Guidance for Determining Cardinality

Properties:

Properties should be 0..1 to 0..n unless one of the following are true:

1. Property has a default value
    a. Example: isUniversallyUnique in AnnotatedIdentifiable
2. Property has a fixed value
    a. Example: totality, reflexivity, symmetry, transitivity defining profile for a BinaryRelation
3. The properties are required for the class to function and the class is always optional
    a. Example: Content in a StructuredString

Relations:

Source Cardinality:

The possible number of **SOURCE** instances with which the TARGET object may have the stated relationship.

1. All **Composition** relationship types should have cardinality 1..1 as there is a lifecycle dependency between the part (target class) and the whole (source class), i.e. if the whole is deleted the part is deleted
    a. 1..1 indicates that the part (the target class) cannot exist except as part of the whole
2. **Aggregation** relationship types have a target class that has an existence independent of the source class may have the any of the following cardinalities:
    a. 0..n when the target class can exist independently and belong to multiple source classes
    b. 0..1 when the target class can exist independently but belong to only one source class
3. A **Simple Association** includes any relationship that does not imply a whole/part relationship
    a. 0..n would be the default source cardinality

Target Cardinality:

The possible number of TARGET instances with which the SOURCE object may have the stated relationship.

Target cardinality regardless of the relationship type should be 0..1 to 0..n to allow for:

- Flexibility in the process of producing metadata
- Restriction of the source class for its use by a Functional View

Target cardinality should be restricted to 1..1 to 1..n only if the following are true:

- The content of the target class is required for the usefulness of the source
- The information for the class is available at all points in the process of producing metadata where the source class would be used
- Restricting the source class by not including the target class in a Functional View renders the source class unusable