

Verbalizing Business Rules: Part 3

*Terry Halpin
Northface University*

Business rules need to be validated by business domain experts, so should be specified using concepts and languages (textual or graphical) easily understood by business people. This is the third in a series of articles on expressing business rules formally in a high-level, textual language. The first article [4] listed criteria for a business rules language, and discussed verbalization of simple uniqueness and mandatory constraints on binary associations. The second article [5] discussed how to use hyphen-binding to improve constraint verbalization, and examined verbalization of internal uniqueness constraints that either span a whole association, or apply to n-ary associations. The current article discusses verbalization of basic external uniqueness constraints—these apply to roles from two or more predicates (allowing that some of these predicates may be represented as attributes in non-ORM notations).

External Uniqueness Constraints

The output report shown in Table 1 provides some details about buildings and rooms. The “?” entry denotes a null value—in this application domain, not all buildings are named, and not all buildings need have rooms recorded. As an exercise, try to model this in your favorite data modeling notation before reading on.

Table 1 Building and Room details.

<i>BuildingNr</i>	<i>BuildingName</i>	<i>RoomNr</i>	<i>HasWindow</i>
1	Presley	301	No
1	Presley	502	Yes
2	?	301	Yes
3	Armstrong	?	?

Figure 1 shows a basic solution in the Unified Modeling Language (UML). Here the Room class has two mandatory attributes, roomNr and isWithWindow, presumably based on numeric and Boolean types respectively. Attribute domains may be displayed on the UML class diagram if desired. The Building class has one mandatory attribute (buildingNr), and one optional attribute (buildingName). By default, attributes in UML have a multiplicity of 1 (exactly 1). The [0..1] multiplicity on buildingName indicates that an entry for buildingName is optional, and may have at most one value. Assuming location semantics, the multiplicities on the Room-Building association indicate that each room is located in exactly one building, and each building includes zero or more rooms.

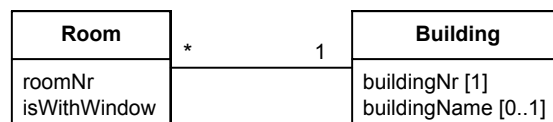


Figure 1 A basic UML class diagram for the application domain described in Table 1.

As you no doubt realized, the model in Figure 1 is missing some constraints. To begin with, it does not display the following uniqueness constraints:

- each** buildingNr refers to **at most one** Building
- each** buildingName refers to **at most one** Building

Although UML has no standard graphic notation for specifying uniqueness constraints on attributes, you may attach these constraints textually in notes. Alternatively, you may invent your own non-standard graphic notation for uniqueness. For example, you might append {U1} to buildingNr, and append {U2} to buildingName, as shown in Figure 2(a) to indicate that each of these attributes has a unique constraint.

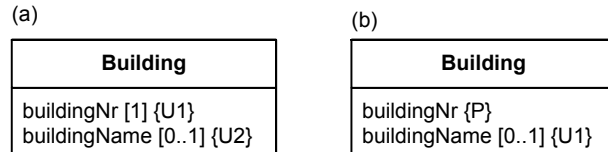


Figure 2 Adding some constraints in non-standard notations.

Because of its roots in object-oriented code design, UML tends to assume that each class instance is identified by some surrogate identifier such as a memory address or object identifier (oid). For communication with business people however, we need to use visible, *value-based identification schemes* (even if oids are used as well in the implementation). In this example, all buildings may be identified by a building number, but only some buildings may be identified by a building name. In this case, building number provides a value-based identification scheme that can be used in human communication. An identification scheme is also known as a *reference scheme*.

If we modified our example to require building names to be both mandatory and unique, then building names may also be used for an identification scheme. In that case, we would have two candidate identification schemes for building: buildingNr, and buildingName. Suppose that the business itself considers it relevant to choose one of these to be the *primary* identification scheme for buildings. For example, the business might choose buildingNr rather than buildingName to be the primary means of identification (in human communication), because building numbers never change but building names might occasionally change. In such cases, we may treat the notion of primary identification as a conceptual issue, and we may introduce a non-standard graphic notation to depict this in UML. For example, the {P} constraint in Figure 2(b) indicates that buildingNr is (at least part of) the primary identifier for Building. Note that this notion of primary reference does not imply that buildingNr must be chosen as the “primary key” in a relational implementation (even if this is likely); it simply records a business decision.

Even if we add these new notations to Figure 1, the model still lacks an important constraint. No identification scheme is provided for Room. Would it be OK to use roomNr for this? The data in Table 1 indicates that the same roomNr (e.g. 301) may refer to more than one room. Here the term “roomNr” is used in a local sense, not a global sense. Only within the context of a single building can we be sure that any instance of roomNr refers to only one Room. So our room identification scheme needs to be based on the constraint that room numbers are unique within a given building. There is no convenient graphic notation in UML to do this. One might resort to qualified associations or even objectified associations, but neither of these approaches is satisfactory (see pp. 369-70 of [5]). The best we can do in UML is to add this constraint as a textual note attached to the relevant attribute and association end as shown in Figure 3.

This informal note may be replaced by a formal expression in UML’s Object Constraint Language (OCL), but the current syntax for OCL is too technical in nature to be reliably validated with non-technical business people. The latest version of OCL (OCL 2.0) allows the possibility of using alternative syntaxes that are more business-friendly. I’ll return to this issue in the next article, when I discuss alternative constraint verbalizations that allow role or attribute names to be used in place of predicate names.

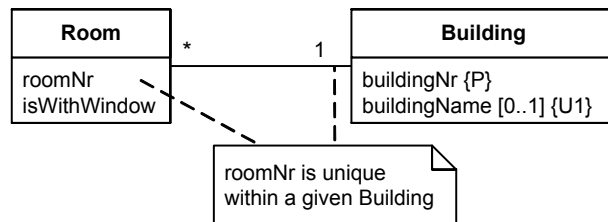


Figure 3 Adding an external uniqueness constraint in a note.

The identification (reference) scheme discussed for Room is called a *composite* reference scheme, because it includes at least two components in the identification scheme—here the attribute roomNr, and the room containment role played by Building. While all versions of Entity Relationship (ER) modeling support the notion of uniqueness constraints on attributes, and even primary identification schemes, most ER versions are very limited in their support for composite reference schemes, typically allowing only one composite reference scheme per entity type. Moreover, this composite reference scheme is typically restricted to multiple attributes within the same entity type. For example, in IDEF1X the identification scheme for Room would be specified using buildingNr and roomNr attributes within the Room entity type, and declaring buildingNr to be a foreign key to the Building entity type (see [5], p. 336). Although the buildingNr foreign key may be considered to be migrated from an identifying relationship with Building, the identification scheme cannot be specified without the foreign key. Since the concept of a foreign key is a relational database concept, not a business concept, such a sub-conceptual representation is unsuitable for communication with non-technical domain experts and will not be discussed further.

Of all the widely used, industrial ER versions, the Barker ER version [1] made popular by Oracle provides the best support for composite identification schemes. Figure 4 shows how the example may be diagrammed in this notation. Notice that attributes may be prepended by various marks. A “*” or “o” indicates that the attribute is mandatory or optional respectively. The “#” indicates that the attribute is a component of the primary identification scheme for the entity type. In the case of Building, the building_nr attribute is the only primary component of the identification scheme. In the case of Room, the room_nr attribute is only part of the identification scheme, because the stroke “|” across the association role played by Room indicates that the co-role (played by Building) is also a component of the identification scheme. Hence this captures graphically the constraint that a room is primarily identified by combining its room number with the building that contains it.

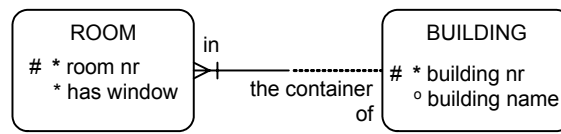


Figure 4 The Building-Room example diagrammed in Barker ER notation.

Although for this example the Barker ER notation does a better job of capturing the semantics than UML’s standard graphic notation, it is still lacking. See if you can spot the missing constraint before reading on.

As you probably noted, the model in Figure 4 fails to capture the uniqueness constraint on the building name attribute. Unfortunately, the Barker ER notation’s declaration of unique attributes is restricted to those that are components of the primary identifier. The missing uniqueness constraint needs to be added somewhere as a note.

Figure 5 shows how the same example is diagrammed using the Object-Role Modeling (ORM) approach. As discussed in previous articles, ORM makes no use of attributes in its base models, always using a relationship instead of an attribute. Here the unary fact type Room has a window is used to record whether a room has a window. By default, a closed world assumption is applied to this unary predicate, so it is equivalent to the mandatory Boolean attribute used in the other models.

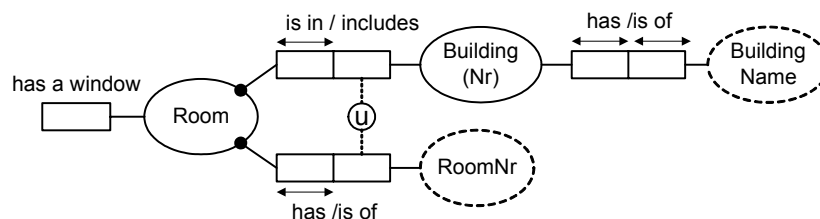


Figure 5 The Building-Room example diagrammed in ORM notation.

Using the verbalization technique discussed in [4], the internal mandatory and uniqueness constraints on the explicit binary relationships may be verbalized thus:

Each Room is in **exactly one** Building.
 Each Room has **exactly one** RoomNr.
 Each Building has **at most one** BuildingName.
 Each BuildingName is of **at most one** Building.

As shown in Figure 6, ORM regards the simple reference scheme Building(Nr) as an abbreviation for the identification relationship Building is identified by / refers to BuildingNr. This relationship is injective (mandatory, 1:1, into), so implicitly the following constraints also hold:

Each Building is identified by **exactly one** BuildingNr.
 Each BuildingNr refers to **at most one** Building.

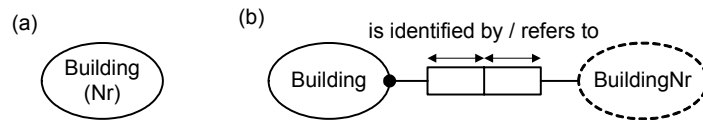


Figure 6 A reference mode in parenthesis (a) abbreviates an identification relationship (b).

In Figure 5, the circled “u” attached to roles played by Building and RoomNr denotes an *external uniqueness constraint* (“u” for “unique”). Unlike internal (intra-predicate) constraints, external (inter-predicate) constraints apply to roles from two or more predicates. This external uniqueness constraint may be formally defined as equivalent to an internal uniqueness constraint applied to the building and roomNr roles bag-projected from the derived association obtained by conceptual joining the Room is in Building and Room has RoomNr associations. For further details on this point, see [2, sec. 4.4] and [3]. The constraint may be formally verbalized in positive form thus:

Given any Building and RoomNr
there is at most one Room that is in that Building
and has that RoomNr.

The phrase “Given any” may be also be rendered as “For each”. In negative form, the external uniqueness constraint verbalizes thus:

It is impossible that
more than one Room is in the same Building and has the same RoomNr.

In general, an entity type may have many candidate identification schemes. If we wish to indicate that an identification scheme is the primary one, this may be displayed graphically by using “P” (for Primary) instead of “u” in the constraint symbol, as shown in Figure 7. In this case, the following is appended to the positive constraint verbalization:

Room is primarily identified by this unique combination.

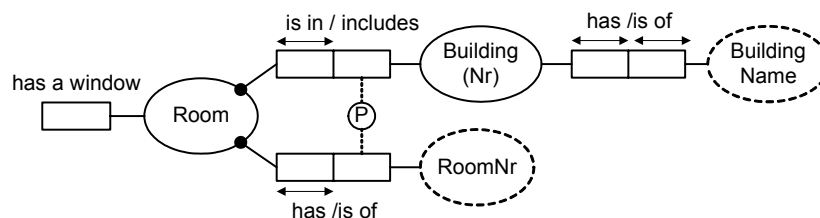


Figure 7 The composite reference scheme is declared to be the primary identifier.

If the external uniqueness constraint spans at least two roles played by the same object type, the object variable instances are distinguished by subscripts instead of the pronoun “that”, as discussed in previous articles. For example, the external uniqueness constraint in Figure 8 verbalizes in positive form thus:

Given any Date₁ **and** Date₂
there is at most one Period **that** began on Date₁
and ended on Date₂.

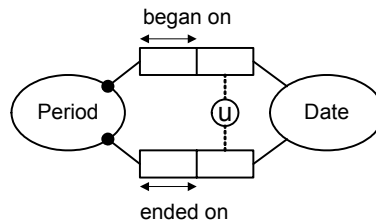


Figure 8 Multiple constrained roles played by the same object type.

So far, all our constraint verbalization examples have used predicate verb phrases (e.g. began on). Alternative verbalizations are possible by using role names (e.g. startDate, endDate). This role-name technique also provides a way of verbalizing various constraints in attribute-based notations such as UML and ER. The next article considers such alternative verbalizations, and also caters for more complex cases of external uniqueness constraints that involve either nesting (objectified associations) or lengthy join paths.

References

1. Barker, R. 1990, *CASE*Method: Tasks and Deliverables*, Addison-Wesley, Wokingham, England.
2. Halpin, T.A. 2001, *Information Modeling and Relational Databases*, Morgan Kaufmann, San Francisco.
3. Halpin, T.A. 2002, ‘Join Constraints’, *Proc. EMMSAD’02: 7th Int. IFIP WG8.1 Workshop on Evaluation of Modeling Methods in Systems Analysis and Design*, Toronto (June, 2002), available online at <http://www.orm.net/pdf/JoinConstraints.pdf>.
4. Halpin, T. A. 2003, ‘Verbalizing Business Rules: Part 1’, *Business Rules Journal*, Vol. 4, No. 4 (April 2003), URL: <http://www.BRCommunity.com/a2003/b138.html>.
5. Halpin, T. A. 2003, ‘Verbalizing Business Rules: Part 2’, *Business Rules Journal*, Vol. 4, No. 6 (June 2003), URL: <http://www.BRCommunity.com/a2003/b152.html>.