits its instances from those classifiers that are declared compatible with the role, that is, that comply with the partial specification that makes the role.

As for the remaining UML roles, namely the association roles and association end roles: they too are not limited to the scope of collaborations. But unlike classifier roles, they are rivaled by an equally suitable (an much better understood) concept: *association overloading*. Overloading, expressing that the same association (operation, relation, or function) has different properties depending on the classifiers (types) being associated, is ubiquitous in object-oriented programming as well as in the theory of type structures [Bläsius et al. 1989], and it gracefully accounts for the fact that an association in a collaboration (with only a subset of the instances involved) may have different properties than the same association outside the scope of the collaboration. At the same time, overloading accounts for most of the semantics of association specialization as a purely structural modelling element — the only obvious difference is that a specialized association can have a different name than its generalization.

Note well, the suggested revision is not a plea against collaborations. Collaborations remain as an important means of zooming into particular structural aspects of a model and as a basis for expressing the interactions that build upon them. The point here is that the structural aspects of a collaboration can be expressed with the same modelling concepts as any classifier structure, be it in or outside the scope of collaboration.

---

[7] perhaps not an interface in the definition of UML, but that is another issue

More specifically, the revision of UML's role concept is comprised by the following four commitments.

1. The metaclasses *Interface* and *ClassifierRole* are merged into a new metaclass *Role*. The restrictions regarding interfaces in UML (that they cannot have attributes or occur in other places than the target ends of directed associations) are dropped. *Class* and *Role* are strictly separated: while classes can be instantiated (unless of course they are abstract), roles cannot.
2. The association between classifier roles and their base classifiers is replaced by a new relationship, named *populates*[8], which relates classes with the roles their instances can play. (It is convenient to speak of a class as *populating* a role and of an instance as *playing* a role. It is important that these are distinguished: populating roughly corresponds to the subclass relationship among classes, and playing to the instance-of relationship of an instance to its class. The diction in this regard is often ambiguous in the literature.)
3. Association ends are required to connect to roles exclusively. Because roles are interfaces and subroles can combine several interfaces, both pseudo-attributes *type* and *specification* are replaced by one new relationship *specifies* associating each association end with one role. The classes whose instances actually participate in an association are specified only indirectly via the *populates* relationship between classes and roles. Association ends need not be given a rolename; if they are, this name must equal the connected role's. Every role must be unique within an association, i.e., no two association ends of one association must specify the same role.
4. The metaclasses *AssociationEndRole*, *AssociationRole*, and the generalization of associations are replaced by *association overloading*. For this purpose, a new metaclass, *Signature*, is introduced whose instances stand between an association and its (overloaded) association ends. Thus, rather than giving rise to an association role, an association restricted in the context of a collaboration specifies a new instance of *Signature*, comprising new association ends, each connected to a role defined by the collaboration.

The metamodel the suggested changes result in is presented in figure 3. First and foremost, it does not distinguish between core and collaboration modelling elements: association ends are always connected to roles, which are always populated by classes (representative of all other classifiers), although a particular diagram may choose not to show this. Second, UML's indifference with regard to a classifier's being a class or an interface is lifted: interfaces, now termed roles, exclusively occur at associations' ends, and classes exclusively as populating them. Last but not least, it accounts for the fact that the same association can occur multiply in one model, each occurrence disambiguated by a different signature, always involving different association ends and usually also connecting to a different set of roles.

[8] Alternative terms for *populates* would be *implements* or *realizes*, but these are rather technical.

Figure 3: The revised metamodel for UML

A final note on the representation of generalization in figure 3. UML introduces *Generalization* as an instantiable metaclass (see figure 1). In order to avoid inconsistencies, it must be declared for all generalizable elements what is inherited down each generalization relationship (instance of the *Generalization* metaclass). The changed metamodel takes a simpler approach: it represents generalization as an overloaded relationship of the metamodel. Note that *generalizes* is of the same order as *populates*[9], while all instances of *Association* are of a lower order. This way, no precautions avoiding inconsistencies and paradoxes need to be taken.

4 Changes for the Modeller

The good news first: there are no changes of notation necessary. The definition of the new role concept as a merger of collaboration roles and interfaces and the emphasis on the interface aspect suggests that roles are drawn like interfaces, i.e., as stereotyped classifiers or, preferably, as circles. The overloading of associations covering both association generalization and association roles is implicitly expressed by different associations with the same name (and, under certain conditions, may be explicated by generalization arrows). All other changes have no impact on the *notation* itself, only on the *style* diagrams are composed.

4.1 A New Style for Structure Diagrams

The most obvious consequence of the new metamodel on diagram style is that all associations must end at roles. Thus, roles (formerly interfaces) should be seen more frequently in class diagrams, where they function as interface specifiers restricting access to instances across the association, but also as partial specifications of the classes whose instances link. In fact, placing roles, not classes, at association ends means lifting the 'program to an interface' maxim (that is considered good practice in object-oriented programming [Gamma et al. 1995; D'Souza & Wills 1998; Steimann 2001]) to the modelling level; it entails that instances of various classes can interchangeably engage in the same association as long as their classes populate the role specified.

In a collaboration diagram at specification level, all associations already end at roles (formerly, classifier roles). However, the classes that (by default) populate the roles, the former base classes (if provided), are no longer implicit parts of the role symbols, but have to be shown explicitly. Note that roles in a collaboration diagram serve no

[9] UML specifies a similar relationship of types and implementation classes: "[...] types may only specialize other types and implementation classes may only specialize other implementation classes. Types and implementation classes can be related only be realization" [3.27.1, p. 3-46]. But the distinction of types and implementation classes is not that of roles and classes.

different purpose than roles in a (new style) class diagram: they specify the interface (including attributes!) that are required from the collaborators, but they do not commit the collaboration to the classes whose instances can play the role. Figure 4 shows a collaboration diagram and a class diagram in the new style, showing structure, but no behaviour.

Because with the new style class diagrams and the structural aspects of collaboration diagrams rely on the same modelling concepts, there is no syntactical difference between the two diagram types: both show associations connecting roles and classes populating them. Consequently, the difference between the two can be based only on intention: drawing a separate collaboration diagram is useful if and only if (a) interaction information is to be provided or (b) associations that are relevant only within the scope of an actual collaboration (formerly: association roles) are to be shown.

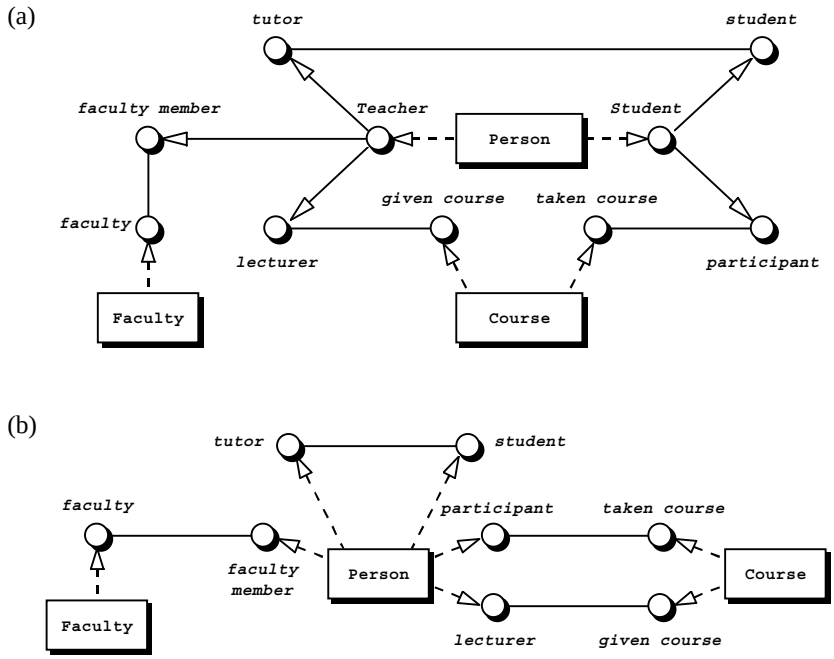

Figure 4: New style collaboration diagram (a) and class diagram (b) corresponding to the diagrams of figure 2. Note that collaboration diagram differs from the class diagram only by the introduction of the subroles *Teacher* and *Student*.

Finally, a collaboration diagram at instance level specifies behaviour by showing prototypical objects as they interact. Although not reserved for the instance level, it is here that UML's lollipop notation unfolds its intuitive expressiveness: in an interaction, objects should be accessed exclusively via the roles they play, and these roles (which are themselves no instances) are connected to their players by unlabeled lines

to express that they are the plugpoints between (substitutable) objects and the specified interaction. Figure 5 gives an example of this.

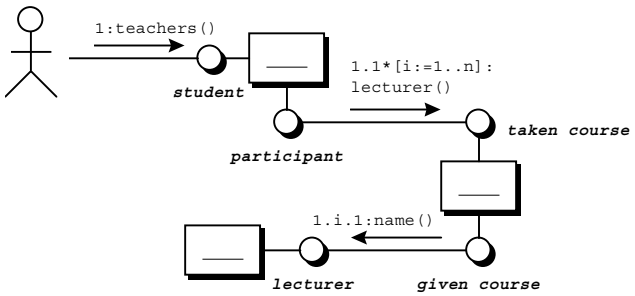

Figure 5: New style collaboration diagram on the instance level. All objects can, but need not, remain anonymous and lack class information. The lollipop notation of an object and its roles is reminiscent of the pieces of a jigsaw puzzle, which is intentional.

Mapping. Mapping of the new style diagrams to instances of the revised metamodel is straightforward. Every class maps to an instance of *Class*, every role to one of *Role*. Realization arrows map to instances of the *populates* association, and generalization arrows to instances of the respective *generalization* associations. Every occurrence of an association with the same name maps to the same instance of *Association*, but to a different instance of *Signature* with correspondingly different instances of *AssociationEnd*. Figure 6 shows the mapping for a simple diagram.

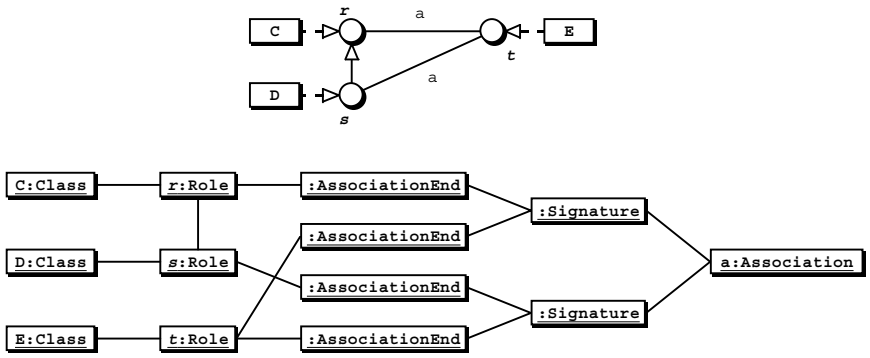

Figure 6: Mapping of a new style structure diagram to an instance of the revised metamodel. Association labels of the object diagram are omitted, but can easily be reconstructed from the class diagram of figure 3.

4.2 Retaining Old Style Diagrams

Every diagram of the new style is a valid UML diagram, mapping to the old as well as the revised metamodel. But there are even better news: all diagrams of the old style that map to instances of the old metamodel also map to the revised one. However, as one may imagine, the mapping is not straightforward, and it is more instructive to look at a systematic transformation of conventional diagrams into diagrams of the new style, for which mapping is one-to-one.

The outline of the mapping procedure is as follows: First, it is ensured that every association end connects to a suitable role. Then, the classifier formerly occupying the association end is related to that role, either as populating, as specializing, or as being identical to it. Finally, in case the association is a specialization or an association role, the necessary overloading is declared. During this procedure it may become necessary to introduce a (default) role for a class that is defined as the class's complete interface. In these cases, the role is given the same name as the class (only that the name is set in italics), and it is understood that the two are different types even though they carry the same names.[10]

Mapping. In class diagrams of the old style classifiers (mostly classes) directly connect to association ends. However, these association ends usually come with rolenames. Even though these rolenames are no roles, but mere labels, they could be, and the first step in mapping is to introduce a role for every association end with the rolename as its name, and to let the classifier formerly connected to the association end populate that role. If no rolename is provided, the role is given the class's name (the default role), and if the classifier is an interface, it is considered a direct subrole of the role introduced by the rolename (if provided), or everything is left as is (if no extra rolename is provided). Figure 7 a shows the mapping for all four cases.

Next come the interface specifiers. The interface specifiers at association ends always come with rolenames, but the association ends may connect to classes or interfaces. The mapping to the new drawing style in either case is as above, only that each interface specifier is made an immediate superrole of the role connected to the association end. If an interface specifier is a class, its default role is assumed. Figure 7 b shows the details. Note that the interface specifiers are independent of the association end, since it is not required that they participate in the same association.

Finally the collaboration roles. Here, we restrict ourselves to single classification, which is still predominating in the object-oriented world. Thus, only one base classifier is listed with each classifier role. If this base classifier is a class the classifier role is mapped to an interface with the base classifier populating that interface (figure 7 c). Whether or not the role is the default role of the class depends on whether the classifier role is a partial specification of the base class, i.e., whether it restricts the available features. By default, we must assume that this is the case so that a new role is introduced carrying the classifier role name if specified or a new name otherwise; if not, the default role can be used. Finally, if the base classifier is an interface, it replaces the new role and the class populating it (figure 7 d). Note in particular that the mapping of interface specifiers and classifier roles have not much in common.

---

[10] The names may be thought of as being qualified by a Class or a Role suffix, respectively, but this suffix is left implicit for the sake of readability.

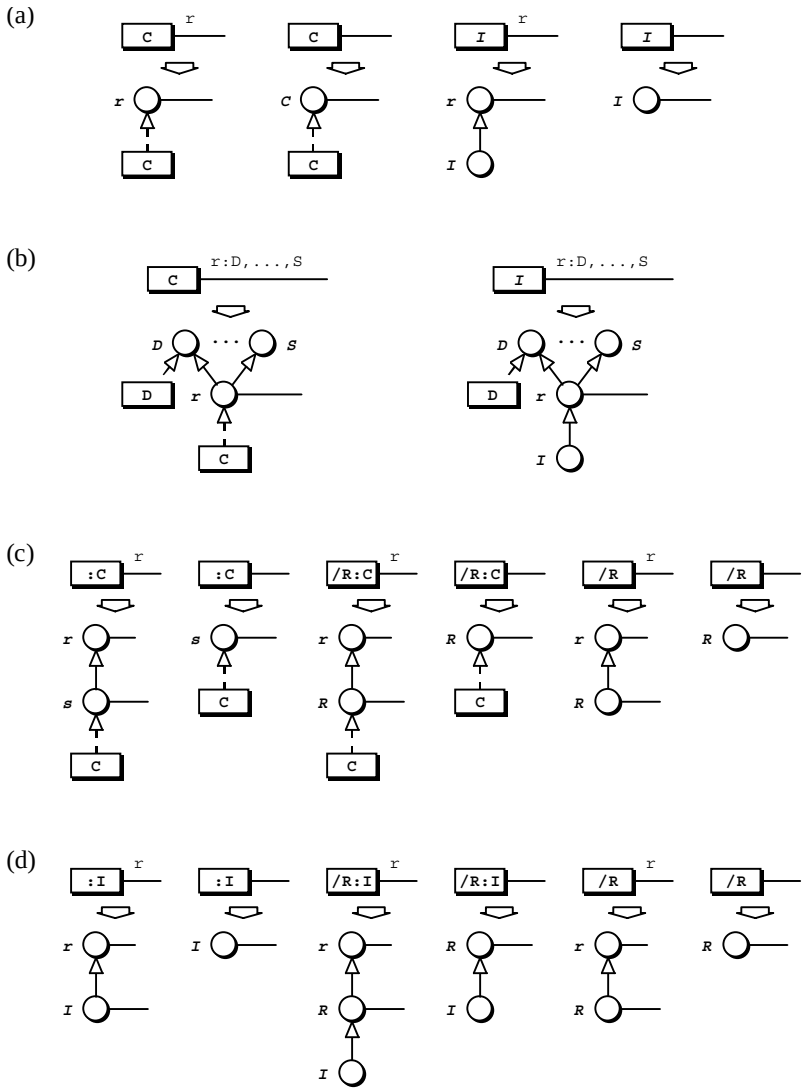

Figure 5: Mapping of old style notation to the new style
(a) for association ends,
(b) for association ends with interface specifiers,
(c) for collaboration roles with classes as bases, and
(d) for collaboration roles with interfaces as bases.

A collaboration diagram does not only introduce classifier roles, but also association roles and association end roles. These roles are implicitly given if (a) the association is unique to collaboration diagrams or (b) the use of the association is restricted in the context of the collaboration (so that the association has a base association). The latter may result from a restriction in multiplicity, navigability, etc. and results in an

overloading of associations (and their ends). Thus, to provide complete mapping rules it would be necessary to consider all other uses of the association. In figure 5, overloading is shown only in the context of classifier roles connecting to association ends with rolenames, because it is assumed that the rolename is heritage from a class diagram with the same association[11] and that this association is being restricted within the scope of the collaboration.

Given these mapping rules the diagrams of figure 2 are easily transformed to the new style diagrams of figure 4 (with multiplicities and overloaded associations omitted). However, a couple of things deserve mention. First, if an association end (of a class diagram) is to be filled with instances of the specified classifier only, the corresponding role must not be populated by other classes. Second, if the same rolename occurs in different associations, it must be checked whether the corresponding roles are actually the same or different. Chances are good that they are the same if they comprise the same set of instances (the role players) and if they come with the same set of features. In case they are different, rolenames can be qualified by their association names.

5 The Big Picture

The definition of roles as presented here is not just another formalization of the same old concept — the relation of roles and classes or *natural types* (as opposed to role types) of individuals as reflected by the metamodel in figure 3 is easily formalized using order sorted predicate logic (including a theory of overloading), and is at the same time deeply founded in disciplines as diverse as sociology, linguistics, and ontology [Steimann 2000]. Particularly the latter two have an obvious influence on modelling, and both leave no doubt that roles and relations are mutually dependent concepts. In this dependency, classes merely serve as the providers of instances (which roles don't have) and as takers of the responsibility for the realization of whatever the roles of a model promise.

The biggest advantage of this separation is that structure and interaction are specified largely independently of the classes that deliver instances and implementation. As a result, classes are kept exchangeable even on the model level. In fact, it seems that modelling with roles delivers on the long requested decoupling of classes in a natural and intuitive way. This has been recognized not only by several object-oriented modelling methods such as OORAM [Reenskaug et al. 1996], OPEN [Firesmith & Henderson-Sellers 1998] and CATALYSIS [D'Souza & Wills 1998], but also in different application areas of object-oriented modelling, for instance by the pattern community [Buschmann 1998] and in framework design [Riehle 2000]. With roles defined as above at hand, instantiating the Composite pattern in a model simply amounts to the involved classes populating (or realizing) the *Component*, *Composite*, and *Leaf* roles.

A major advantage of the introduction of roles as *partial specifications of classes* is that polymorphism and substitutability in modelling are no longer bound to the generalization hierarchy of classes (which mainly serves as an abstraction mechanism

[11] *introducing* rolenames for association ends in collaboration diagrams is redundant, since classifier roles can have rolenames

and, not only in UML, to specify the paths of inheritance), but apply equally to roles and their implementors. The advantage here is that substitutability, if bound to roles, is less demanding than requiring the full substitutability of (the instances of) one class for another[12], because a role promises only limited functionality and behaviour. In fact, the literal meaning of the term polymorphism is that objects (not functions) have different forms, and an object playing different roles may indeed be considered polymorphic in this literal sense [Steimann 1999].

Last but not least, roles give interfaces a prominent conceptual abstraction that would otherwise be missing in object-oriented modelling. Perhaps this lack explains why UML treats interfaces so negligently, but be that as it may, with roles as a natural and intuitive modelling concept (comprising the purposes of interfaces and classifier roles), interfaces should finally get their rightful place in object-oriented modelling.

6 Conclusion

The distinction between class diagrams as static structure diagrams (with classifiers as types and as interface specifiers of association ends) on one side and collaboration diagrams as behaviour diagrams (with classifier roles and their base classifiers) on the other is notoriously difficult to understand (and even more difficult to write about). This unfortunate situation is worsened by the fact that UML's offered diversity of modelling concepts opens the door wide up for introducing inconsistencies into models that are not easily caught unless an automated tool with full mastery of the UML specification is used. In fact, it may be conjectured that the building of such a tool (mapping all diagrams into a single instance of the metamodel) would unveil many more flaws in the UML specification. Keeping this specification is likely to result in practitioners ignoring large parts of it, if only to be on the safe side.

What is equally disappointing is that the UML specification does not promote the use of the interface concept, which is today a proven design and implementation construct. Instead, the metamodel allows it that interfaces and other classifiers are interchangeably used where either one or the other is required. For instance, in UML neither the interface specifiers of association ends nor the base classifiers of classifier roles need be interfaces, although their purpose is clearly that of specifying an interface. Instead, it seems that UML has adopted JAVA's notion of interfaces and follows it rather slavishly to the point that interfaces can only occur at the target ends of directed associations — an undue limitation that is unacceptable for the role concept. CORBA and IDL, other OMG standards, are much more flexible in this regard.

References

[Bläsius et al. 1989]
KH Bläsius, U Hedtstück, CR Rollinger (eds) *Sorts and Types in Artificial Intelligence* Lecture Notes in Artificial Intelligence 418 (Springer 1989).

[12] which is never really given because otherwise there would be no need to have different classes

[Buschmann 1998]
F Buschmann "Falsche Annahmen (Teil 2)" *OBJEKTspektrum* 4 (1998) 84–85.

[D'Souza & Wills 1998]
DF D'Souza, AC Wills *Objects, Components and Frameworks with UML: The CATALYSIS Approach* (Addison-Wesley, Reading 1998).

[Firesmith & Henderson-Sellers 1998]
DG Firesmith , B Henderson-Sellers "Upgrading OML to version 1.1 part 2: additional concepts and notations" *Journal of Object-Oriented Programming* 11:5 (1998) 61–67.

[Gamma et al. 1995]
E Gamma, R Helm, R Johnson, J Vlissides *Design Patterns: Elements of Reusable Object-Oriented Software* (Addison-Wesley 1995).

[OMG 1999]
OMG *Unified Modeling Language Specification* Version 1.3 (www.omg.org, June 1999).

[Övergaard 1999]
G Övergaard "A formal approach to collaborations in the Unified Modeling Language" in: *UML '99* LNCS 1723 (Springer, 1999).

[Reenskaug et al. 1996]
T Reenskaug, P Wold, OA Lehne *Working with Objects — The OORAM Software Engineering Method* (Manning, Greenwich 1996).

[Riehle 2000]
D Riehle *Framework Design: a Role Modeling Approach* doctoral dissertation (ETH Zürich, 2000).

[Rumbaugh et al. 1999]
J Rumbaugh, I Jacobsen, G Booch *The Unified Modeling Language Reference Manual* (Addison-Wesley, Reading 1999).

[Steimann 1999]
F Steimann "On the representation of roles in object-oriented and conceptual modelling" *Data & Knowledge Engineering* to appear. http://www.kbs.uni-hannover.de/~steimann

[Steimann 2000]
F Steimann *Formale Modellierung mit Rollen* thesis (Universität Hannover, 2000).

[Steimann 2001]
F Steimann "Role = Interface: a merger of concepts" *Journal of Object-Oriented Programming* (2001) to appear. http://www.kbs.uni-hannover.de/~steimann