<<UML>>'98

Best of Both Worlds - A Mapping from EXPRESS-G to UML

Dipl.-Inform. Florian Arnold, Dipl.-Inform. Gerd Podehl

Research Group for Computer Application in Engineering Design, University of Kaiserslautern
P. O. Box 3049, 67653 Kaiserslautern, Germany
Phone: +49 631 205 - 3871
Fax: +49 631 205 - 3872
E-mail: arnold@mv.uni-kl.de, podehl@mv.uni-kl.de

Abstract
On the one hand, in the world of Product Data Technology (PDT), the ISO standard STEP (STandard for the Exchange of Product model data) gains more and more importance. STEP includes the information model specification language EXPRESS and its graphical notation EXPRESS-G. On the other hand, in the Software Engineering world in general, mainly other modelling languages are in use - particularly the Unified Modeling Language (UML), recently adopted to become a standard by the Object Management Group, will probably achieve broad acceptance. Despite a strong interconnection of PDT with the Software Engineering area, there is a lack of bridging elements concerning the modelling language level. This paper introduces a mapping between EXPRESS-G and UML in order to define a linking bridge and bring the best of both worlds together. Hereby the feasibility of a mapping is shown with representative examples; several problematic cases are discussed as well as possible solutions presented.

Keywords: UML, Correspondence with other notations, EXPRESS-G

Introduction

Within the world of Product Data Technology (PDT) and Computer Aided technologies (CAx), the need to overcome the proprietary data formats of system suppliers led to the development of the ISO standard STEP (STandard for the Exchange of Product model data). STEP was defined specifically to deal with the information consumed or generated during the product lifecycle from design to manufacturing. STEP includes the information model specification language EXPRESS and its graphical notation EXPRESS-G.

Although the main focus of STEP and PDT is not software, there are some points of contact between the fields of PDT and Software Engineering (SE). For example, there is a need for software to convert data between the various proprietary and system specific file formats and EXPRESS for data exchange purposes. Besides this, software tools for the modelling, specification and manipulation of product data are necessary. In the case of graphical modelling, mainly EXPRESS-G is used in this field. In the SE world in general, mainly other modelling languages are in use though. Particularly the Unified Modeling Language (UML), recently adopted as a standard by the Object Management Group, will probably achieve broad acceptance. So, in spite of the strong interconnection between PDT and SE, there is a lack of bridging elements concerning the modelling language level. This paper introduces a mapping between EXPRESS-G and UML in order to define a linking bridge and bring the best of both worlds together.

In the following, some background information on EXPRESS-G is given first. Then, after a short visit to UML some fundamentals for the mapping are explained. After this, the mapping with examples is presented whereby several problematic cases are discussed and possible solutions presented. At the end, some conclusions are drawn and an outlook is given.

STEP, EXPRESS and EXPRESS-G

STEP (STandard for the Exchange of Product model Data) is an international standard (ISO-10303, Industrial automation systems and integration – Product data representation and exchange) for the computer interpretable representation and the exchange of product model data. Part 11 of STEP is the EXPRESS language reference manual. The formal description language EXPRESS is not a programming language but a specification language for the consistent and logical description of the information models of STEP [Holland 1995]. EXPRESS contains object-oriented and procedural concepts as well as data base concepts. It enables the complete and non-ambiguous description of a mainly static product model. EXPRESS specifies an information domain in terms of entities, i.e.

classes of objects sharing common properties which are represented by associated attributes and constraints. In EXPRESS, constraints are written using a mixture of declarative and procedural language elements.

EXPRESS-G is a formal graphical notation of EXPRESS which can however only reach part of the expressiveness of EXPRESS. The static components like entities, attributes, type declarations and hierarchies of inheritance can be represented by EXPRESS-G. But there is a lack of possibilities to visualise functional components, local or global rules as well as algorithms. Despite this, EXPRESS-G instead of EXPRESS has been chosen for the mapping to enable the evaluation of two graphical representations with examples. Thus it should be possible to compare EXPRESS-G and UML in the categories of readability, clearness, comprehensibility and complexity.

Note that this paper is not intended to explain EXPRESS-G (or even EXPRESS) fundamentals at all. A good introduction to EXPRESS and EXPRESS-G can be found in [Schenck & Wilson 1994].

Unified Modeling Language (UML)

The Unified Modeling Language (UML) defined by the three "amigos" Grady Booch, Jim Rumbaugh and Ivar Jacobson from Rational Software Corporation as a metamodel (and an incidental visual modelling language) for the specification, visualisation, construction and documentation of the artifacts of software systems [UML 1997] is especially suited for the modelling of complex, distributed and concurrent systems [Oestereich 1997]. In November 1997, the UML in its current version 1.1 was adopted to the Object Management Architecture of the Object Management Group and therefore was accepted to become a de facto official industry standard for the metamodel in the area of object-oriented software design.

General remarks on the mapping

The mapping from EXPRESS-G to UML shows the association of model structures with adequate examples. In doing so, all elements of EXPRESS-G are covered and critical cases are highlighted.

Entities, schemas and entity level relations

Because EXPRESS-G does not model dynamics, i.e. it only allows for modelling the *data* of a *static* product model, there are only static structural diagrams (i.e. class diagrams) needed on UML side. Hereby some basic modelling elements are corresponding in a quite obvious way: EXPRESS-G entities are mapped onto classes and schemas to packages (left half of Figure 1) because they share the same semantics.

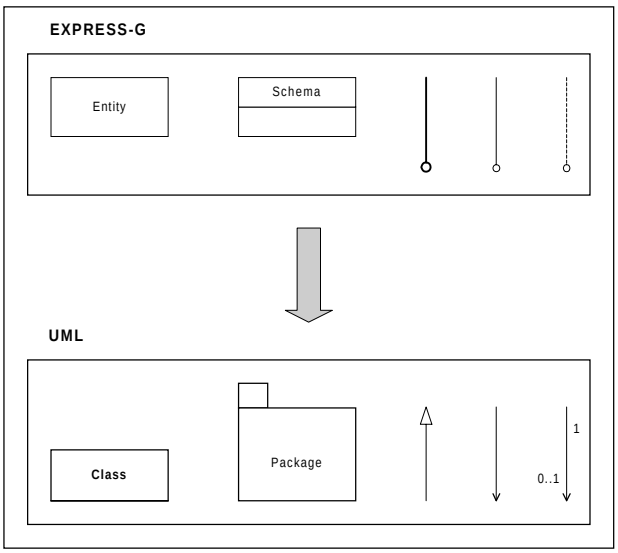

Figure 1: Mapping of some basic elements

EXPRESS-G relations (right half of Figure 1) are bi-directional but nevertheless one direction is emphasised through an open circle. The EXPRESS-G supertype-subtype relation symbol is a thick line with a circle on the subtype end of the relationship and is mapped to the generalisation arrow in UML.

Regarding EXPRESS-G attribute relations, it has to be kept in mind that there is a big difference between the semantics of relational symbols on entity level and on schema level. On entity level, a dashed line with a circle at one end denotes an optional attribute and all other attribute relationships are stated as solid lines with a circle at one end. Thereby the circle is always on the attribute type end. The explicit attribute relationship symbols of EXPRESS-G are mapped to unidirectional (navigable) associations in UML. In the case of optional attributes, a multiplicity of 0..1 is added at the arrow end (compare Figure 1) of the navigable UML association. Derived, inverse and redeclared attributes will be examined in the following chapters. The totally different meaning of relationship symbols on schema level will also be visited later on.

Some primitive types

The EXPRESS-G predefined simple types `Binary`, `Boolean`, `Integer`, `Logical` (`TRUE`, `FALSE` or `UNKNOWN`), `Real`, `Number` (`Integer` or `Real`) and `String` can directly be seen as primitive types in UML.

EXPRESS-G user defined types (like `Date`, Figure 5, and `Strings`, Figure 7) are also mapped to classes. Another possibility would have been to declare them as new user defined primitive types in UML.

The way of defining new user defined primitive types in UML has been chosen for the enumeration type of EXPRESS-G. So the respective enumeration definitions are not noted in UML (as they are not in EXPRESS-G).

Regarding the EXPRESS-G collection types `Set`, `List`, `Array` and `Bag` there are two cases to distinguish. First, a certain collection type is simply assigned to a user defined EXPRESS-G type. In this case, an aggregation association is used in UML (see class `Date`, Figure 5). Second, they are used to state the type of an attribute (see attribute `values` of class `FromEnt`, Figure 7, and attribute `elements` of class `Mesh`, Figure 13). In all cases, the actual kind of collection type can easily be denoted: an unordered set is the default case when denoting a multiplicity greater than one on an association end in UML. A list may be indicated in UML by adding the constraint `{ordered}`, an array by adding `{array}` and a bag by adding `{bag}`.

Complex constraints

In some cases, complex constraints which cannot be directly represented in EXPRESS-G and therefore have to be noted outside the EXPRESS-G diagrams (i.e. in EXPRESS) have been included in the UML representation.

Representational aspects

While assigning names to entities, schemas, classes etc., the different naming conventions of EXPRESS-G and UML have been kept to point out the differences and to illustrate the transferability. It has been waived to map the page referencing symbols of EXPRESS-G because they have no semantic contents and are only used for structuring and formatting purposes of documents in the case of large models.

A mapping with examples

The following EXPRESS-G examples are essentially based on the examples from [Schenck & Wilson 1994] and mainly have the purpose to clarify certain aspects of the usage of this language rather than being examples for good information modelling in general.

Generalisation issues

Figures 2 and 3 show that inheritance trees look very much the same in both notations, with EXPRESS-G being something more explicit by using the (ABS) prefix for abstract classes (Figure 2). The supertype-subtype inheritance hierarchy symbol is labelled with digit 1 highlighting that it is a `OneOf` relation.

In UML, this is realised through a generalisation with a `{disjoint}` constraint. Disjoint applies to a set of generalisations, specifying that instances may have no more than one of the given subtypes as a type of the instance [UML 1997].

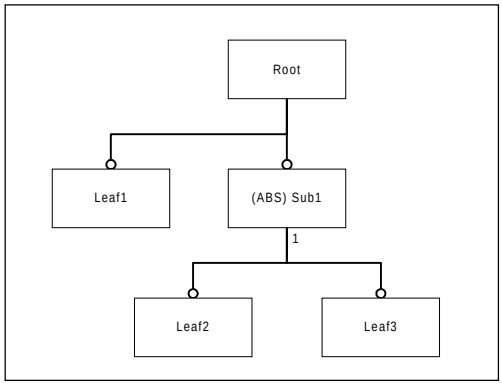

Figure 2: Inheritance in EXPRESS-G

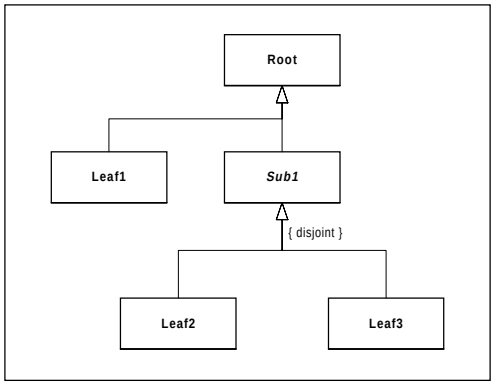

Figure 3: Inheritance in UML

Attributes and constraints

Figure 4 shows the visualisation of a concept `Person` in EXPRESS-G notation. Here, a person has several characteristics like a first name and a last name, an optional nickname, a special type of hair, a date of birth and implicitly a certain age. `Age` has been prefixed with `(DER)`, for derived, to denote that it is a derived attribute.

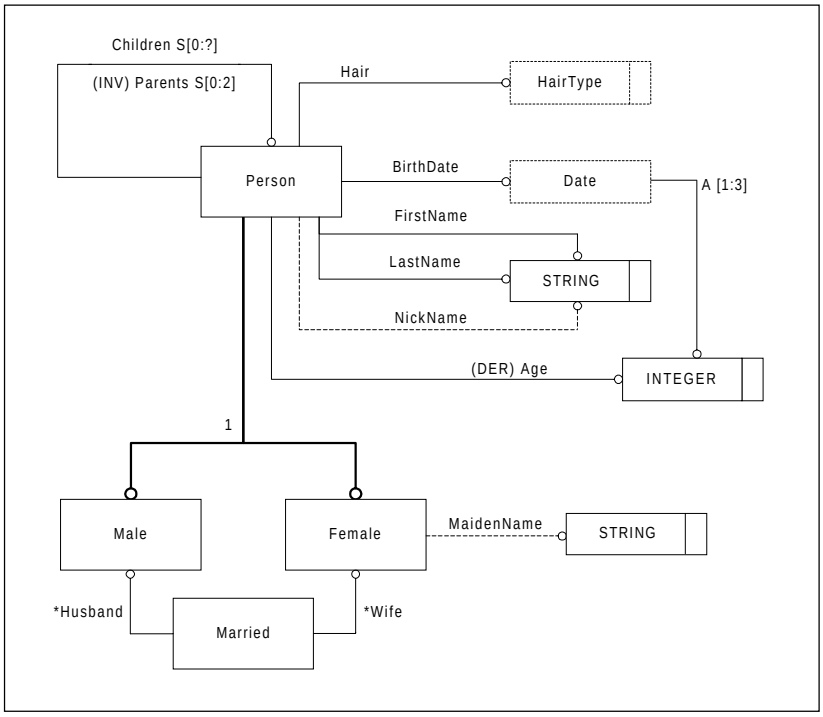

Figure 4: Concept person in EXPRESS-G

The enumeration `HairType : {bald, dyed, natural, wig}` has to be noted outside the diagram. In the example, a person is either female or male. If female, the person optionally has a maiden name. (This relation surely depends on the country's laws and can be regarded being sexistic.) A person may have children and up to two (living) parents, who naturally are persons, too. The attribute `parents` is defined as being inverse to `children` by a

preceding (INV). In EXPRESS-G, an inverse attribute denotes a bi-directional relationship between two entities: an inverse attribute of an entity A references an entity B that itself references entity A [ProSTEP 1994].

A man and a woman may be married whereby in the chosen example polygamy as well as (for equality reasons) polyandry are forbidden through uniqueness constraints, i.e. that the values of husband and wife must be unique across all instances of entity Married. In EXPRESS-G, only the pure existence of these constraints can be displayed by prefixing Husband and Wife with an asterisk while the constraints (no_polyandry and no_polygamy) themselves have to be noted and defined outside the diagram, i.e. in EXPRESS.

When translating this concept to UML (Figure 5), there are several points to be taken into account. The relationship between Person and Male respectively Female is a OneOf relationship, i.e. a person can either be female or male but not both at the same time. A Male and a Female may be married. Hereby the constraints described above can straightforward be denoted in UML.

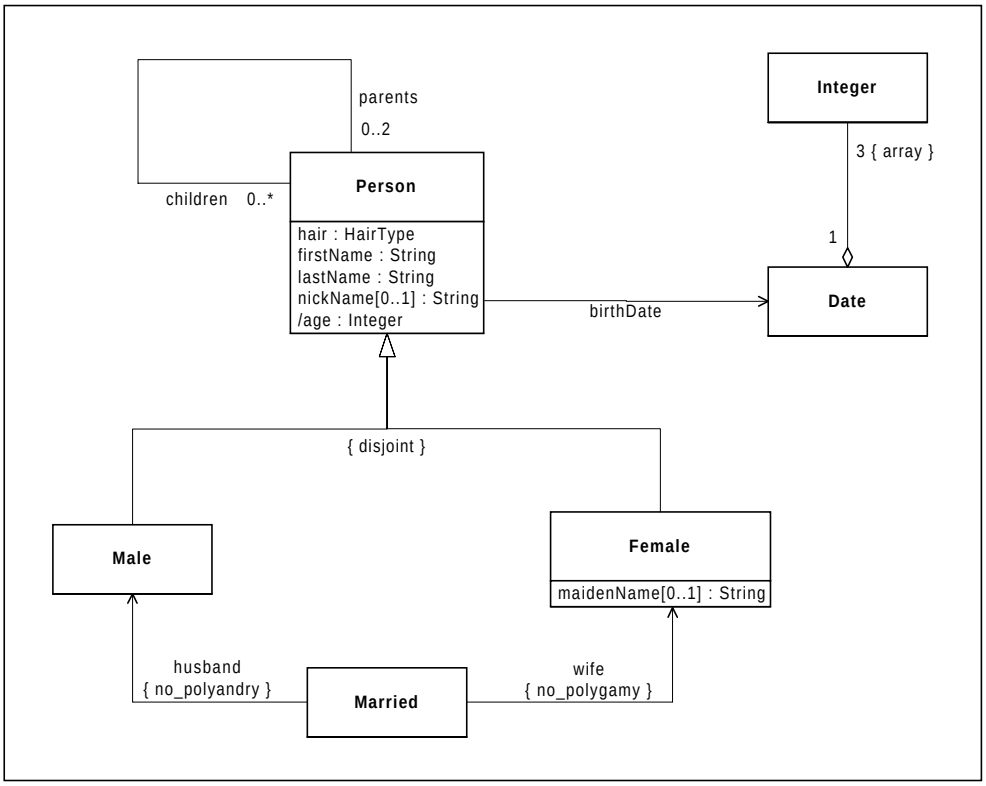

Figure 5: Concept person in UML

Person is modelled in UML (Figure 5) as a class with several attributes. The attribute nickName is marked as optional by the use of [0..1] as can be seen in the attribute compartment of class Person. /age denotes that it is a derived attribute. The children and parents attributes of EXPRESS-G are modelled in UML as one association with the role names children and parents and appropriate multiplicities.

An additional example mainly for the use of EXPRESS-G constraint concepts is given in Figure 6. Pick is of Type select. An EXPRESS-G select type defines a named collection of other types (entity types or defined types). A value of a select type is a value of one of these specified types. The EXPRESS-G symbol for a select type is a dashed rectangle with two vertical lines at the left end and can be modeled in UML by an aggregation association with an {or} constraint (Figure 7). The {or} constraint indicates a situation in which only one of the several potential associations may be instantiated at one time for a single object [UML 1997]. For mapping the defined class Name, the {alias} constraint denotes the simple renaming of the class STRING into Name.

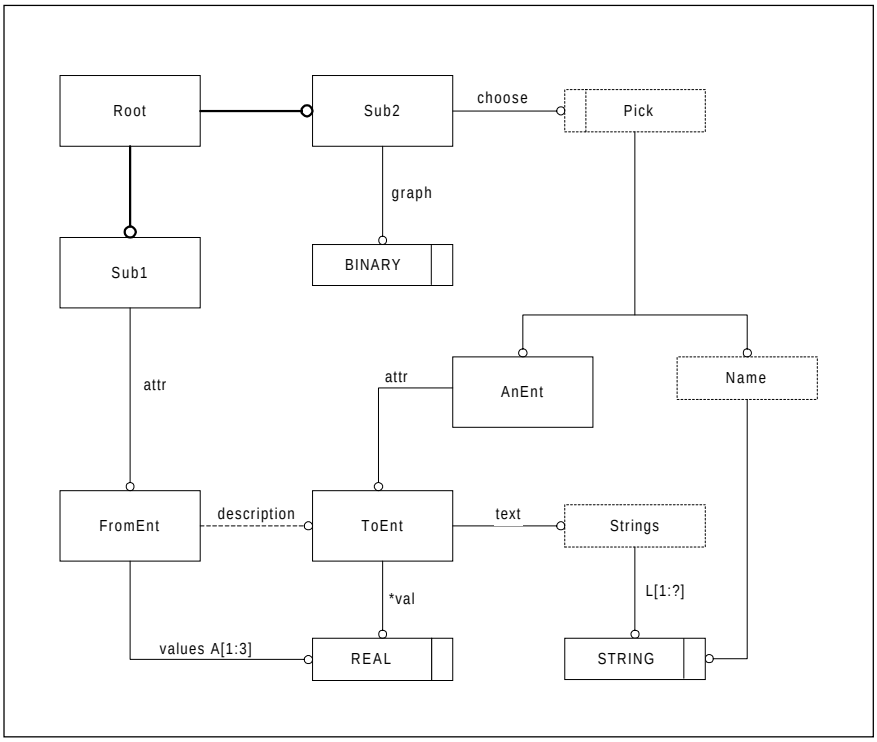

Figure 6: Defined classes, select classes, and constraints in EXPRESS-G

The EXPRESS-G user defined type `Strings` is mapped to an aggregation association while adding an `{ordered}` constraint to state that it corresponds with a `List` collection type in EXPRESS-G. The `val` attribute is constrained as well (Figure 7). The asterisk preceding an attribute denotes that there is a description of the rule in the accompanying documentation (i.e. noted in EXPRESS: `WHERE positive : val >= 0.0`). For this example, a positive value constraint was included into the UML description.

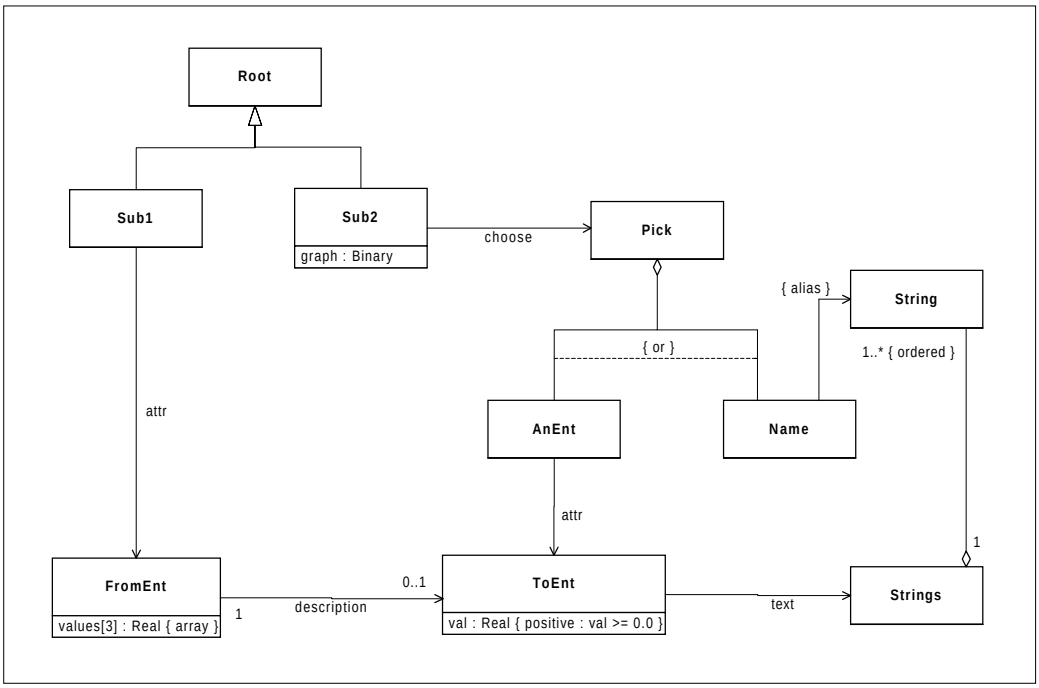

Figure 7: UML notation for figure 6

Attribute redeclaration

Attribute redeclaration is a concept which is specific for EXPRESS respectively EXPRESS-G and has no direct equivalence within UML. A subtype may redeclare its inherited attributes by preceding their name by (RT). In Figure 8, `Leaf` inherits the optional attribute `Attr` from `Middle` and redeclares it to be of type `Sub` which must be a specialisation of the original type `Super` of `Attr`. The same applies to the second redeclaration where the attribute `Num` is redeclared to be of type `INTEGER` instead of type `NUMBER` (which could be `REAL` or `INTEGER`).

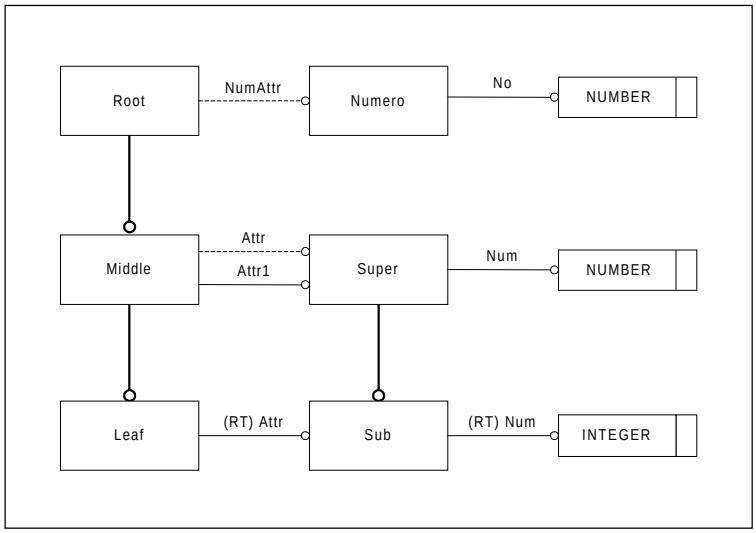

Figure 8: Attribute redeclaration in EXPRESS-G

UML does not support the possibility to inherit the structure of a class and then redeclare it. This is why a trick was used to model attribute redeclaration (Figure 9). `Leaf` inherits only the operations of `Middle` but not the structure (stated by a dashed generalisation arrow [UML 1997]). This makes it possible to again declare `attr` within `Leaf`.

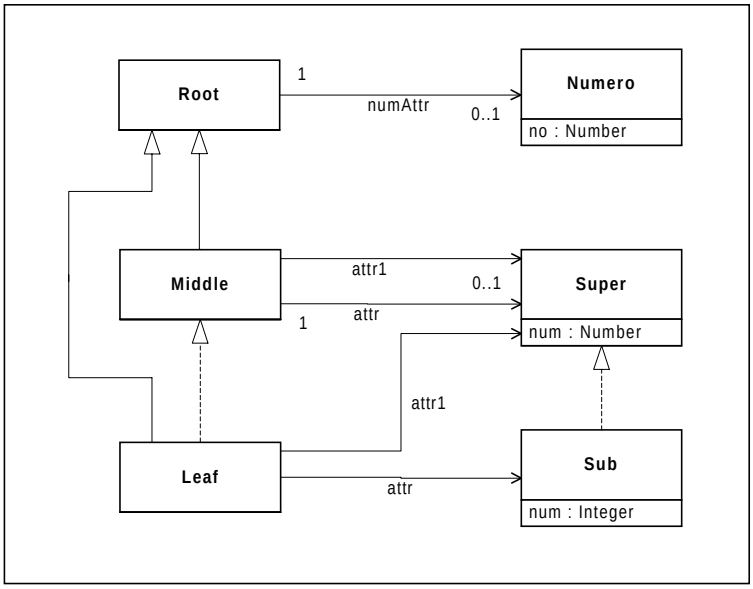

Figure 9: Attribute redeclaration in UML

The drawback of this approach is first that also the attribute `attr1` needs to be declared again. Second, if there is a supertype `Root` of `Middle` it is necessary for `Leaf` to inherit not only the operations but also the attributes from `Root`. Therefore an additional generalisation arrow between `Leaf` and `Root` is inserted. When transferring this to implementation level, care has to be taken in order to avoid conflicts because of the possibility of multiple inheritance of identical operations in `Leaf` from both `Middle` and `Root`.

Schemas

All EXPRESS-G examples investigated until now are entity-level models, i.e. only the contents of one single schema has been displayed. Next we are going to examine a schema-level model, i.e. an example with multiple schemas where solely schemas and their relations are shown.

Remember that on schema level, two EXPRESS-G relation symbols have a meaning which differs from the one on entity level. Sad but true! A dashed line with an open circle at one end denotes a schema-schema 'reference', i.e. instances of referenced entities can only occur when required as attribute values. A normal line with an open circle at one end displays a 'use' relation between schemas, i.e. one or more foreign entity declarations are treated as local declarations.

Figure 10 shows three schemas on schema level with schema `fem` using the entity `property` from schema `mat` and referencing the entity `point` from schema `geom` while giving `point` the alias `node`.

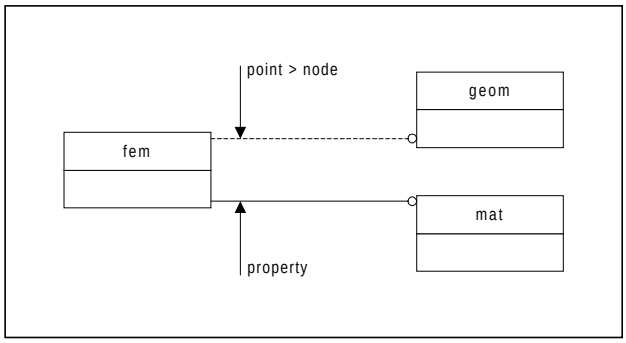

Figure 10: Schema level model in EXPRESS-G

In UML (Figure 11) this complies to a package `Fem`, where the class `Mat::Property` is explicitly imported by using its full qualified pathname. Besides this there is a class `Node` which is associated with class `Point` of package `Geom`. Hereby the property-string `{reference}` has been added to `Node` to indicate the special kind of relationship between `Node` and `Point` which cannot be directly expressed in UML.

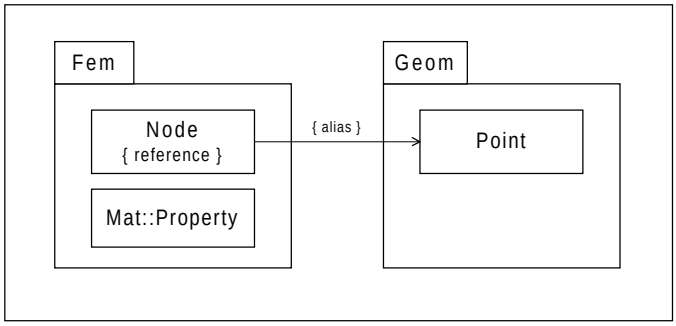

Figure 11: Representation of EXPRESS-G schemas in UML

Another possibility would have been to use property strings `{EXPRESS-G references}` and `{EXPRESS-G uses}` for reasons of a similar treatment of both relationships. In principle, it may also be possible to map these

relationships to stereotypes in UML, but the stereotype `<<uses>>` has already been defined with totally different semantics so this would surely lead to misunderstandings at least.

Inter-schema references

Figure 12 shows an EXPRESS-G model on entity-level that corresponds to the schema-level model of Figure 10 and should clarify the usage of inter-schema references. The entity `element` has two attributes. Thereby `nodes` is a 'reference' to entity `point` (aliased as `node`) from schema `geom`. The fact that it is a 'reference' is expressed by the dashed rectangle. The optional attribute `material` 'uses' (EXPRESS-G semantics) entity `property` from schema `mat`, denoted by the solid rectangle. Entity `mesh` also has two attributes: `elements`, a non-empty list of elements of type `element`, and `material` which also has type `property` from schema `mat`.

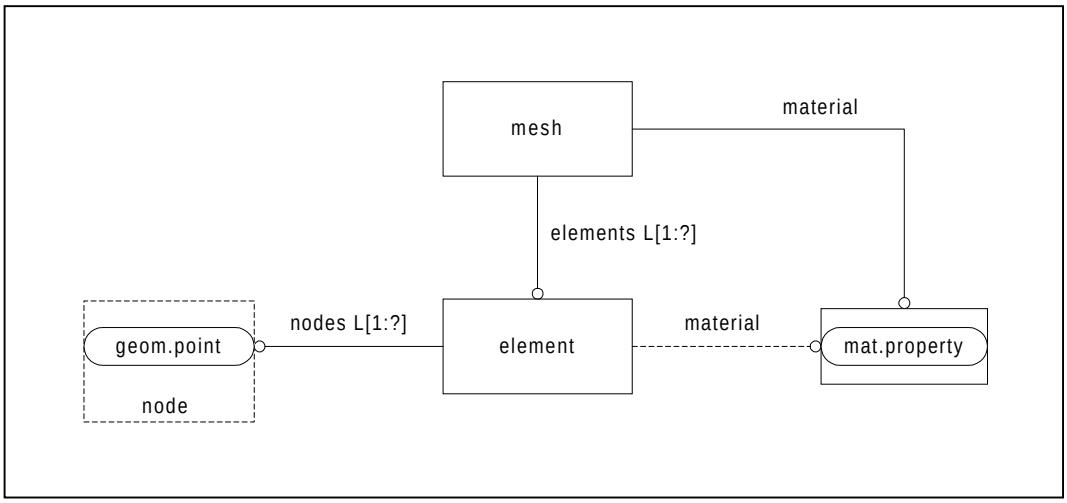

Figure 12: Inter-schema references in EXPRESS-G

Regarding the UML representation (Figure 13) of this example, the following particularities are worthwhile explaining: there is no direct counterpart in UML for the 'uses' and 'references' statements of EXPRESS-G. Thus another way has to be chosen: the 'uses' statement is modelled by the qualified import `Mat::Property` of a class from another package. On the other side, class `Node` is associated with class `Point` from package `Geom` and simply aliases it. The class `Point` itself is thereby not known inside package `Fem` to match the EXPRESS-G 'reference' semantics.

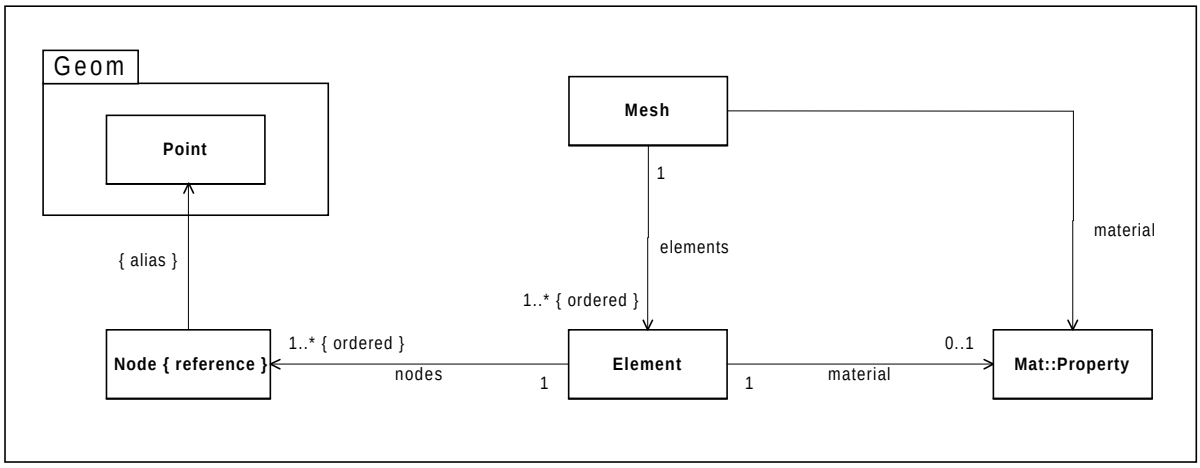

Figure 13: Representation of EXPRESS-G inter-schema references in UML

Conclusion and outlook

The above examples show the feasibility of a mapping between EXPRESS-G and UML. But since EXPRESS-G is not consequently object-oriented and shows some specifics, the mapping to UML is by no means trivial or unambiguous. Not in all cases really elegant solutions are possible but nevertheless a complete and consistent mapping can be realised.

Although the final judgement on the introduced mapping will evolve over time and has to be left to the reader, some further impressions may be noted:

- Having once gotten familiar (to some degree) with both EXPRESS-G and UML there is no general advantage of one notation over the other concerning readability and comprehensibility.
- The expressive power of UML is not at all exhausted to reach the expressiveness of EXPRESS-G (the mapping is not onto at all).
- Complex constraints which cannot be directly represented in EXPRESS-G and have to be attached (in pure EXPRESS notation) to the diagrams can easily be represented in UML. An example can be found in the explanatory text for Figure 6 and Figure 7.

A mapping from EXPRESS to UML is implicitly given for those elements which can be expressed by EXPRESS-G while the rest still has to be examined in detail. Just before delivering the final version of this paper we were notified that since a few weeks there is a CASE tool extension *"that converts your EXPRESS models to UML"* [SoftLab 1998]. It is rather new (available since March 31st, 1998) and we just examined it very roughly in the last days. This tool extension is certainly based on a mapping from EXPRESS to UML, but we have the impression that this mapping is quite different from ours and has surely been developed independently. Their mapping seems not to be final in all details today, e.g. definitions of `select` and user defined types, as well as constraints and inter-schema references are only managed internally but they are not visualised in UML. So we assume that there are other CASE tools evolving in the same direction and that the future will show which tool (and mapping) is most convenient to meet the requirements.

Bibliography

[UML 1997] G. Booch, I. Jacobson, J. Rumbaugh*: "The Unified Modeling Language, Documentation Set 1.1"*, September 1997

[Holland 1995] M. Holland: *"Produktdatentechnologie und STEP"*, STEP Grundschulung, ProSTEP GmbH, Darmstadt, 1995

[Oestereich 1997] B. Oestereich: *"Objektorientierte Softwareentwicklung mit der Unified Modeling Language"*, 3., aktualisierte Auflage (UML 1.0), Verlag R. Oldenbourg, München, 1997

[ProSTEP 1994] ProSTEP GmbH: *"EXPRESS-Grundkurs - Schulungsunterlagen"*, ProSTEP GmbH, Darmstadt, 1994

[Schenck & Wilson 1994] D. Schenck, P. Wilson: *"Information Modeling the EXPRESS Way"*, Oxford University Press, January 1994

[SoftLab 1998] SoftLab AB (SoftLab is a subsidiary of Rational Software Corporation): *"Rational Rose EXPRESS Extension"* (software), http://www.softlab.se/extern/products/express_uml/index.htm, Sweden, available since March 31st, 1998