Requirements for genericode 0.4

Anthony B. Coates, abcoates@mileywatts.com

18th September, 2006

Table of Contents

1.Introduction....2
2.Current Requirements....2
R1 genericode should be a pure code list representation that is not tied to any particular code list validation process or software....2
R2 The code list metamodel should be expressed as a UML logical model, not just as a physical model (e.g. an XML schema)....2
R3 The code list format should support complex code list definitions, but it should not be complex to define simple code lists....2
R4 Support for multiple, alternative codes....2
R5 No particular choice of code is the preferred choice....2
R6 Codes are part of the metadata for the code list....3
R7 Only unique metadata can be used for codes....3
R8 Every code list must have at least one key....3
R9 Code list metadata does not need to be unique....3
R10 Unique metadata does not have to be defined as an alternative code....3
R11 The order of codes in a code list is not important....3
R12 The order of metadata in a code list is not important....3
R13 Metadata can be simple or complex in structure....3
R14 Metadata can have a particular simple type....3
R15 Metadata can be left undefined....4
R16 A code list can be derived from an existing code list by adding/removing rows/columns/keys....4
R17 A code list can be derived from existing code lists by aggregating their rows....4
R18 A code list can be derived from existing code lists by aggregating their columns....4
R19 A code list can be derived from existing code lists by removing all common rows before aggregating the remaining rows....4
R20 A code list can be derived from existing code lists by removing all common columns before aggregating the remaining columns....4
R21 A derived code list can be required to contain a source code list as a row-wise subset....4
R22 A derived code list can be required to contain a source code list as a column-wise subset....4
R23 The basic code list operations can be composed arbitrarily and to any depth to create a derived code list from a set of source code lists....5
R24 The operations used to derive particular code lists must be unambiguously encodable so that they are repeatable and auditable....5
R25 Column sets can be represented....5
R26 Column sets can contain keys....5
R27 Code lists can use columns and keys defined in other code lists or in column sets....5
R28 Metadata-only code lists can be represented....5
R29 Each code list has a unique identifier, independent of its individual versions....5
R30 Each code list version has a unique identifier, different to the version-independent identifier for the code list....5
R31 Each column or key in a code list or column set can have a unique identifier....5
R32 Location URIs are distinct from identification URIs....5
R33 Sets of code list versions can be represented....6

R34 Documentation and annotations can be applied to definitions.....6
R35 Documentation has a language identifier, and there can be documentation in multiple languages.....6
R36 Short and long names are supported.....6
3.Future Requirements.....6
F1 It should be possible to represent code lists that cannot be enumerated.....6
F2 Support for multiple alternate code values for the same code list entry.....6
F3 Start/expiry dates/times for code lists, code list sets, and individual codes.....6

1.Introduction

This document contains the current and future (outstanding) requirements for genericode 0.4. It is presented as a basis for the requirements for the OASIS Code List Representation TC. A key principle for accepting requirements for genericode has been that new requirements should not have a significant adverse impact on the implementation (and quality thereof) of existing requirements. In particular, some potential future requirements that would be relatively straightforward to implement for explicitly defined code lists have not yet been accepted because of the difficulty in correctly integrating them into genericode's support for derived code lists.

genericode has a tabular model for code lists. genericode **"rows"** are individual entries in a code list, where an entry is a set of one or more codes, plus other metadata, that is associated with a single conceptual entry in the code list. genericode **"columns"** are individual (typed) pieces of metadata that can be applied to each entry in a code list. So columns define what kind of data can be in the code list, while rows define what actual data is in the code list.

genericode also has the concept of **"keys"**, where a key is a set of one or more columns which uniquely identifies each row in the code list. Where a key has more than one column, it is a compound key.

Note: it is possible that some requirements have been missed from this document. If so, they will be added to a later version as appropriate.

2.Current Requirements

R1 genericode should be a pure code list representation that is not tied to any particular code list validation process or software

R2 The code list metamodel should be expressed as a UML logical model, not just as a physical model (e.g. an XML schema)

R3 The code list format should support complex code list definitions, but it should not be complex to define simple code lists

R4 Support for multiple, alternative codes

It must be possible to have multiple, alternative codes for the same code list.

R5 No particular choice of code is the preferred choice

Where a code list has multiple, alternative codes, there must be no "preferred" choice of code. The choice of code is a "late binding" decision that is made by users of the code list, not by publishers of the code list.

R6 Codes are part of the metadata for the code list

The codes in a code list (whether multiple or not) are part of the metadata for the entries in the code list. They may be used as codes in some contexts, but they be used as non-code metadata in other contexts.

R7 Only unique metadata can be used for codes

Columns can be part of a key (a "code" in common parlance) only if the entries (rows) in those columns are unique, i.e. no two rows in the code list have the same key value(s).

R8 Every code list must have at least one key

A code list must have at least one key, since the keys are the "codes" in common parlance.

R9 Code list metadata does not need to be unique

Columns are not required to contain unique metadata values unless they are used in a key. In a compound key, individual columns can contain non-unique metadata values, so long as the compound key value is unique.

R10 Unique metadata does not have to be defined as an alternative code

Metadata columns (or sets thereof) with unique values do not have to be defined as keys for a code list. For example, some columns may contain "gratuitously unique" data – data that is currently unique, but which is not guaranteed to be unique over the life of the code list (and its versions).

R11 The order of codes in a code list is not important

The order of codes (rows) in a code list (in an XML file or other ordered representation) should not be used to convey any meaning (semantic information). The code list metadata itself (any set of columns) should be used to define any ordering that is appropriate, in a way that is independent of the ordering in any particular code representation.

R12 The order of metadata in a code list is not important

The order of metadata (columns) in a code list should not be used to convey any meaning. Column identifiers and/or column metadata (which applies to the column, not to any of the rows) should be used to identify columns.

R13 Metadata can be simple or complex in structure

Metadata columns can contain "simple" data values (in the sense of schema simple types), or they can contain "complex" data values (XML fragments).

R14 Metadata can have a particular simple type

Metadata columns can have a defined data type. The W3C XML Schema simple types are the default set, but a different datatype library can be specified (through the use of an identifying URI).

Facets can be defined to restrict the data type (e.g. length, minimum/maximum, or pattern restrictions).

Note: there is an option question around whether it should be possible to define the "type" of complex data (XML fragments) in some sense.

R15 Metadata can be left undefined

Particular metadata columns can be defined as "nillable", meaning that they can contain undefined (nil) values. Nillable columns cannot be used as part of a key.

R16 A code list can be derived from an existing code list by adding/removing rows/columns/keys

New rows cannot be added if they violate the uniqueness of the keys. Columns cannot be removed if they are used as part of a key, and if a key is removed, at least one other key must remain.

R17 A code list can be derived from existing code lists by aggregating their rows

Aggregation of rows can only occur if the source code lists have the same columns, or if the columns which are not common to all source code lists are allowed nillable columns.

R18 A code list can be derived from existing code lists by aggregating their columns

Aggregation of columns can only occur if the source code lists have at least one key in common (which also implies they have one or more columns in common). Where any of the source code lists have the same column, the values in that column must be the same in each source code list (which means that the values for any common key must be the same across the source code lists).

R19 A code list can be derived from existing code lists by removing all common rows before aggregating the remaining rows

R20 A code list can be derived from existing code lists by removing all common columns before aggregating the remaining columns

It should also be possible to remove common keys as well as common columns. A column cannot be removed unless all keys that it is part of are removed, and there aggregate must contain at least one key.

Note: support for keys as well is columns is not yet implemented.

R21 A derived code list can be required to contain a source code list as a row-wise subset

Note: there is an open question about whether it should be possible to specify that only keys are compared, or that only particular keys are compared.

R22 A derived code list can be required to contain a source code list as a column-wise subset

It should be possible to specify the subset via keys as well as via columns.

Note: support for keys as well is columns is not yet implemented.

R23 The basic code list operations can be composed arbitrarily and to any depth to create a derived code list from a set of source code lists

R24 The operations used to derive particular code lists must be unambiguously encodable so that they are repeatable and auditable

R25 Column sets can be represented

Sets of code list columns can be defined independently of any particular code lists, and any number of the columns from a column set can be used in the definition of a code list or another column set.

R26 Column sets can contain keys

Column sets can contain keys which can be used in the definition of a code list or another column set. Where a key from a column set is used in a code list or another column set, all of the columns in that key must also be used.

R27 Code lists can use columns and keys defined in other code lists or in column sets

Where a key from a code list is used in another code list, all of the columns in that key must also be used.

References to externally defined columns or keys must use the unique identifier for the column/key, and may optionally include one or more location URIs for the column set or code list in which the column/key is defined.

R28 Metadata-only code lists can be represented

Code lists which contain metadata only (no rows) can be represented. These are code lists for which the row data is not published. The representation must be different from that of code lists which are empty (zero rows), but which nonetheless contain a complete publication of the data in the code list.

R29 Each code list has a unique identifier, independent of its individual versions

R30 Each code list version has a unique identifier, different to the version-independent identifier for the code list

The code list definition contains the version number (or string) as well as the unique identifier (a URI).

Note: there is an open question around whether metadata-only code lists should be required to have explicit versions, especially for code lists where it is unlikely that the columns and keys will change over time.

R31 Each column or key in a code list or column set can have a unique identifier

R32 Location URIs are distinct from identification URIs

It should not be assumed that URIs used for identification correspond to any retrievable asset (even

if the URI is a URL). There must be explicit support for (multiple) location URIs for retrievable representations of code list sets, column sets, or code lists.

R33 Sets of code list versions can be represented

It must be possible to specify a "configuration" of versions of code lists that together form a coherent set for some purpose.

Note: there is open question around whether it should it be possible to include a version-independent reference to a code list, instead of a reference to a particular version of a code list.

R34 Documentation and annotations can be applied to definitions

Documentation and arbitrary annotation data (in the sense of W3C XML Schema) must be able to be specified for code list sets, column sets, code lists, columns, keys, rows, and individual values in rows.

R35 Documentation has a language identifier, and there can be documentation in multiple languages

R36 Short and long names are supported

A code list set, column set, code list, column, or key must have a short name (token name) which can be used for naming software artefacts. It can also have any number of long names. These can be in different languages. Where necessary, a long name can have an "identifier" attribute, which is a string that sufficiently identifies a particular long name from a set of long names.

Columns and keys must have short names that are unique within the column set or code list in which they are defined.

3.Future Requirements

F1 It should be possible to represent code lists that cannot be enumerated

Some code lists cannot be explicitly enumerated, e.g. because they are too large to practically enumerate, or because they are proprietary (so they can be matched against, but not made available in their entirety).

F2 Support for multiple alternate code values for the same code list entry

The difficulty with this requirement is how to specify the addition and/or removal of alternate code (or metadata) values in derived code lists, without undue complexity.

F3 Start/expiry dates/times for code lists, code list sets, and individual codes

The difficulty with this requirement is how to specify the addition, removal, or modification of start/expiry dates/times in derived code lists, without undue complexity.