



June 22, 2007

Cardinality:

What it is, and how we use it in agcXML

Cardinality refers to the range, or number of times, a value may be entered for a particular data element in a data set. Any cardinality from zero to infinity [0..*] is possible. The narrowest range is a cardinality of [1]. A data element with a cardinality of [1] is a required data element (zero is not an option, so neither is not entering a value). A value for that data element must be entered in every data set once, but can be entered no more than once. For example, the data element "Agreement Date" in a schema for Owner / Contractor Agreement would be assigned a cardinality of [1]. The parties must enter an agreement date into the data set, but can only enter one. Assigning cardinality to a data element, therefore, is a type of rule. But as you will see from the following discussion, it is a type of rule designed to preserve maximum flexibility within a data set and maximize the options for software companies at the software implementation level.

If every data element in a data set had a cardinality of [1], there would be no flexibility in the XML schema. Every data element would be a required element, and only a single value could be entered for each data element. If AEC business practices were so rigidly codified, there would be no need for consensus standards such as agcXML. The reality is that even a basic document such as an Owner / Contractor Agreement contains many optional data elements, and some data elements may have multiple values. If we wish to include all possible data elements for "Owner / Contractor Agreement" in a single schema, one of the ways to begin handling these options is to assign the appropriate cardinality to each data element in the schema.

A cardinality of [0..1] means that the number of possible values for that data element is between zero and one. In other words, entering a value for that data element is optional. If one chooses to enter a value, only one value can be entered. For example, for most construction projects, the default project start date is the same as the Agreement Date. But many standard Owner / Contractor Agreements allow the parties the option of entering a "Project Start Date" into the agreement that is different from the Agreement Date. A cardinality of [0..1] allows the parties the option of including a Project Start Date in the data set that is different from the Agreement Date. But if they do include it, they can only enter one project start date; a cardinality of [0..1] does not permit the parties to enter two project start dates. We can safely assign this cardinality to this data element based on what we know about common business practice.

A cardinality of [0..2] means that entering a value is optional, but up to two values can be entered. A cardinality of [1..2] means that entering a value is not optional (there must be at least one), but up to two values can be entered. Again, any cardinality from zero to infinity [0..*] is possible. Assigning fixed numbers to the lower or upper ends of the range is a design decision; it prescribes the number of possible values that can be entered for that data element. We can only assign cardinality of a fixed range if we have sufficient confidence about how that data element is commonly used in practice.

A cardinality of [0..*] means that entering a value is optional, but there is no limit to the number of values that can be entered for that data element. This cardinality is useful when the number of possible values for a data element in a particular data set is unknown to the schema or software designer, and must be left to the end user to decide. For example, an Owner/Contractor Agreement may (or may not) include a list of Drawings. If a list of drawings is included, the number of drawings may vary with the size and scope of the project. And for each individual drawing, values must be entered for several descriptive data fields: Drawing Number, Drawing Title, and Drawing Date. Each of these data elements, therefore, has a cardinality of [0..*]. Each data element may (or may not) be included, but if they are, there is no limit to the number of values that may be entered.

Cardinality is not sufficient for handling the data elements in the above example. Additional rules would have to be embedded in the software at the user-interface level. For example, the software might first ask the user to respond to the yes/no question, "Do you wish to include a list of drawings in this Agreement?" If the answer is "No," the data fields for Drawing Number, Drawing Title, and Drawing Date would be hidden from the user or "grayed out." If the answer is "Yes," then the user would be required to enter at least one value for Drawing Title and Drawing Date for each Drawing Number. In this way, the cardinality of each data element is combined with higher-level rules at the software level to allow the user to properly complete a document or business process.

One Schema or Many?

Within any "generic" business process or document type, there are many data elements (or groups of data elements) that are needed for some transactions or documents but not others. A design decision has to be made that establishes how many data element options will be allowed within a single schema. If a single schema includes all possible data elements for a "generic" business process or document such as "Owner / Contractor Agreement," then the software applications that generate agreements may have to be more complex and the end user may have to make many decisions each time an Agreement is drafted. On the other hand, if multiple schemas are developed for defined subtypes, the number of options for end users within any individual schema may be greatly limited.

For example, in a "multiple schema" approach, one schema could be developed for Owner / Contractor Agreements in which the terms of compensation are lump sum, and another for Owner / Contractor Agreements that are cost-plus. In contract document software, the user interface would be simpler, with fewer options to choose, but the choice of certain options (in this case, terms of compensation) would be elevated to a higher level than others. Also, many common data elements would have to appear in both schemas, which introduces a different set of challenges for schema design and revision.

In the end, the more specialized the schema, the less flexible it is. Users would have to make a choice of schema without the full benefit of knowing which data elements would be included, and making or changing the "higher level" decision becomes more difficult and cumbersome. For example, if the parties to an Agreement have not yet decided on the compensation terms, they cannot begin to prepare an Agreement until they do. And if they subsequently decide to change the compensation terms before signing the Agreement, they might have to scrap the original draft agreement and start over with another using a different schema.

The results of the comparative analysis of ten AIA and AGC standard form Owner / Contractor Agreements produced 184 possible data fields, which we have organized into 13 general categories of information. With the possible exception of the contact information categories, any of these categories could arguably be used to define a subtype of Owner / Contractor Agreement. The most obvious category for defining subtypes is "Compensation Provisions," but even a cursory review of the comparative analysis reveals that there is little "exclusive commonality" of data elements in that category.

For these reasons, we are electing at this time to develop a single agcXML schema for owner / contractor agreements. Your thoughts? Please post your comments on the agcXML listserv.

And now that you understand the concept of cardinality, please review the comparative analysis and let us know if, in your opinion, the correct cardinality has been assigned to each proposed data element.