SKOS Evolution Relations

# Introduction

Temporal evolution for Concepts has always been an important aspect for Authority lists (NALs) at MDR. Currently NALS are edited and managed as Excel worksheets. The first two columns express the identity of a unique record Id and an (possibly repeated) authority code. The authority code bears the identity of the described concept while the record id uniquely identifies the version of the concept description. Hence a concept may have several version descriptions.

The concept description is essentially the rest of the columns that carry various properties semi formally defined by the column name/type. Each new version of the concept is a record in the worksheet with all properties distributed linearly: properties of the concept, of the concept labels (various types), relations to successor or predecessor version of the concept (referenced by record id) and relations to concepts or concept versions in other tables.

The challenge now is to express the NALs in RDF form by adhering to the SKOS principles and definitions. Ideally the semantics of the resulting KOS should allow for the same level of expressivity maintain and detail as in the Excel worksheets.

One successful attempt to achieve this was expression as simple SKOS. The resulting file expresses the latest version for each concept and the set of afferent labels in various languages. However this variant does not permit:

1. expressing additional specifications for labels (such as context, script, creation, IMMC approval, start use dates etc.)
2. expressing relations for other tables such as location, label type, etc.
3. expressing various concept versions

To amend the above constraints has been created an application profile (AP) titled SKOS-AP. It is based on SKOS-XL which (among others) defines a new class: Label to allow for reified label relations. The SKOS-AP also adds various (relevant) properties for the concept and label and for the additional two classes XlNote and XlNotation.

SKOS-AP enables expressing label properties, relations to other tables and also pseudo-versions of the concept. The versioning of the concept can be expressed through instances of Label, XlNote and XlNotation classes each having assigned a version number and/or start and end use dates.

The pseudo-versioning approach proposed in SKOS-AP works for relatively simple NALs. It works for NALs that focus mainly on label definitions. However not all the NALs are simple records of concept labels. Many of them (Countries, Corporate Bodies, Places etc.) make multiple references to concepts in other tables. Moreover the references to concepts in other tables may differ by concept version and/or may make a reference to a specific version of the foreign concept.

Expressing the relation between a concept version and a version of a foreign concept is not possible in SKOS-AP as it lacks the distinction between a concept (in abstract sense) and a specific version of that concept (in specific sense or as an instance of the abstract concept). This limitation of SKOS-AP in expressing concept versioning constitutes the main focus of the current paper.

Evolutionary relations in SKOS are not a new topic. A set of important discussions and proposals is listed on W3C SKOS-Issues page[[1]](#footnote-1).

Below I present a set of requirements, point out to mentions of best practices and a potential solution.

# The problem

# Requirements

* Express evolutionary relations among concepts.
* Concept versions may be in various temporal relations, beyond the intuitive successor relation i.e. concomitant, distanced, overlapped, strict successor relations etc. Such relations are defined by Allen Interval algebra.
* Relations differ in scope: some strictly concern concepts (regarding their coordination/compositions/decomposition) and some concern temporal relations for evolution (and not only). This is documented by Baker et al (Baker, et al. 2013)
* The model may adopt (a) snapshot approach where all properties are specified for a particular time point or interval or (b) delta approach where only properties with changed/new values are specified
* Maintain a stable Concept URI while still be able to refer to a specific version of the concept.
* The solution should be SKOS, SKOS-XL, SKOS-AP backward compatible, i.e. maintain the semantics of the legacy models.

# Best Practices

* We recommend that you **do not** change the URI for a concept with each scheme version. The more stable your concept URIs are, the more consistently they will be applied.[[2]](#footnote-2)
* owl:priorVersion
* owl:versionInfo
* owl:versionIRI
* dcterms:hasVersion
* skos:historyNote
* dcterms:isReplacedBy / dcterms:replaces

# Current SKOS-AP-EU: evolution as reified properties

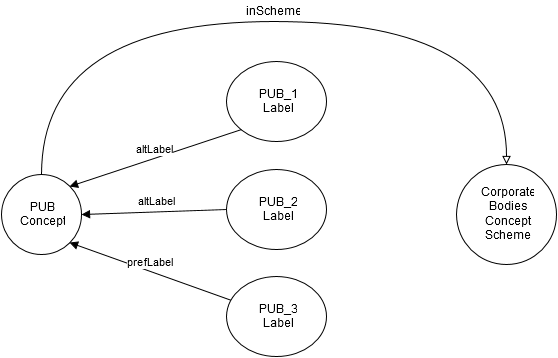


Figure Representation of concept Pseudo-Evolution via reified label property

Currently the evolution of skos Concepts can be captured only as changes in label, note or notation. These properties have been transformed into classes skosxl:Label, euvoc:XlNote and euvoc:XlNotation and allow euvoc:startDate and euvoc:endDate for temporal validity constraints.

The advantage of reifying label relations is that skos-xl:Label instances can carry additional information such as label version, start/end dates for the label validity, label status (whether it is in in use or deprecated), label source etc.

Additionally with a minor modifications of SKOS-AP-EU specification (v29 from Sep 2015) two more properties can be used for this purpose: dct:replaces and dct:isReplacedBy. At the moment their domain is only skos:Concept, but the domain constraints can be relaxed to include every class acting as a reification.

If the structure of the MDR data would be strictly concerned with terminological relations, such a model would perfectly suffice such as in the case of EuroVoc. However the MDR tables vary in their complexity some of which require table specific (data and object) properties. Moreover MDR is expected to create an open number of NALs complexity of which is unknown at the moment.

Given the context of MDR authority tables, current approach has several limitations:

* No abstraction: mix of class and instance levels
  + There is no distinction between concept and the concept snapshot/instance. Such a definition would clearly separate transient (subject to change in time) from intransient properties (time invariant). Where transient properties are only ascribed to concept instances/versions and the intransient ones only to the concepts.
* Lack of scalability
  + Authority Lists (NALs) may be simple (i.e. when the SKOS-XL suffices to express the table logics, such as ConeptStatus, Usecontext etc.) and complex (i.e. when the table has extra properties that are not covered by SKOS-XL and required definition of table specific ones, such as CorporateBodies, Countries, etc. ).
  + In the case of complex NALs, some (table specific) properties are transient meaning that to ensure their versioning with current mechanism, they would have to be reified (transformed into classes) leading to model updates every time there is a change in the model.
* Potential source of errors and difficulty to maintain on the long run
  + Moreover these reified properties are not ontologically established classes but rather technical a work around. This inevitably leads to a large and potentially messy model where business logic is mixed with technical wrappers making it costly and difficult to maintain on the long run; not to mention that it is an additional source of errors and confusion.
  + The concept will have all versions of a label (e.g altLabel-short, long, etc.) and other information attached to the concept. Without a proper editor would, this will become difficult to maintain. At the moment, it is clean and clear because the editing is done in Excel where each new row is a new version of the concept. However in VocBench things this might get less clear especially when the number of versions and the number of new information (other labels and extra properties) per concept-version will grow.

# Addressing the limitations and requirements

## Distinguishing Abstract Concept and Concept Instance

The current solution consists of introducing a distinction between the Abstract concept and the Instance or Snapshot of a Concept. This idea is well documented by Tennis & Sutton (Tennis and Sutton 2008).

The Proposed solution is depicted in Figure 2. It is a simplified representation for PUB authority code from the Corporate Bodies NAL. On the left side there is a generic PUB concept whose identity is fixed and considered invariant across time. To it, there are three Concept instances, each corresponding to a concept version with specific preferred and alternative labels (including short, long and other labels).

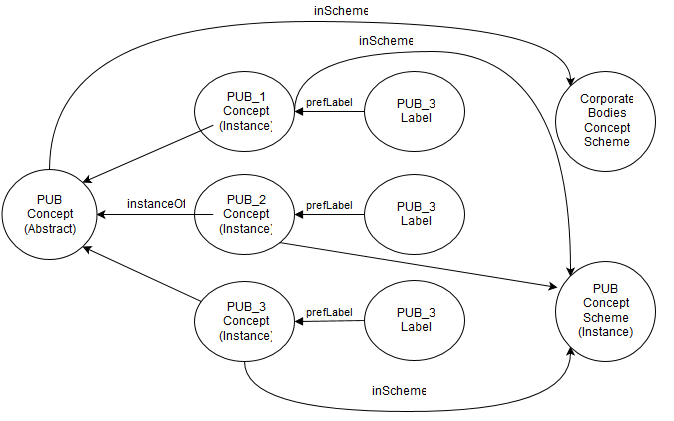


Figure Relationship between abstract concept, concept instance and concept scheme in which the instances are declared

The notion of concept instance can be made explicit in the context of following core assertions.

* **Core Assertion 1:** A **Concept** is an "abstract idea or notion; a unit of though"[[3]](#footnote-3) identified by URI.
* **Core Assertion 2:** A **Concept Instance** is a concrete manifestation of a concept within a scheme and is identified by URI;
* **Core Assertion 3:** A **Scheme** is a collection of concept instances and is identified by URI.
* **Core Assertion 4:** A Scheme may embody more than one concept instance of the same concept (e.g. a historical sequence of instances reflecting change states)
* **Core Assertion 5:** A **Scheme Snapshot** is a point in time image of the state of the scheme concepts, relationships and documentation.

## Distinguishing Schemes for Concepts and Schemes for instances

Since we separate the types of concepts into two categories based on their level of abstraction, it is instrumental to use the same distinction for the concept schemes as well. Since many schemes at the moment have the purpose of grouping (abstract) concepts under one umbrella, it is only useful to distinguish another type of grouping for specific version(s) of concepts.

PLACE AN EXAMPLE

## Backward compatibility with pseudo-evolution approach via reified properties

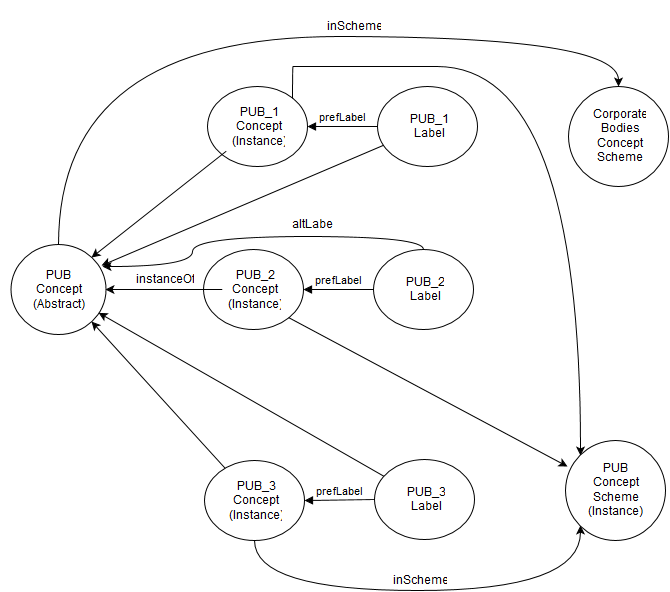
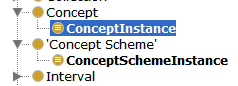
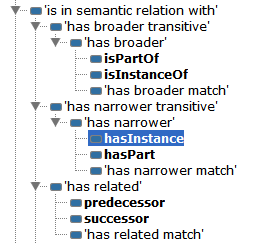
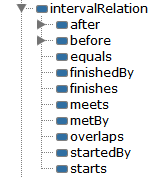


Figure Backwards Compatibility

## Distinguishing the scope of relations: meaning and time







# Conclusion

# Bibliography

Baker, Thomas, Sean Bechhofer, Antoine Isaac, Alistar Miles, og Guus Schreber. «Key choices in the design of Simple Knowledge Organization System (SKOS).» *Web Wemantics: Science, services and Agents on the world Wide Web*, 2013.

Tennis, Joseph, og Stuart Sutton. «Extending the simple Knowledge Organization System for Concept Management in vocabulary Development Applications.» *Journal of the American society for information science and technology*, 2008: 25-37.

1. https://www.w3.org/2001/sw/wiki/SKOS/Issues/ConceptEvolution [↑](#footnote-ref-1)
2. http://www.w3.org/wiki/SkosCoreGuideToc/SectionVersioning [↑](#footnote-ref-2)
3. http://www.w3.org/TR/swbp-skos-core-spec [↑](#footnote-ref-3)