The Use of Subtypes and Stereotypes in the UML Model

BRIAN HENDERSON-SELLERS, University of Technology, Sydney, Australia

Based on users’ experiences of Version 1.3 of the Unified Modeling Language (UML) of the Object Management Group (OMG), a Request For Information in 1999 elicited several responses which were asked to identify “problems” but not to offer any solutions. One of these responses is examined for “problems” relating to the UML metamodel and here some solutions to the problems identified there are proposed. Specifically, we evaluate the metamodel relating to stereotypes versus subtypes; the various kinds of Classifier (particularly Types, Interfaces and Classes); the introduction of a new subtype for the whole-part relationship; as well as identifying areas in the metamodel where the UML seems to have been used inappropriately in the very definition of the UML’s metamodel.

INTRODUCTION

The use of graphical notations for describing object-oriented models is common. Over the past decades, many dozens of these notations have been created. More recently, it has been widely acknowledged that, to be of use to the developer, these notations must have a unique, unambiguous meaning. This means that all users of a particular graphical notation mean exactly the same and the “modelling language” is understandable by all. In order to create this well-defined modelling language, some kind of formal or semi-formal approach is needed. The adopted approach, often a metamodel, forms the basis for tool implementation and, most importantly for the developer, tool interoperability.

The impetus within the Object Management Group (OMG) to standardize on an OO modelling language had tool interoperability as its main objective. From this OMG initiative came the Unified Modeling Language or UML, which became an Object Management Group (OMG) standard in November 1997. Since that Version 1.1, it has evolved to its current Version 1.3 (OMG, 1999a; OMG, 2000) with a minor 1.4 revision originally due to have been ratified in late 2000. This seems to have occurred in May 2001 (Kobryn, email to OMG’s Analysis and Design Task Force (ADTF) dated 22 May 2001), although the official and publicly available version of V1.4 on the OMG website is dated September 2001 (OMG, 2001). Plans have been made for further evolution and a call for suggested directions was made in 1999 in the form of an RFI (Response For Information). One of the responses to this RFI (OMG, 1999b) listed a number of problems with the UML metamodel – solutions were precluded from the RFI responses.

Here, we examine a subset of the issues relating to the metamodel which were raised in the OMG (1999b) response and then suggest some possible solutions arguing on conceptual, theoretical grounds.

Specifically, we evaluate how stereotypes are defined and how they relate to subtypes in the metamodel and make recommendations for their future use in UML Version 2.0. Some mixed use of stereotypes and subtypes is identified in the area of types, interfaces and classes and this area of the UML metamodel is examined next, with a new metamodel being proposed which avoids force-fitting these concepts into a single inheritance hierarchy in the metamodel. We then take these ideas and see how they appear to have been misused in the very definition of the UML metamodel itself and conclude with a few comments on other areas of concern within the UML metamodel. An underpinning rationale for the evaluation, as well as on theoretical and conceptual grounds, is whether the decisions taken in the metamodel support or confound the ease of learning of the UML.

METAMODELLING

The core of the UML is its metamodel (the other two components being the notation and the Object Constraint Language or OCL). Since the UML metamodel essentially defines the rules by which all models of the software system are depicted (using the UML notation) when developers create software applications, it is clear that these rules (and hence the metamodel) must be as high quality as possible (i.e., be as “correct” as possible) and must be unambiguous as well as, ideally, easy to learn. It is tempting to evaluate a modelling
There are many subtypes, representing basic modelling concepts, already built into the UML metamodel (level M2). However, for concepts not available in the UML, an extension mechanism is needed. UML provides three such extension mechanisms: stereotypes, tagged values and constraints. Of these, stereotypes are the least understood and potentially confusing and are most likely to provoke controversial discussion (Atkinson and Kühne, 2000a) and misuse in practice.

A Stereotype is a GeneralizableElement that can be associated with any ModelElement and in Version 1.3 consists of Constraints and Tagged Values (OMG, 1999a, p2-67; OMG, 2000, p2-70) or Constraints and TagDefinitions in Version 1.4 (OMG, 2001, p2-76). Stereotyping is intended to permit the user effectively to create a (virtual) subtype in the M2 layer without altering the M2 metamodel but by specifying elements in the M1 (model) level. This leads to the UML definition of a stereotype (OMG, 1999a, p2-65; OMG, 2000, p2-68) as “a UML model element that is used to classify (or mark) other UML elements so that they behave in some respects as if there were instances of new “virtual” or “pseudo” metamodel classes”. This virtual metalevel (or M2) class is then described as a specialization of an explicit M2 class (Atkinson, 1999, p24). This means that a stereotyped class (at the M1 level) is an instance of a specific metasubclass (Figure 1) – indeed, Atkinson and Kühne (2000a) suggest that “the stereotype mechanism represents an alternative way of expressing the instantiation relationship without offering any additional modelling power.”

In the example in Figure 1, ControlClass (the metalevel virtual subclass) is a subtype of Class and thus inherits all its metalevel features and relationships. Bird is a (direct) instance of ControlClass and an indirect instance of Class. Consequently, since a Class has an attribute (at the M2 level) of isAbstract, then so does the M2 ControlClass. This is manifested in the M1 level Bird class as a class attribute value, abstract. Any meta-attribute of ControlClass (but not of Class), such as isAbstract, is similarly instantiated in the M1 level class as a class level attribute. In addition, since a (M2) Class possesses Features (Attributes and Operations) so does ControlClass. The M1 level Bird class, which is an instance of the M2 ControlClass, possesses features such as hasFeathers, fly(), which are instances of the Attributes and Operations of class Class. Note that these are obtained indirectly from the M2 class class; connections to M2 classes Attribute and Operation. What comes from the M2 class ControlClass? Anything defined (as a meta-attribute) in ControlClass (but not in Class) is manifested at the class level in Bird. It is thus not possible for the M1 class, here Bird, to obtain actual attributes and operations (which would then be available to instances of Bird) from ControlClass directly. Hence these class level attributes, together with the stereotype «control», are shown in the Name section of the class icon together with any other Constraints applicable to ControlClass but not to Class.

This interplay between subtyping within a metalevel (e.g., within M2 or within M1) and the instance-of relationship appears between metalevels can, as we noted above, cause much confusion (see discussion in Atkinson, Kühne and Henderson-Sellers, 2000a). A typical example is shown in

SUBTYPES AND STEREOTYPES

There are many subtypes, representing basic modelling concepts, already built into the UML metamodel (level M2). However, for concepts not available in the UML, an extension mechanism is needed. UML provides three such extension mechanisms: stereotypes, tagged values and constraints. Of these, stereotypes are the least understood and potentially confusing and are most likely to provoke controversial discussion (Atkinson and Kühne, 2000a) and misuse in practice.

A Stereotype is a GeneralizableElement that can be associated with any ModelElement and in Version 1.3 consists of Constraints and Tagged Values (OMG, 1999a, p2-67; OMG, 2000, p2-70) or Constraints and TagDefinitions in Version 1.4 (OMG, 2001, p2-76). Stereotyping is intended to permit the user effectively to create a (virtual) subtype in the M2 layer without altering the M2 metamodel but by specifying elements in the M1 (model) level. This leads to the UML definition of a stereotype (OMG, 1999a, p2-65; OMG, 2000, p2-68) as “a UML model element that is used to classify (or mark) other UML elements so that they behave in some respects as if there were instances of new “virtual” or “pseudo” metamodel classes”. This virtual metalevel (or M2) class is then described as a specialization of an explicit M2 class (Atkinson, 1999, p24). This means that a stereotyped class (at the M1 level) is an instance of a specific metasubclass (Figure 1) – indeed, Atkinson and Kühne (2000a) suggest that “the stereotype mechanism represents an alternative way of expressing the instantiation relationship without offering any additional modelling power.”

In the example in Figure 1, ControlClass (the metalevel virtual subclass) is a subtype of Class and thus inherits all its metalevel features and relationships. Bird is a (direct) instance of ControlClass and an indirect instance of Class. Consequently, since a Class has an attribute (at the M2 level) of isAbstract, then so does the M2 ControlClass. This is manifested in the M1 level Bird class as a class attribute value, abstract. Any meta-attribute of ControlClass (but not of Class), such as isAbstract, is similarly instantiated in the M1 level class as a class level attribute. In addition, since a (M2) Class possesses Features (Attributes and Operations) so does ControlClass. The M1 level Bird class, which is an instance of the M2 ControlClass, possesses features such as hasFeathers, fly(), which are instances of the Attributes and Operations of class Class. Note that these are obtained indirectly from the M2 class class; connections to M2 classes Attribute and Operation. What comes from the M2 class ControlClass? Anything defined (as a meta-attribute) in ControlClass (but not in Class) is manifested at the class level in Bird. It is thus not possible for the M1 class, here Bird, to obtain actual attributes and operations (which would then be available to instances of Bird) from ControlClass directly. Hence these class level attributes, together with the stereotype «control», are shown in the Name section of the class icon together with any other Constraints applicable to ControlClass but not to Class.

This interplay between subtyping within a metalevel (e.g., within M2 or within M1) and the instance-of relationship appears between metalevels can, as we noted above, cause much confusion (see discussion in Atkinson, Kühne and Henderson-Sellers, 2000a). A typical example is shown in

SUBTYPES AND STEREOTYPES

There are many subtypes, representing basic modelling concepts, already built into the UML metamodel (level M2). However, for concepts not available in the UML, an extension mechanism is needed. UML provides three such extension mechanisms: stereotypes, tagged values and constraints. Of these, stereotypes are the least understood and potentially confusing and are most likely to provoke controversial discussion (Atkinson and Kühne, 2000a) and misuse in practice.

A Stereotype is a GeneralizableElement that can be associated with any ModelElement and in Version 1.3 consists of Constraints and Tagged Values (OMG, 1999a, p2-67; OMG, 2000, p2-70) or Constraints and TagDefinitions in Version 1.4 (OMG, 2001, p2-76). Stereotyping is intended to permit the user effectively to create a (virtual) subtype in the M2 layer without altering the M2 metamodel but by specifying elements in the M1 (model) level. This leads to the UML definition of a stereotype (OMG, 1999a, p2-65; OMG, 2000, p2-68) as “a UML model element that is used to classify (or mark) other UML elements so that they behave in some respects as if there were instances of new “virtual” or “pseudo” metamodel classes”. This virtual metalevel (or M2) class is then described as a specialization of an explicit M2 class (Atkinson, 1999, p24). This means that a stereotyped class (at the M1 level) is an instance of a specific metasubclass (Figure 1) – indeed, Atkinson and Kühne (2000a) suggest that “the stereotype mechanism represents an alternative way of expressing the instantiation relationship without offering any additional modelling power.”

In the example in Figure 1, ControlClass (the metalevel virtual subclass) is a subtype of Class and thus inherits all its metalevel features and relationships. Bird is a (direct) instance of ControlClass and an indirect instance of Class. Consequently, since a Class has an attribute (at the M2 level) of isAbstract, then so does the M2 ControlClass. This is manifested in the M1 level Bird class as a class attribute value, abstract. Any meta-attribute of ControlClass (but not of Class), such as isAbstract, is similarly instantiated in the M1 level class as a class level attribute. In addition, since a (M2) Class possesses Features (Attributes and Operations) so does ControlClass. The M1 level Bird class, which is an instance of the M2 ControlClass, possesses features such as hasFeathers, fly(), which are instances of the Attributes and Operations of class Class. Note that these are obtained indirectly from the M2 class class; connections to M2 classes Attribute and Operation. What comes from the M2 class ControlClass? Anything defined (as a meta-attribute) in ControlClass (but not in Class) is manifested at the class level in Bird. It is thus not possible for the M1 class, here Bird, to obtain actual attributes and operations (which would then be available to instances of Bird) from ControlClass directly. Hence these class level attributes, together with the stereotype «control», are shown in the Name section of the class icon together with any other Constraints applicable to ControlClass but not to Class.

This interplay between subtyping within a metalevel (e.g., within M2 or within M1) and the instance-of relationship appears between metalevels can, as we noted above, cause much confusion (see discussion in Atkinson, Kühne and Henderson-Sellers, 2000a). A typical example is shown in

SUBTYPES AND STEREOTYPES

There are many subtypes, representing basic modelling concepts, already built into the UML metamodel (level M2). However, for concepts not available in the UML, an extension mechanism is needed. UML provides three such extension
Related Content

Privacy in Multidimensional Databases
[www.igi-global.com/chapter/privacy-multidimensional-databases/26973?camid=4v1a](www.igi-global.com/chapter/privacy-multidimensional-databases/26973?camid=4v1a)

Semistructured Data and its Conceptual Models
[www.igi-global.com/chapter/semistructured-data-its-conceptual-models/11212?camid=4v1a](www.igi-global.com/chapter/semistructured-data-its-conceptual-models/11212?camid=4v1a)

Using Business Rules within a Design Process of Active Databases
[www.igi-global.com/article/using-business-rules-within-design/3251?camid=4v1a](www.igi-global.com/article/using-business-rules-within-design/3251?camid=4v1a)

A Unified Fuzzy Data Model: Representation and Processing
[www.igi-global.com/article/unified-fuzzy-data-model/62033?camid=4v1a](www.igi-global.com/article/unified-fuzzy-data-model/62033?camid=4v1a)