Validation of Instances of Functional Views

Knutholmen Sprint | 23–27 May 2016

Validation of instances of Functional Views is crucial to maintain data integrity. Especially in case of importing instances of other sources, validation is important. Each binding has its own limitations of validation.

Validation can mean here: allowed classes, allowed properties, and allowed relationships specific to a Functional View.

### XML Schema

* Existence of used/required classes can be validated with an XML parser on the basis of an XML Schema.
* Validation of used/unused class properties has limitations. It would be necessary to refine class definitions in XML Schema for this type of validation. This would take effect on interoperability.
* Validation of internal/external relationships (from the perspective of the Functional View) has limitations. It would be necessary to refine class definitions in XML Schema for this type of validation. This would take effect on interoperability.
* Secondary-level validation (after validation with the XML parser) could be done on the basis of a formal representation of used properties and relationships of a specific Functional View. For example, related Schematron rules could be generated on this basis.

### OWL/RDF-S

The concept of validation doesn’t exist in OWL/RDF-S.

* Validation can be done with tools using validation languages like ShEx. A formal representation of used classes, used properties and relationships in a specific Functional View would be necessary. The ShEx rules could be generated on this basis.

Relational Databases

The definition of a database scheme allows the definition of various rules.

* The used classes can be defined in the database scheme.
* The definition of used properties and relationships in the database scheme would result in the definition of a variation of a class in the database scheme. This is not really desired because then there would be multiple variations of this class which is not desired for interoperability reasons. A SQL check routine could be generated on the basis of a formal representation of used properties and relationships of a specific Functional View.

### Program Languages

Validation routines can be generated on the basis of a configuration file with rules on the used properties and relationship.

### Conclusion

It looks like the most flexible and efficient approach is to make most validations regarding used classes, used properties, and relationships with an additional program which uses a configuration file. This configuration file will be anyway generated for the purpose of the documentation generation of a Functional View. The proposed YAML file for this purpose could also be used for the input of validation programs. The approach would be on a secondary level beyond the possibilities of the specification of the specific binding. Nevertheless all possibilities of the specification of the specific binding (like with XML Schema or RDBMS) should be used inside the limitation that class definitions should not be specific to functional view. It would be conceivable if the rules of the configuration file could be represented in some way in the syntax of the specific binding specification.

A general decision is necessary how internal/external references (from the perspective of a specific Functional View) and not used properties are treated. The above described approach seems to be the way to go.