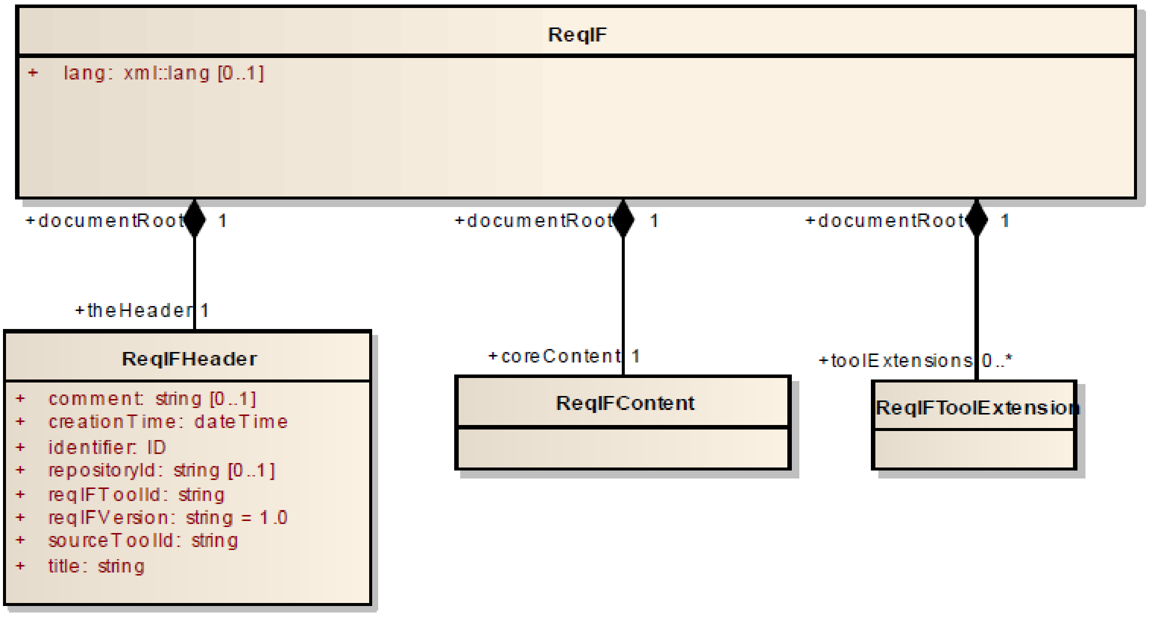
ReqIF

ReqIF (Requirements Interchange Format) is an XML file format that can be used to exchange requirements, along with its associated metadata, between software tools from different vendors.

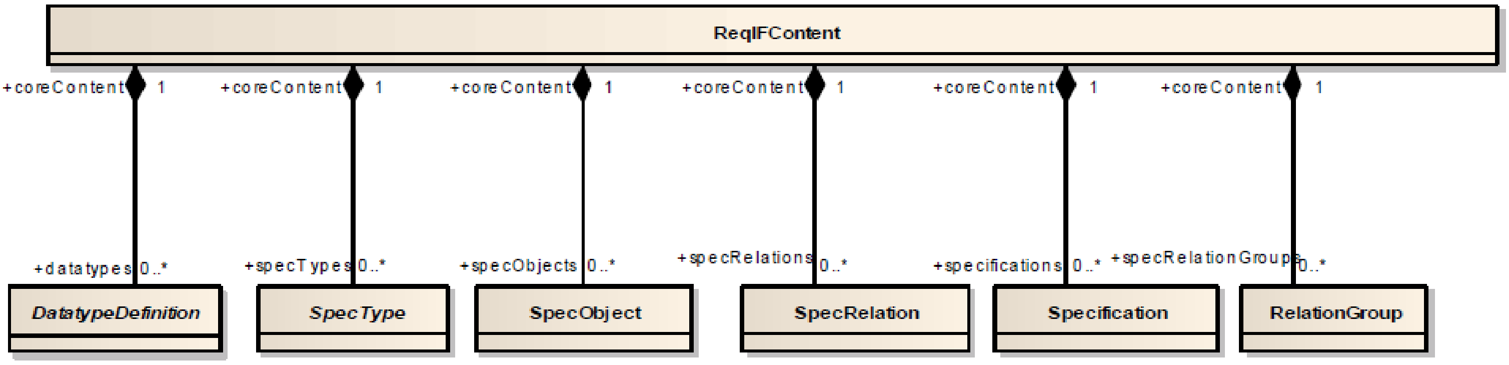
A) ***RegIF "Black box" view :***

A ReqIF document is composed of data types, and requirements types and content. Its structure is the following:



⇨ See a document structure : Structure.txt

B) ***« ReqIFContent »:***



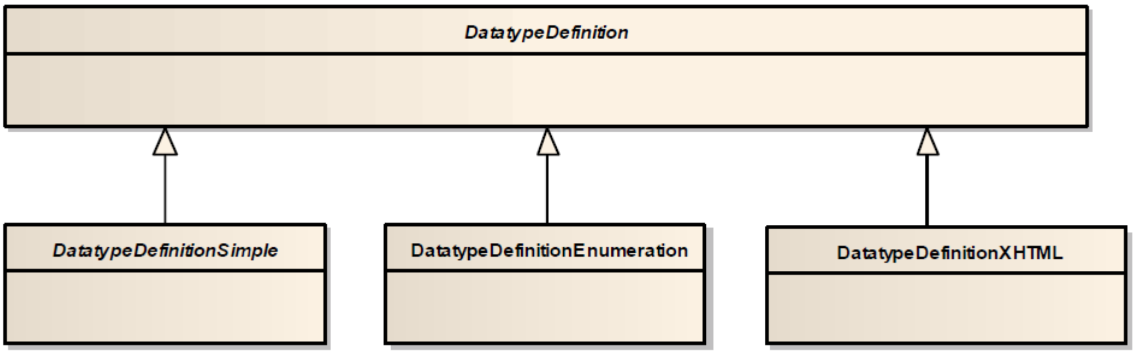
So, we have in « ReqIFContent » view four things:

1- Data types, with "DataTypeDefinition" (In file : "DATATYPES");
2- Specifications types, with "SpecType" (In file : "SPEC-TYPES");
3- Actual requirement content, with :
    a. "SpecObject" (In f. : "SPEC-OBJECTS");
    b. "SpecRelation" (In f. : "SPEC-RELATIONS");
    c. "Specification" (In f.: "SPECIFICATIONS") and "RelationGroup" (In f.: "RELATION-GROUPS").
4- And, finally, there are access restrictions, with 'AccessControlledElement'.

B-1) Data types:

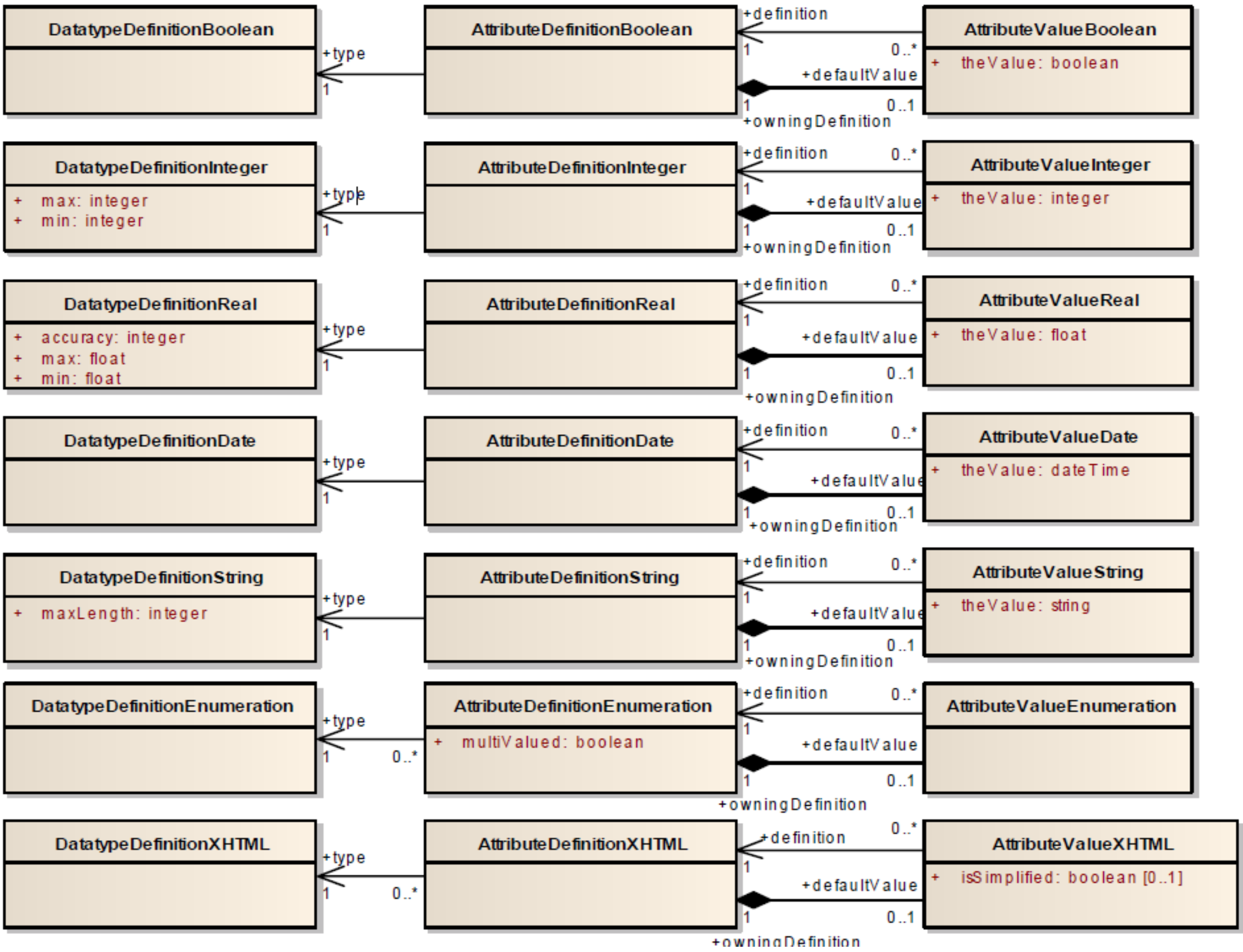
There exist three data types:

- Simple data types (Integer, Date, Boolean, Real and String);
- Enumeration;
- Formatted content (For example, referencing of external objects, like a picture.).

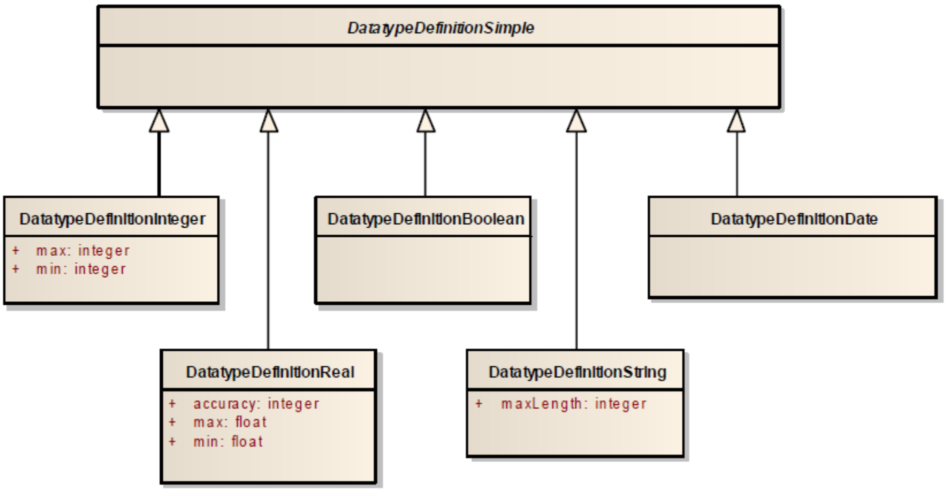


Each attribute value is related to its data type, via an attribute definition:

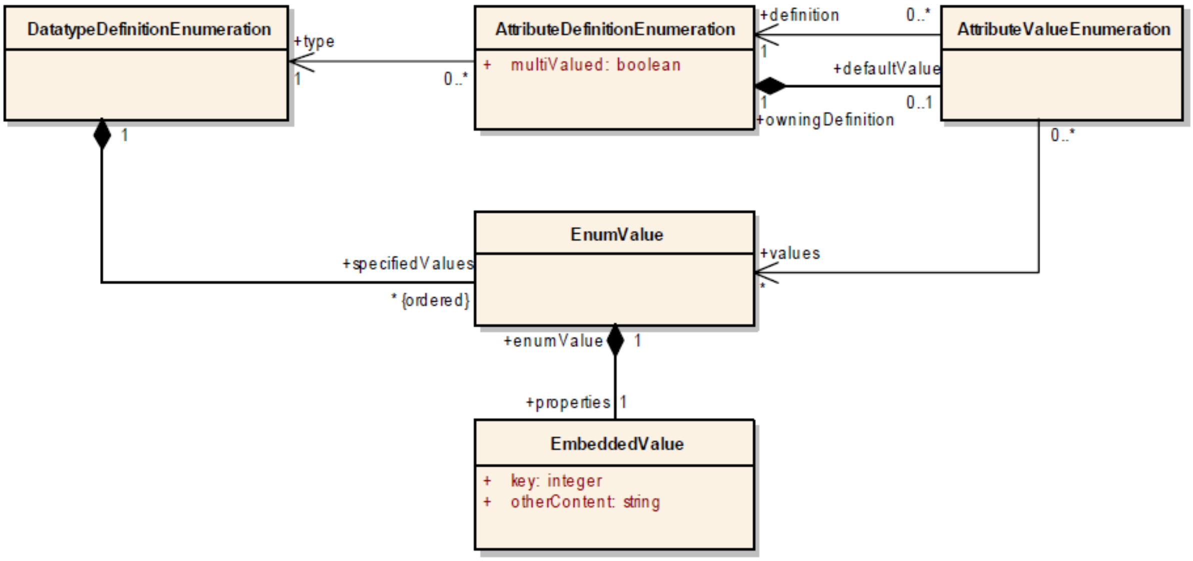
So, ‘AttributeValueX’ is related to ‘AttributeDefintionX’, and this last one is related to ‘DataTypeDefintionX’. An ‘AttributeDefinition’ element may have a default value, which is used when no corresponding ‘AttributeValue’ is supplied.



The primitive types supporting by ReqIF are the following:



Enumeration data types:

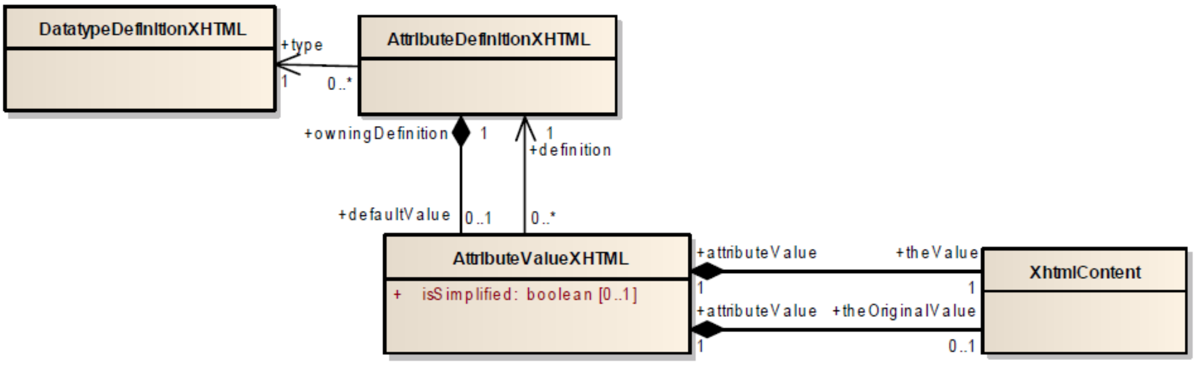


Xhtml content:

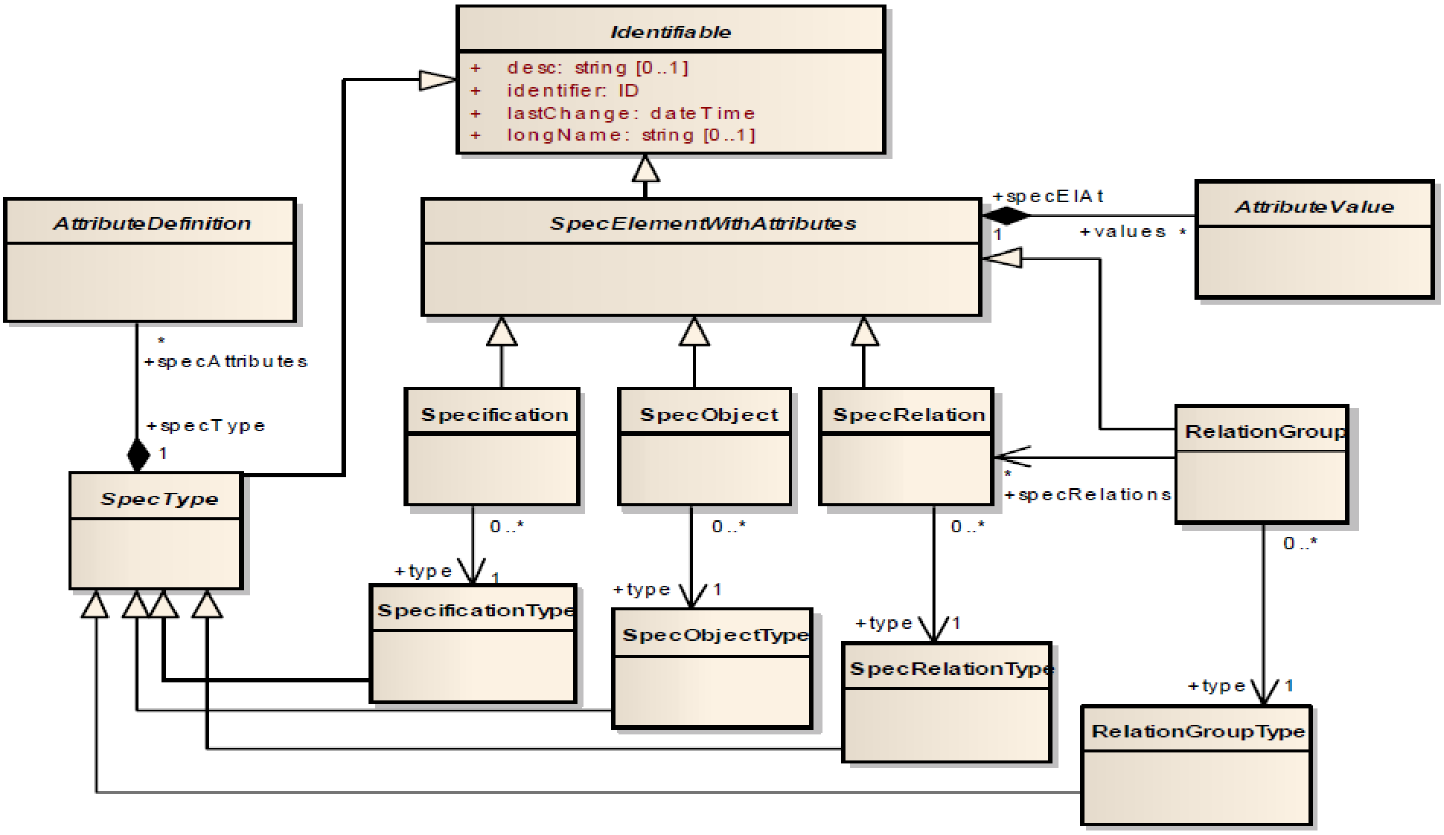
Two functionalities:

- Storing formatted text, using XHTML;
- Inclusion of external objects.

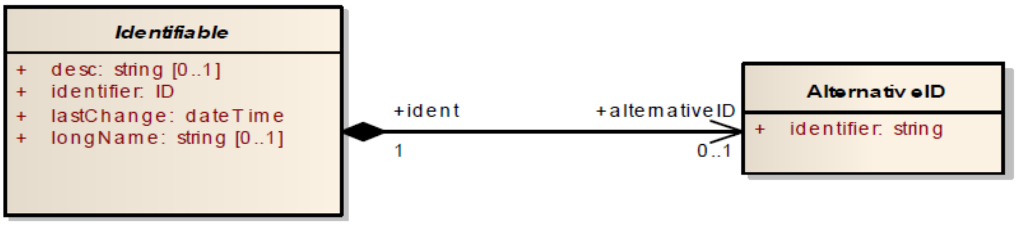
'AttributeValueXHTML' is a wrapper for XHTML document, embedded in XML document. The XHTML documents are modeled as a ReqIF element in 'XhtmlContent' (Which switches XML namespace to XHTML namespace.).



B-2) Specifications types:



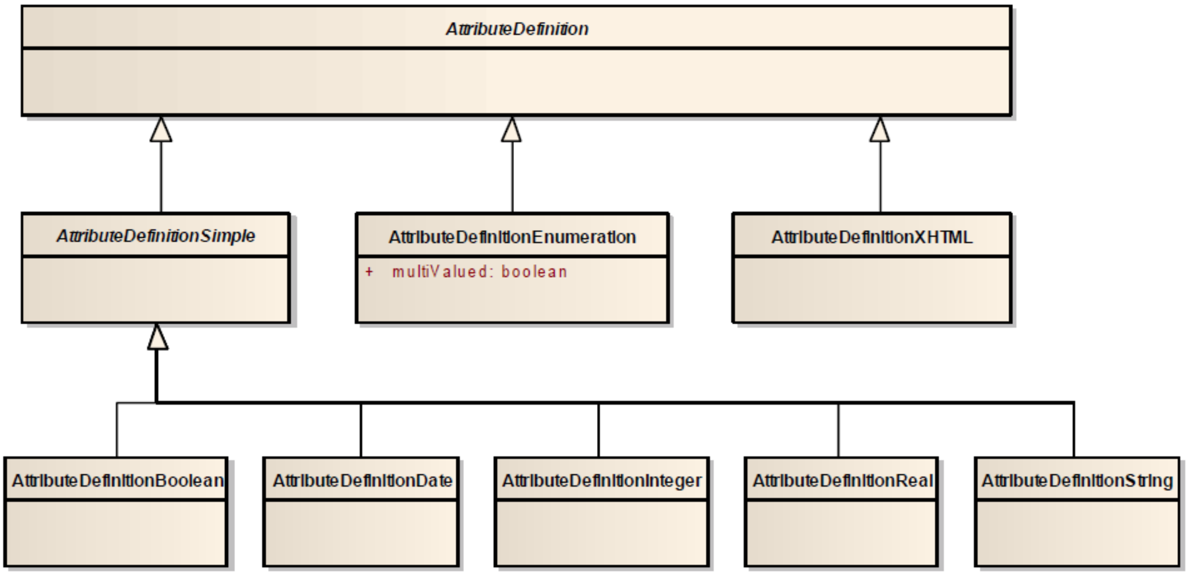
To identify elements, the class 'Identifiable' is used. But, if a tool can't handle the original I.D., then it's possible to complete it with an alternative string I.D. So, conceptually, we have this:



a. Attributes definition:

To define some attributes, there are classes inheriting of ‘AttributeDefinition’:

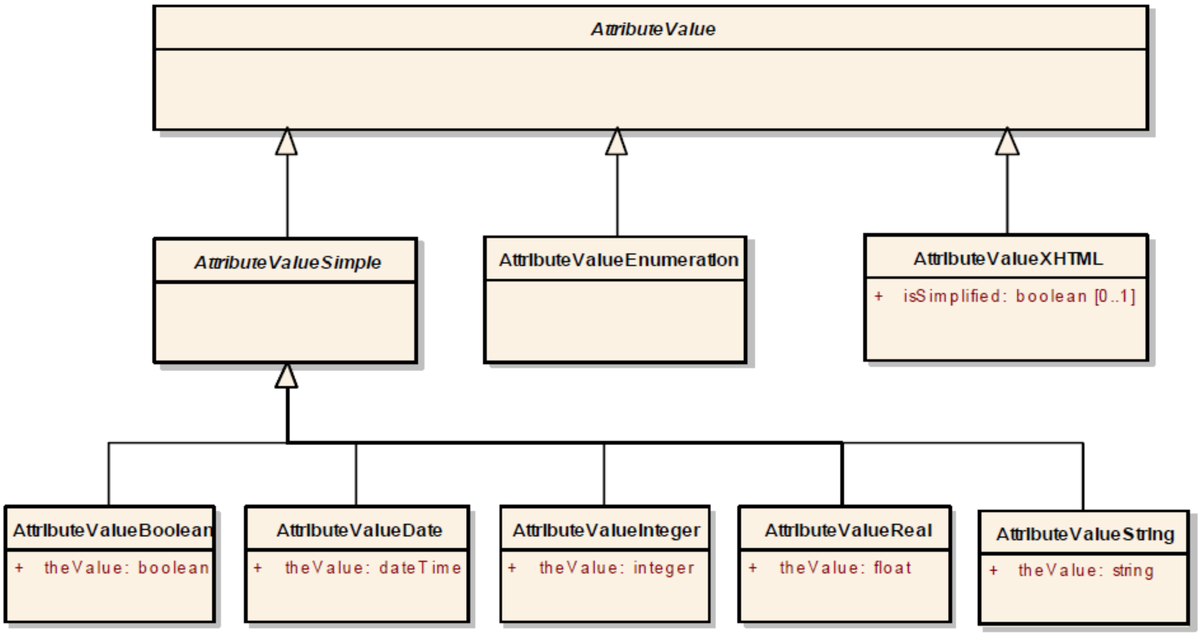
- ‘AttributeDefinitionSimple’: to define usual kind of attributes, like integer, string, etc…;
- The two others are for enumeration and xhtml type.


AttributeDefinition
AttributeDefinitionSimple
AttributeDefinitionEnumeration
+ multiValued: boolean
AttributeDefinitionXHTML
AttributeDefinitionBoolean
AttributeDefinitionDate
AttributeDefinitionInteger
AttributeDefinitionReal
AttributeDefinitionString

B-3) Actual requirement content:

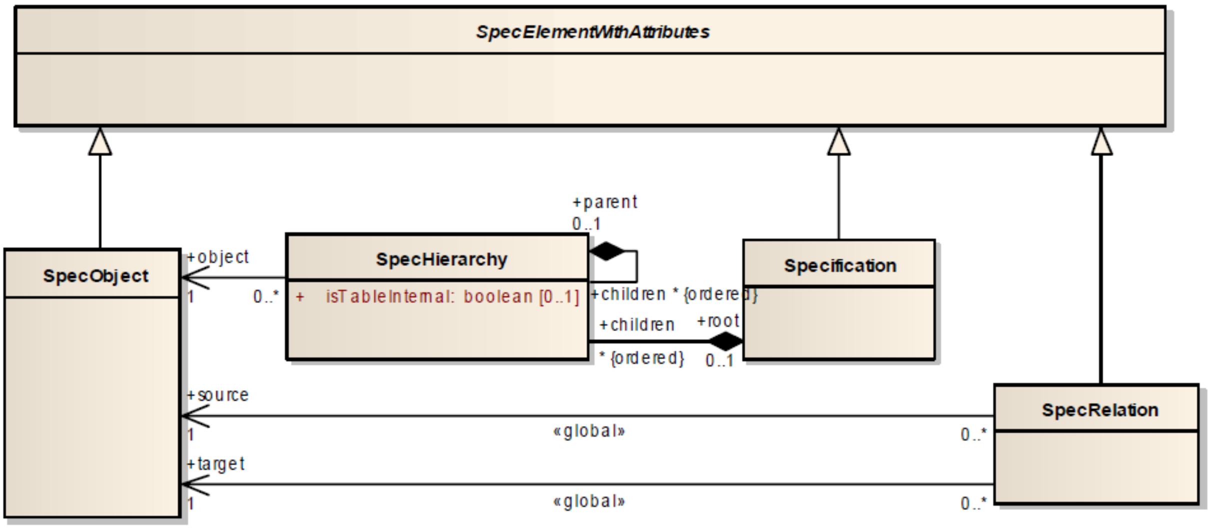
a. ‘SpecObject’ definition:

If we observe the diagram on part ‘B-2)’, we could see a class named ‘AttributeValue’: it allows defining the different values of the requirement content. For each concrete subclass of ‘AttributeDefinition’ (See part ‘B-2) a’.), one concrete type exists:



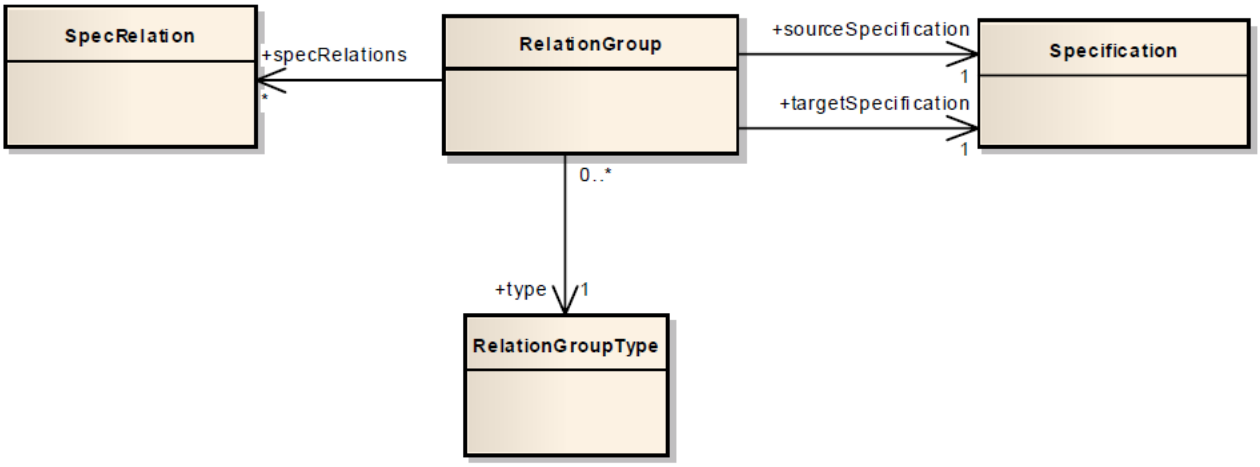
b. “SpecRelation” (In f. : “SPEC-RELATIONS”):

Two requirements may have a relation to each other. This relation is represented by an association of one ‘SpecRelation’ to a target and a source:



c. “Specification” (In f. : “SPECIFICATIONS”) and “RelationGroup” (In f. : “RELATION-GROUPS”):

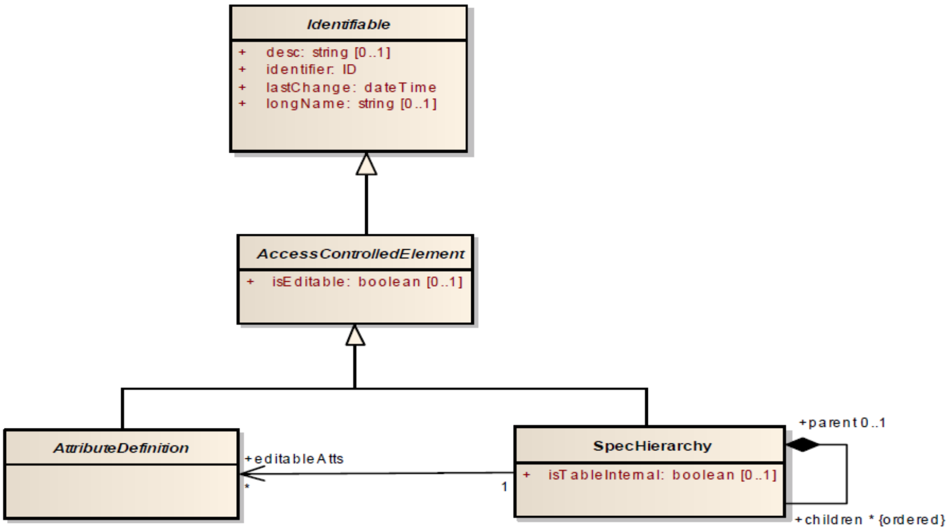
The two requirements related have their specifications related to each other, and are referred by the “sourceSpecification” and “targetSpecification” associations of a “RelationGroup” instance:



B-4) Access restrictions:

When partners have different access right, ReqIF allows specifying edition or read-only. There are three cases of restrictions:

- On specification hierarchy;
- On attributes in a spec. hierarchy;
- On attributes definitions.


Identifiable
+ desc: string [0..1]
+ identifier: ID
+ lastChange: dateTime
+ longName: string [0..1]
AccessControlledElement
+ isEditable: boolean [0..1]
AttributeDefinition
SpecHierarchy
+ isTableInternal: boolean [0..1]
+editableAtts
*
1
+parent 0..1
+children * {ordered}