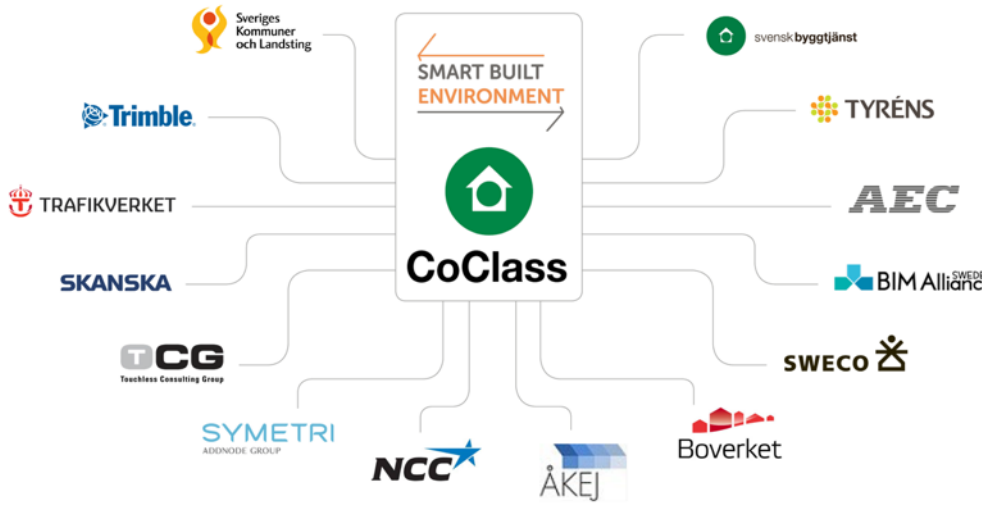
Rapport enl. Formas nr

INDUSTRY PRACTICES FOR APPLICATION OF COCLASS IN SOFTWARE

SMART BUILT ENVIRONMENT



Med stöd från:

VINNOVA

Energimyndigheten

formas

STRATEGISKA INNOVATIONS-PROGRAM

Table of contents

1 Reading notes

This Industry practice is intended as a starting point for organizations intending to implement CoClass in their respective software and processes. Readers can range from developers of the built environment, planners, consultants, and software developers responsible for implementing CoClass in their software.

For information about the project, read the project report in *Appendix 1 – Report, Industry practices for application of CoClass in software*.
For information on the Industry practice and how to implement CoClass in software, read this document. For information on Svensk Byggtjänst's products CoClass Studio, the CoClass API and how to utilize them, read *Appendix 2 – CoClass Studio and API*.
For a list of definitions used in these documents, read *Appendix 3 – Definitions*.

2 Identified needs

Based on several workshops and interviews with identified stakeholders, a number of needs and requirements were identified.
The identified needs can be categorized in the following:

- Needs related to objects
- Structured requirements
- Create, change and search information

For a detailed report on the identified needs, refer to *Project Industry practices for application of CoClass in software - AP1, 2019*.

2.1 Needs related to objects

In digital representations of the world, objects of interest can be stored in multiple forms: as geometry in CAD files, objects in files based on standards such as IFC, as rows in a spreadsheet file, or as records in databases, just to mention the most common forms. An "object" as discussed here is the complete collection of digital data describing a real-world object of some sort: a building, a road, a space, a construction element and so on. A popular way of describing this collection of data is to call it a "digital twin".

Depending on the user's requirements, there is a strong need to connect several types of information to objects. Classification according to CoClass is only one of these.
Data can be packaged for different purposes into "property sets" with values to objects and may be tracked over time. Beyond this, there is also a need to connect indirect types of information to objects: documentation, images, drawings, software etc. One option of doing this is by adding properties to the object, referring to other data sources.
As a link between different data sources describing an object, a reference designation can be used.

2.2 Structured requirements

The objects need to be connected together in a formalized structure that is possible to communicate, between phases of the building process, between different stakeholders and between different software applications.
There needs to be an unambiguous understanding of the different parts of these structures, and a universal way to communicate this between all participants in a project.
There is a strong need to streamline and unify the objects and information types that are part of the building process.

2.3 Create, change and search information

Based on this there is a need to, in a united way, be able to create, change and search several types of information. They can be summarized in the following categories:

- Create
- Store
- Update
- Validate
- Search
- Calculate
- Combine
- Analyze
- Visualize
- Publish
- Merge

Many of these categories are dependent on version management of the information related to applied CoClass classification.

3 Identified requirements

The figure below illustrates the role and aim of the requirements specification.

- **CoClass**, i.e. the actual content and structure of CoClass that needs to be accessed by users and applications
- The **Web application and Web services (API)** offered by Svensk Byggtjänst for the distribution of the content of CoClass
- The **Industry practices** which is a deliverable from this project
- The **Applications** which offer functionality for end users during various stages of the life cycle. The requirements are collected by the participants of this project but should be expressed in a generic way to be applicable for similar applications covering similar needs at similar stages of the life cycle

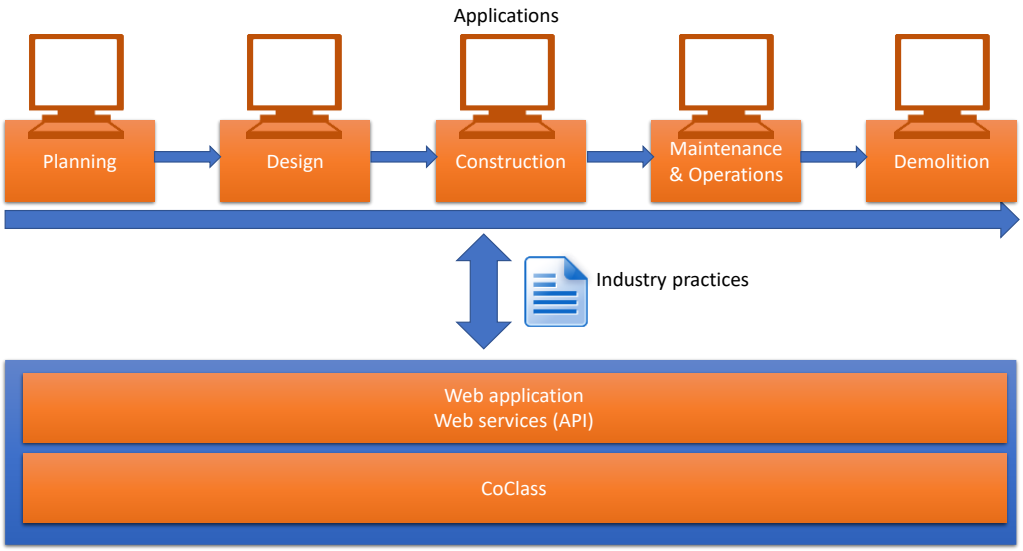


Figure 1 – A possible way of interacting with CoClass through the CoClass API

As mentioned, the actual set of requirements are listed separately. Even though some duplicate requirements have been removed, there might still be duplicate requirements which needs to be evaluated and resolved. However, due to time restrictions, no combined prioritization and complete resolution of potential duplicates has been performed. It is therefore recommended that any future processing of these requirements considers this. It is also recommended that certain guidelines are observed when prioritizing, within

subsequent work packages of this project or beyond. The following criteria should result in higher prioritization:

- A requirement affects many applications or actors
- A requirement has a strong component of interoperability/communication between applications and therefore is interface-related.

Furthermore, all requirements under the category "CoClass development" are forwarded to the CoClass maintenance process to be handled there, since this project does not handle CoClass development.

For a detailed report on the identified requirements, refer to *Project Industry practices for application of CoClass in software – AP2, 2019.*

4 Information model

In order to have a seamless experience with CoClass in multiple, different, software environments, it is important to agree upon a common information model, with clear rules and guidelines as how to classify information. This section depicts the agreed upon guidelines that the project has worked out.

4.1 Ontology and taxonomy

Taxonomy is the practice and science of classification. Typically, a taxonomy organizes concepts within a subject area into a hierarchical structure of sets and subsets based on the essential properties defining a set and separating a set from the other sets. Maybe the most widely known example is the Linnaean taxonomy set up by Carl Linnaeus in his Systema Naturae.
An *ontology* is a set of concepts and categories in a subject area or domain that shows their properties and the relations between them (Oxford Dictionary). A *taxonomy* may be viewed as a specific kind of ontology only dealing with sets and subsets. Other kinds of ontologies are *meronomies* (dealing with classification of the parts of a whole), and *general ontologies* dealing with a set of concepts and categories in a subject area or domain that shows their properties and the relations between them where relationships may be of any kind and not only according to taxonomies, meronomies.

The CoClass classification system is essentially a taxonomy of classes for the built environment. However, since CoClass builds on the ISO 12006-2 standard, a meronomy is also an essential characteristic of the system through the whole-part relationships between Construction complexes,

Construction entities and Construction elements. Construction complexes may be decomposed into Construction entities. Construction entities may be decomposed into Construction elements. Furthermore, complex Construction elements – systems – may be decomposed into their components.

4.2 Information included

Based on the information available in the CoClass system, the following information aspects are covered:

- Classification information
- Composition structures
- Reference designations
- Property types

4.2.1 Classification information

The joint information model is based on the CoClass classification system. The relationship between the classes is described by ISO 12006-2.

The schema below describes ISO 12006-2, which comprises the whole life cycle. The upper part is about the physical results. The part below describes the process, it´s resources with products and goods, aids, actors and information. Here we can also see the lifecycle processes: pre-design, design, production, and maintenance. (Bold lines describe a “type-of” relationship, thin lines describe other types of relationships.)

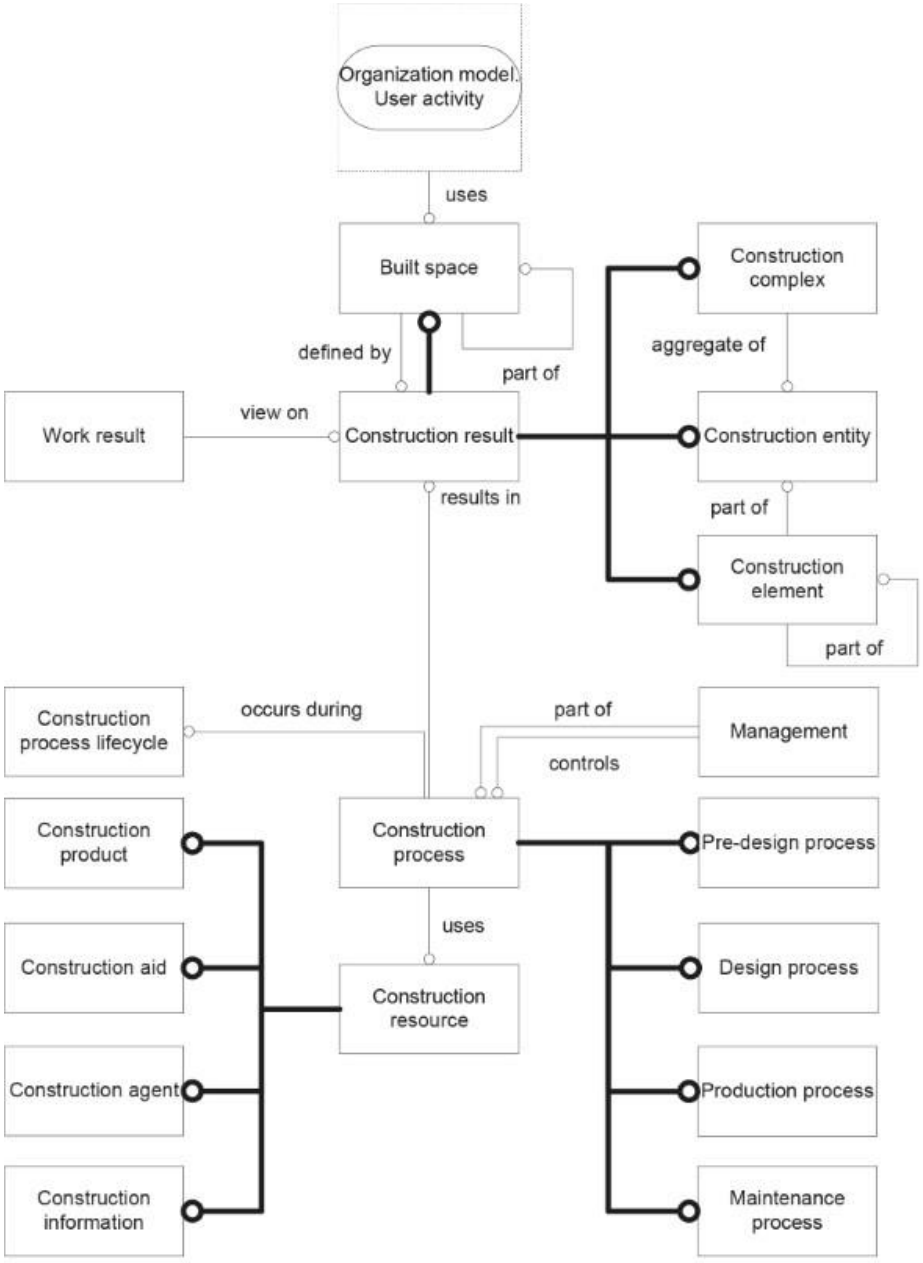


Figure 2 – description of the ISO 12006-2 schema

The information model can be simplified into a triangle, described below:

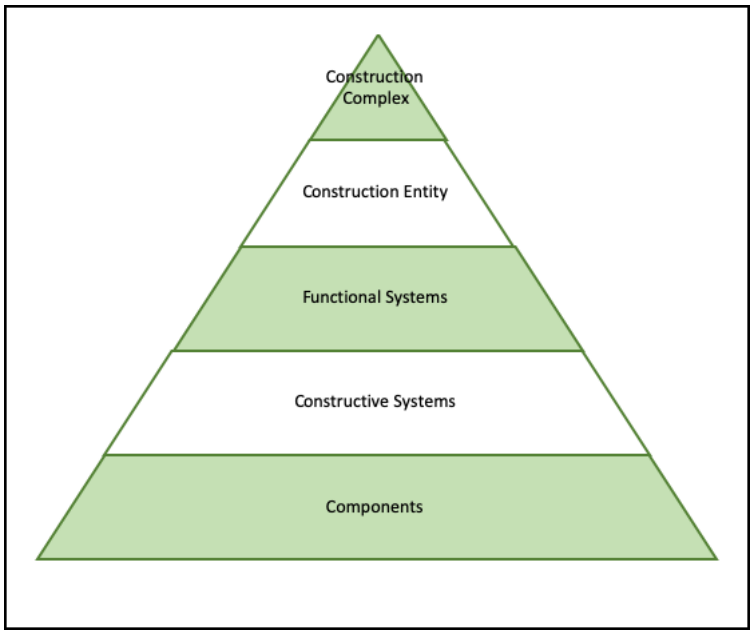


Figure 3 – The CoClass hierarchy for the built environment perspective

The fundamental information needed to describe an object is the CoClass class, defining its inherent function as a Construction complex, a Construction entity or a Construction element. The class code can thus be viewed as the key to the function of the object.

4.2.2 Composition structures

Based on CoClass classes, properties and activities, structures describing a complex object such as a Construction entity can be constructed. These are built up as a hierarchy, where one object can contain other objects in a part-of relationship, be associated with activities and characterized by properties. An object can have many children, but only one parent.

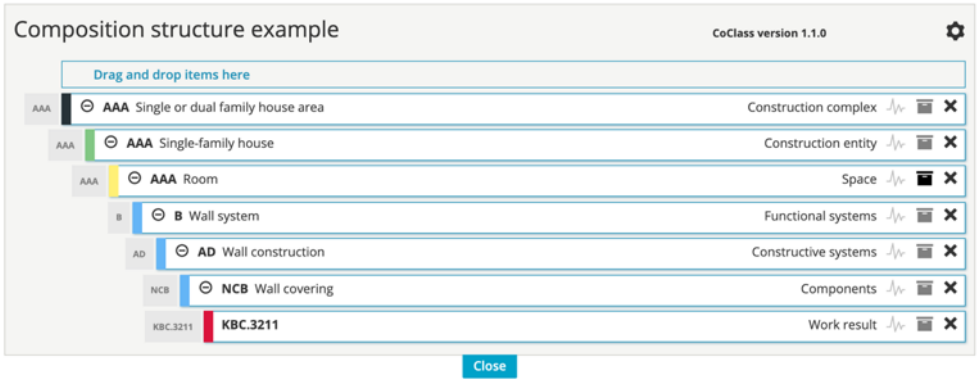


Figure 4 – Example of a Composition structure built in CoClass Studio, For details, see Appendix 2 – CoClass Studio and API

An object can include properties and activities. This Industry practice covers Classes, Properties, Activities and their values. The Properties are also classified, and they are used to describe material and cultural aspects of the object. Properties can also contain referrals to external documentation of different kinds: drawings, product data sheets, images etc.

These Industry practices does not cover how these Composition structures are to be composed. For implementation in software it is necessary that each class, regardless of level, can stand on its own and be constructed as the user sees fit. This means that the software does not need to limit the user on which class should be possible to add to another class. However, it is recommended that the software gives help and context to the user when composing Composition structures.

4.2.3 Reference designations

These structures can also be used in Reference designations, identifying occurrences of objects. Rules for Reference designations in CoClass is a combination of recommendations in SS-EN 81346-1:2010 and SS-ISO 81346-12:2019. Example follows below.
To show which table is referred to, a table code can be used within angle brackets, i.e. <BX>, or with colon after, i.e. BX:. It is recommended to use colon in software applications. Table codes can be found on each table's information page.
A prefix is used to show which aspect of the object that is described:

=	Function	What an object is intended to do or what it actually does
-	Product (Assembly)	By which means an object does what it is intended to do; with what parts the object is assembled

+	Placing	Intended or actual placing of the object
++	Location	Intended or actual location of the object
%	Type	Type of construction element within the same class

In order to differentiate between construction elements of different complexity, the code for *Functional systems* has one letter, the code for *Constructive systems* has two letters, and the code for *Components* has tree letters.
An example is shown below:

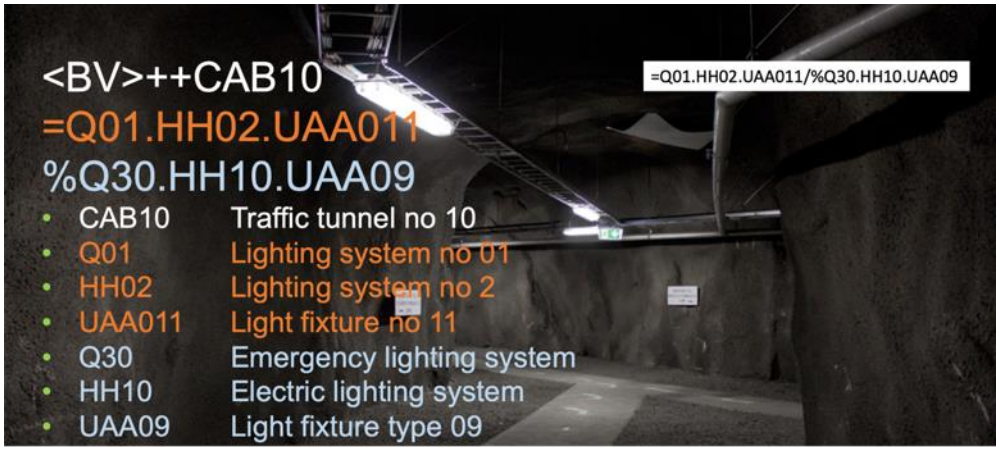


Figure 5 – Reference designation

As an alternative to showing the type aspect, numbered types can be used in the function or product aspect as shown below:

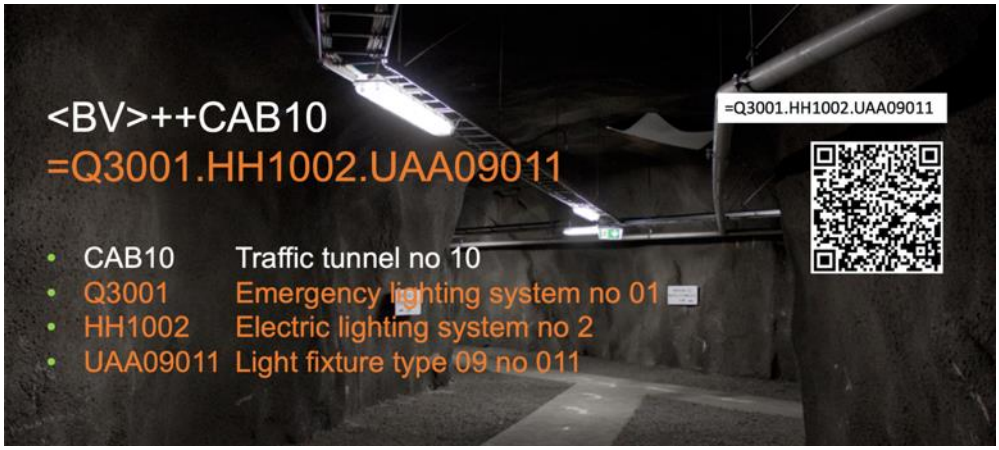


Figure 6 – Reference designation, numbered types

It should be possible to view information and objects from several different aspects and perspectives, depending on the users needs.

4.2.4 Property types

There are a number of Properties in the Properties table that can be added to each CoClass class in a Composition structure.
Properties in CoClass have a Swedish and English term, a code and a recommended attribute tag. Example: Märkström, Nominal current, ELCN, ccElectrCurrentNominal.
The code can be used for expressing property values in a text string. Values are added after the code of the property. The values are surrounded by parenthesis and separated by semicolon. Example: BV:AAA (ARTX:25; ARVS:Yes) shows a BV:AAA Room with ARTX ccAirTemperatureMax of 25 and ARVS ccAirVentilatedByAirSupply set to Yes.
Unit and value type are defined for all properties. Possible value types are:

- Boolean
- String
- Integer
- Real number
- Value list
- Percent

It is recommended to not include Properties and their values in a Reference designation if it is to be used as an identifier for the object.

4.3 International standards

CoClass is based on the following international standards:

- SS-ISO 12006-2:2015, Building construction – Organization of information about construction works – Part 2: Framework for classification
- IEC-EN 81346-1:2009 Industrial systems, installations and equipment and industrial products – Structuring principles and reference designations – Part 1: Basic rules
- IEC 81346-2:2019 Industrial systems, installations and equipment and industrial products – Structuring principles and reference designations – Part 2: Classification of objects and codes for classes
- ISO 81346-12:2018 Industrial systems, installations and equipment and industrial products – Structuring principles and reference designations – Part 12: Construction works and building services

The ISO 12006-2 standard defines a classification system for a built environment, how you structure the built environment.
IEC 81346-1, IEC 81346-2 and IEC 81346-12 defines the Reference designation, the classes for construction elements (components), and the built space.

5 Usage of CoClass in software

The basis of usage in software is the CoClass classification system, based on table codes, class codes and their relationships, applied according to the rules for Reference designations.
This industry practice does not specify which format to use when communicating the information. However, the CoClass API uses JSON.
The documentation for the communication of classes, properties and other objects is available at: https://developer.byggtjanst.io/docs/services/sb-public-api-coclass-public
The documentation for the communication of Composition structures based on classes, properties and other objects is available at: https://developer.byggtjanst.io/docs/services/sb-public-api-coclass-public-structure

5.1 Guidelines for implementing CoClass

Based on the work in this, and other, projects a number of guidelines on how to implement CoClass in applications have been developed.

1. It shall be possible to transfer CoClass classification information from one application in another.
   When using CoClass codes, Reference designations and Composition structures in an application, and objects are possible to export from the application, the corresponding CoClass information shall be exported together with it. If a Composition structure is possible to export from an application, the corresponding CoClass codes and Reference designations shall be exported together with them.
2. It shall be possible to verify the CoClass codes, the Reference designations and the Composition structures.
   In order to ensure that the CoClass information in the application is possible to transfer to other applications, the verification shall ensure that the codes, Reference designations and Composition structures

are valid according to the rules and guidelines set out in this document and the corresponding standards that CoClass is based on.

3. When referring to CoClass codes and information, the corresponding CoClass version shall be specified.
4. The language used when referencing CoClass and its content shall be specified.
   The CoClass classification is language independent, that is, the class codes are the same for all languages. However, the information contained in each CoClass object is available in Swedish and English. The default language for CoClass is Swedish.
5. When presenting a CoClass code to a user, it shall always be accompanied by its corresponding heading.
6. Information as to what CoClass codes are available based on the use case in the application shall be presented, and if possible, which are suitable to use at that time.
7. Relevant context information depending on the user's use of the application shall accompany the CoClass information that is presented.
8. It shall be possible to switch between English and Swedish without losing CoClass classification information.

5.2 Version control of CoClass

In implementing CoClass in software, it is important to include and handle different versions of CoClass, since the meaning and content of codes change based on the version of CoClass.

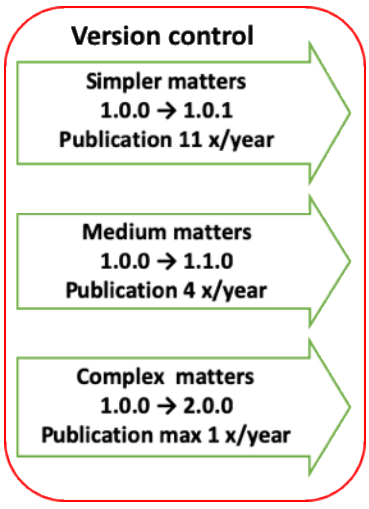


Figure 7 – Overview of levels of versions in CoClass

Updates to CoClass is handled with version control, in three severity levels:

1. Simpler matters (publishing takes place at most 11 times / year): Minor linguistic corrections, adjustment of definitions, adjustment of examples, additional or replacement of existing images, additional or change of "Corresponding of" (refers to mapping to other standards).
2. Medium matters (publishing takes place at most 4 times / year, but in consultation with Product Manager CoClass can be extended to a maximum of 11 times / year): New and / or modified terms, new and / or canceled codes (including removal of codes).
3. Complex cases (publishing takes place at most 1 time / year): Changes that affect one or more groups of codes and / or are deemed to be time-consuming and / or need to be coordinated with the publication of other products, depend on other activities and / or requires IT development. These cases are, if possible, published in connection with major annual releases, but in special cases may be forced to be postponed in anticipation of the next AMA cycle, i.e. up to three years.

Older versions of CoClass are accessible through the CoClass API and CoClass Studio.

5.3 Other standards and industry practices

There are several Industry practices, standards, and formats that are relevant for CoClass and its use in software. These include, but are not limited to:

- BIMTypeCode - http://www.bimstockholm.se/EN/typecode.html
- BIP codes - http://www.bipkoder.se/
- CityGML - https://www.opengeospatial.org/standards/citygml
- fi2xml - https://www.bimalliance.se/verktyg-och-stoed/standarder/datamodell/fi2xml/
- IFC - https://www.iso.org/standard/51622.html
- InfraGML - https://www.opengeospatial.org/standards/infragml
- Lantmäteriets Nationella specifikationer https://www.lantmateriet.se/sv/Om-Lantmateriet/Samverkan-med-andra/lantmateriet---utvecklingsmyndighet-for-samhallsbyggnadsprocessen/nationella-specifikationer/

It is proposed that this area be the focus of an additional project in the Smart Built Environment program.

6 Interaction between software

Interaction between software should use the above principles in order to ensure that information related to CoClass classification and structures is communicated in a clear and unambiguous way. Interaction shall be implemented by utilizing open standards when possible.

In order to access the latest version of the classification system and its components, Svensk Byggtjänst provides IT services for CoClass Studio and the CoClass API, located at https://coclass.byggtjanst.se/ and https://developer.byggtjanst.io/ respectively.

There are several ways to interact with these services in software.
One way is to create a local copy of the classification system and use that internally. Another is to have an interactive connection to the CoClass API. The CoClass API will always contain the most updated version of CoClass.

An overview of Svensk Byggtjänst's services and how to use them is provided in *Appendix 2 - CoClass Studio and API*.

7 Future development

Based on these Industry practices, potential areas of future work have been identified in order to facilitate the implementation and usage of CoClass in software.

7.1 CoClass as a naming standard

The project believes that CoClass as it is now work very well as a Classification standard.
However, there is a need to ensure that CoClass can be used as a naming standard based on common principles.
It needs to be developed, defined and described what this means for implementers of CoClass in software.
It needs to be described what implementers need to follow, and how this can be done in a flexible way to meet different perspectives of the information.

The project proposes that the CoClass Management group initiates a reference group that clarifies this.

7.2 Verification of CoClass information

In order to ensure that CoClass Codes, Reference designations and Composition structures are used in the same way from different applications, the Industry practices proposes that this verification is possible to do via the API that Svensk Byggtjänst provides.
This verification should be able to answer:

- If a specific CoClass code is valid, and give information relating to that code
- If a specific Reference designation is valid, and what CoClass objects it consists of
- If a specific reference designation is valid according to a specific composition structure
- If a specific Composition structure is valid, and what CoClass objects it consists of

7.3 Modification of Composition structures

In order to work seamlessly with CoClass information in more than one application, there is a need to be able to not only retrieve a Composition structure, but also to modify it.
A possible way to do this is to open up API calls for the modification of structures stored in CoClass Studio and other applications.
There is also a need to be able to track changes in composition structures via version control.

7.4 CoClass API Improvements

Based on the Proof-of-Concept implementations, and future work, further improvements and feature additions to the CoClass API have and will be found. It is crucial that these are prioritized with the help of relevant stakeholders and implemented in future versions of the CoClass API.
In order to facilitate this, the Industry practices proposes that a reference group for future development is established.

7.5 URI

The classes in CoClass are unique resources that needs to be identifiable in an unambiguous way from other data sources. Currently, each class and table is identified (or referenced) by a code which essentially is a string of (a few) characters. This string of characters is not contextualized explicitly in the way that the code by itself shows that it belongs to CoClass. The users therefore need to know through other means that this is the case.
Furthermore, the code does not explicitly provide means for the user, or a

machine, to find more information about the concept in question. This principle still works well in most situations. It is also well integrated with the reference designations, described elsewhere in this document.

One way to address this issue was suggested by the VERA project. According to this suggestion, each class and table in CoClass could be referenced using resolvable internet URI:s, directly linking the corresponding CoClass concept in both humanly- (e.g. html) and machine- (e.g. JSON, XML, Turtle) readable form. A URI (Universal Resource Identifier) is a string of characters that unambiguously identifies a particular resource as specified by the W3C. Among the forms of URI:s, the URL (Uniform Resource Locator) is probably the most well-known.

By using URI:s, CoClass may be easily and unambiguously referenced from anywhere in a machine-interpretable fashion that also works on the world wide web. It is worth noting that this would be an alternative to the use of class codes that still may be a viable option for many users.
To achieve this, a URI strategy should be defined. To do this, it is recommended to use the following W3C guidelines: https://www.w3.org/TR/ld-bp/. For URI:s, the following recommendations are made:

- Use HTTP URI:s
    - *To benefit from and increase the value of the World Wide Web, governments and agencies should provide HTTP URIs as identifiers for their resources. There are many benefits to participating in the existing network of URIs, including linking, caching, and indexing by search engines. As stated in [howto-lodp], HTTP URIs enable people to "look-up" or "dereference" a URI in order to access a representation of the resource identified by that URI. To benefit from and increase the value of the World Wide Web, data publishers should provide URIs as identifiers for their resources.*
- Provide at least one machine-readable representation of the resource identified by the URI
    - *In order to enable HTTP URIs to be "dereferenced", data publishers have to set up the necessary infrastructure elements (e.g. TCP-based HTTP servers) to serve representations of the resources they want to make available (e.g. a human-readable HTML representation or a machine-readable Turtle). A publisher may supply zero or more representations of the resource identified by that URI. However, there is a clear benefit to data users in providing at*

least one machine-readable representation. More information about serving different representations of a resource can be found in [COOLURIS].

- A URI structure will not contain anything that could change
  - *It is good practice that URIs do not contain anything that could easily change or that is expected to change like session tokens or other state information. URIs should be stable and reliable in order to maximize the possibilities of reuse that Linked Data brings to users. There must be a balance between making URIs readable and keeping them more stable by removing descriptive information that will likely change. For more information on this see Architecture of the World Wide Web: URI Persistence*
- URI opacity
  - *The Architecture of the World Wide Web [webarch], provides best practices for the treatment of URIs at the time they are resolved by a Web client: Agents making use of URIs should not attempt to infer properties of the referenced resource. URIs should be constructed in accordance with the guidance provided in this document to ensure ease of use during development and proper consideration to the guidelines given herein. However, Web clients accessing such URIs should not parse or otherwise read into the meaning of URIs.*

8 Maintenance of the Industry practices

This Industry practices proposes that Svensk Byggtjänst establish a reference group for maintaining and developing these Industry practices. It is proposed that the reference groups tasks should be:

- Maintaining and developing the *Industry practices for application of CoClass in software*
- Proposing future improvements and changes to the CoClass API and CoClass Studio application

It is proposed that the Chair for the reference group is Svensk Byggtjänst, and that initially the group consists of representatives from the project that has developed these Industry practices.
The Chair of the reference group is responsible for setting the timeline, process and agenda for the reference groups work.

9 Annexes

Appendix 1 – Report, Industry practices for application of CoClass in software
Appendix 2 – CoClass Studio and API
Appendix 3 – Definitions

SMART BUILT
ENVIRONMENT

Med stöd från:

VINNOVA

Energimyndigheten

formas

STRATEGISKA INNOVATIONS-PROGRAM

Smart Built Environment c/o IQ Samhällsbyggnad | Drottninggatan 33 | 111 51 STOCKHOLM | info@smartbuilt.se | 070-645 16 40 | www.smartbuilt.se