Needs Deriving from the Design of the Updated Architecture of the Building Data Standard (DSS)

Contents

1 Introduction

1.1 Purpose

This document describes needs deriving from the design of the updated architecture of the Building Data Standard (DSS). These needs may result in demands on CoBuilder. The document is intended to provide CoBuilder with well-informed input on the needs of the DSS updates in question.

This document has been written on the understanding that we are not able to assess in detail which tools in the CoBuilder family may be affected by the update – Define, Public View, Link, or any other tool. The purpose of the document is to describe the needs that have been identified; we are not in a position to judge whether those needs impose any demands on tools, or on which ones.

The document outlines our needs by describing the target status that we wish to achieve. It may therefore contain parts that are descriptions of the current situation, even where no change is required, or where we are not exactly sure of the current functionality of the CoBuilder family of tools. The document does not aim to provide CoBuilder with a brief that differs from the current situation, i.e. to describe individual requirements that, applied to the current situation, would lead to the expected future situation. Such change-related requirements will be expertly defined by CoBuilder itself for its individual tools. Where we consider it appropriate for purposes of clarification, the document also explicitly describes the current situation, with a sentence to the effect that "CoBuilder systems are already compliant".

1.2 Key

For the reasons outlined above, tool names such as Define, Public View, and Link are not used in the text. The fulfilment of the requirements does not depend on which tools are used to achieve this, assuming, of course, that the user-friendliness, data availability, confidentiality, integrity, and undeniability of the performance of user activities are maintained. However, as it is impossible to circumvent the need to describe the system for DSS editing by experts and the system for viewing DSS by the public, we define the following terms:

- **Writer**
  = tool for editing DSS by experts.
  In other words, this is the administration space for DSS creation and management.
  The requirements for Writer are described in the introductory section 3 "*DSS conceptual data model*" and in section 4.1 "*Functionalities required for Writer*".

- **Browser**
  = a tool enabling the public to use the DSS.
  In other words, this is a public portal for DSS use, for making requests for information based on the chosen/selected CCI context and UIP prerequisites for a particular user case. Browser allows users to log in and, within their user profile, enables them to create and manage their xIR requests.
  Browser requirements are described in section 4.2 "*Functionalities required for Browser*"
- **API**
  = API for the machine retrievability of DSS content
  API functionality requirements are described in section 4.3 "*Functionalities required for data access via API*".

The other terms used in the document (e.g. data template, property, property group) are sufficiently well known to both parties (the Czech Standardisation Agency (CAS) and CoBuilder). If a term needs to be made more specific, this is done directly in the relevant passage of the text.

To make the text shorter and clearer, in some cases “data template” is referred to simply as “template”.

The term **CCI facets** includes the facets and their levels described here:

- CCI 0 Construction Complexes
- CCI 1 Construction Entity
- CCI 2 Built Space
- CCI 3 Functional Systems
- CCI 4 Technical Systems
- CCI 5 Component

The term **UIP** refers to the “level of information needs”. The term **UIP facets** includes the following facets:

- Milestone
- Actor
- Use case

2 Summary

This section takes the form of an “executive summary” and describes the most important needs in bullet points:

- Link is currently capable of filtering in combinations under section 6 “Description of the structure of the link of the CCI context and UIP prerequisites between the property and the template” only for UIP facets. CCI facets need to be added to the same behavioural model.
- The values of CCI facets need to be imported in the source format, and the image of CCI facets in the system needs to be kept up-to-date even when the CCI classification is being developed.
- The term “Identification” is introduced to extend the classification. This term is explained in 3.4 “Data template”.
- Entities in the SNIM classification tree will be determined by Identification: SNIM, while a parallel classification, is uniquely (1:1) mappable as classification + Identification. The author of the Identification codes will add the SNIM code to the Identification values. A data template’s SNIM code is then determined automatically by the system.
- In addition, the data template will be tied to a value in the KLASS code list. The system will retrieve the code list via API and keep its image in the system up-to-date (see 3.4 “Data template”).

3 DSS conceptual data model

In order to properly understand this document, it is recommended to first read the document "*Description of the Building Data Standard (DSS) Architecture Concept*".

The individual subsections of this section describe the CAS's individual needs with respect to DSS updates. The needs are not presented in any specific order.

The DSS glossary unifies the minimum requirements for the properties of construction objects for all cases involving the CCI context and UIP prerequisites.

3.1 Needs related to enumerative data types

Property with an enumerative data type:

- is linked to a specific code list (as an entity, a code list instance)
- defines whether a single value from the code list or multiple values from the code list are allowed in its use, i.e. whether the link between the property and its value is single or multiple.

Code lists have the following properties:

- code lists are subject to creation and management by the DSS experts– they are therefore managed in Writer
- code lists are closed (users – anyone outside Writer – cannot edit them)
- code lists are single-level; they are not trees
- a code list and its fields are linked – mapped – to IFC Enumeration (IfcPropertyEnumeration). We need to have the option of editing this mapping (example: the local convention for right/left doors is the exact opposite of what is prescribed by the IFC. We intend to resolve this problem by "reverse" mapping between the code list value and the corresponding IFC value). It must be possible to map both the code list entity as a whole and individual items onto the IFC
- the system must inherently enable an unlimited number of code lists to be created
- changes to code lists are subject to proposal and approval
- the system must be able to retrieve references to a selected code list item, even in older versions of DSS
- the value of code list items is generally any data type; in the case of texts, multilingual text
- the display of the code list in the user interface must facilitate value lookup, preferably using the "incremental search" method

3.2 Facet-related needs

3.2.1 Needs for facets in general

The individual CCI + UIP facets have a hierarchical (tree) structure with unlimited hierarchical subdivisions (8 floors are probably sufficient in practice). Writer must extract CCI facets via API or controlled import from the formats in which the CCI is published.

From the perspective of the internal data structure, individual CCI facets and individual UIP facets can be handled by the same technical means – UIP facets are subject to the same requirements as CCI facets. There is no reason for the CCI and UIP facets to be handled by different technical means in the system.

The needs specified for code lists (the approval of edits, the retrieval of references, etc.), as described under 3.1 "*Needs related to enumerative data types*", are generally applied to facets.

If the classification tree is expanded to include a new value, the system will enable the following operations:

- Fill in the requirement in all combinations
- Do not apply the requirement anywhere (empty requirement)
- Copy requirements from another existing code list item as selected by the user
- Copy requirements from another existing code list item as selected by the user only for selected property groups or only for selected templates

3.2.2 Needs for CCI facets

When we describe the attribute (data entity composition) of a "CCI facet" in the text of a data model, we generally have 1 to 6 CCI facets in mind. Not all facets need necessarily be filled in; some may be irrelevant (if we assume that there will be a data slot for each facet, some will be null, but this method of implementation would be unwieldy, since in general there may be a varying number of CCI facets).

3.2.3 Needs for UIP facets

Unrestricted hierarchisation is also required for UIP facets (e.g. two hierarchical levels will be needed for the Actor facet). However, their number is fixed – there are three facets: Milestone, Actor, Use Case.

3.3 Needs related to properties in templates

Properties are grouped into templates via Property Groups.

A property in a template has mapping onto PSet-Property as per the IFC standard.

3.3.1 Property groups

There are several types of property groups:

General Property Group

Definition: contains properties that need to be recorded for all templates without distinction. In DSS, there is one General Property Group with fixed properties.

Specific Property Groups

Definition: property groups specific to a given template or templates that can be shared entirely, i.e. one and the same specific property group may appear in multiple data templates. There may be several of these property groups within a data template.
In DSS, we must somehow be able to express the fact that a data template, if used in a particular project, needs to be able to specify the application of a particular construction element it is describing in that project via CCI codes. The particular Specific Property Group that is used in each template is therefore a CCI group for objects that contains CCI class codes chosen according to the application of a data template in a particular project.

Both of the above types of property groups are collectively called **Systemic Property Groups**, and they:

- are used for the administration of the data standard in Writer.
- are not available to Browser (Browser knows nothing – they are not displayed or exported)
- a property may appear in more than one property group.

Thematic Property Group

Definition: groups properties together for them to be displayed in Browser.

When the content of a data template is displayed in Browser, all properties are displayed, but grouped by thematic property groups.

In other words:

- thematic property groups are collections of properties that are displayed "clustered" together.
- a thematic property group is global. If the properties to be displayed in a template belong to the same thematic group, they are displayed clustered together.

Thematic property groups are intended solely for ease of user orientation and have no meaning in terms of data connectivity.

The grouping of properties into thematic groups can be thought of as the labelling (tagging) of properties in a template with sticky labels: yellow, green, red, etc.

A property may be in one thematic property group in a given template. The point is that these are properties per se (a property as an entity), not properties in a template. In other words:

- the inclusion of a property in a thematic property group is independent of its assignment to a specific template
- a property has a "thematic group" attribute, is not multiple, and must be filled in.

In Writer, the list (code list) of thematic property groups can be managed centrally (this list is global in DSS and does not depend on data templates).

3.3.2 Link between a property and a template from the perspective of the CCI context and UIP prerequisites

When a property group is inserted into a data template, a link is created between each property and the template. This link is used to filter properties in the template.

3.3.2.1 Parameters of a link between a property and a template

The parameters of a link between a property and a template have two parts:

- **CCI**
  The CCI facets that are included in the link depend on which construction object the data template is created for.

- **UIP**
  A link always has all three facets: Milestone, Actor, Use Case

The data of a link is entered in Writer. For each facet that is part of a link, the following must be specified:

- zero value (or a reference to the root of the facet tree): indicates that this facet is not decisive for the link. In other words, the link includes all the values of the facet; or, when filtering, the value of the filter in this facet is not important.
- one or more values (the logical sum between values applies – conjunction): data is defined, on the basis of which it is possible to decide, in this facet, whether or not the property matches the filter specified by the user.

The input of these values into the filter in Browser (and also Writer, which should also be able to filter) is a separate issue. In view of the considerable scope of this description, it is presented in the separate section 6 "*Description of the structure of the link of the CCI context and* UIP prerequisites between the property and the template".

3.3.2.2 Representation of properties via IFC

At the level of a data template, the mapping of the properties contained therein onto a combination of an existing property set (Pset) and a property from IFC will also be addressed; for this purpose, it will also be possible to use the common international table created on the basis of IFC 4.0.2.1 (or the subsequent version once it has been approved as an EN ISO). For an expanding (i.e. not included in IFC) property sets and properties, it will be necessary to use a separate table, which is gradually created and continuously updated, containing at least the code designations in IFC syntax and the names in international English and Czech.

3.4 Data template

3.4.1 Introduction

A data template is a set of systemic property groups containing properties relevant to what it describes (the construction object being addressed), potentially applicable to at least one combination of CCI context and UIP prerequisites as per EN 17412-1. At the level of the data template, a record will be made, for each property contained therein, specifying the combinations of CCI context and UIP prerequisites to which it applies. This will facilitate the filtering of the contained properties in order to obtain a predetermined specification of the level of information needs.

A data template is a core DSS component that is linked to a single internationally defined construction object and contains the relevant properties via a selection of relevant systemic property groups. It also contains its own metadata (attributes) that is needed for its description and management. Data templates will not be created randomly in DSS, but will use the CCI structure.

The term "data template" is not the same as the term "construction object"; there may be multiple data templates for a single construction object.

3.4.2 Data structure

A data template has the following data structure:

- *Metadata*

  note: this is simply a matter of organising the description of this text; the Metadata level has no actual relevance to the data model; it is not a separate data entity. Both Browser and Writer will enable searches based on all of the data listed below.

  - *Descriptions*: such as full name, technical definition, examples
    … CoBuilder systems are already compliant
  - *Attributes*: such as name, status, who created it and when, visibility, GUIDs
    … CoBuilder systems are already compliant
  - References – reference document
    … CoBuilder systems are already compliant
  - *Construction object* – a relationship to an internationally defined item grouping data templates also outside DSS, managed by various entities and applicable to various countries and various needs.
  - *Classification* – a link to one specific node of a specific CCI facet. Meaning: the classification indicates "what" the data template is created for, e.g. in the case of a roof, whether it is for the entire Functional System (CCI 3) "D-Roofing System", or for the Technical System (CCI 4) "BE-Roof Structure", or for the Component (CCI 5) "NCE-Roofing", etc. In other words, the classification is linked not only to a specific CCI, but also to its specific node (which does not have to be a leaf in the classification tree).
  - *Identification* – a reference to a specific value in the appropriate code list that corresponds to the *Classification*. Meaning and reason for introduction: a new data template for the same construction object is created when it is necessary to describe a construction element or a structure with a different composition of properties (even if it differs by a single property) that cannot be controlled by Classification or UIP. This might concern, for example, technological variants of a construction element - e.g. a brick/concrete/glass/wooden wall. The CCI hierarchy ends with the term "wall". Since the individual variants of the construction element "wall" have different properties, a separate data template needs to be created for each variant. These data templates will have the same classification, but will differ in their identification. The identification expands on the classification and makes it possible to *identify* a brick, concrete, glass or wooden wall. This usage also implies the clear requirement that for each node (although this will be used primarily for leaves of the tree) of each CCI facet, a code list exists in Writer to allow identification. Taking the construction object "wall", this code list will have, for example, the values brick / concrete / glass / wood.
    Therefore, identification is perceived as expanding on the classification (making it more specific). In addition to the code (e.g. 00), name/value (e.g. "brick"), this code list will also contain mapping onto the *SNIM TSP-PSP* code list (e.g. AB00).
  - *IFC Class* – mapping of the template onto the corresponding specific IFC class (Ifc_Class, e.g. IfcDoor)
  - *CZ-Documentation* – classification according to Decree 499/2006 on the documentation of buildings (code list)
  - *Cz_MMR* – link to no more than one specific value in the code list defined by the Ministry for Regional Development of the Czech Republic (abbreviation "MMR") under the designation "KLASS". This code list is defined only for certain CCI facets.

Writer will retrieve the code list via API. For data templates created for construction objects other than Construction Entities and Construction Complexes, this attribute will be null.

- *Specific Property Groups*… CoBuilder systems are already compliant
  These are individual Specific Property Groups.
  All properties are IFC mapped. If such a Pset or property is not available in the IFC ISO, DSS will create its own CZ_PSet or CZ_Property.

4 Functionalities required for each application

The main functionalities required can be categorised as follows:

4.1 Functionalities required for Writer

Respect the content of section 3 "*DSS conceptual data model*", allow for the entering/editing/deleting of data (the marking of data as deleted) + the approval such proposed changes.

All records are dynamic: it must be possible to add, edit and expand the data dictionary, all code and other lists, including the number of levels of CCI and UIP facets.

Logged-in users must have the option, directly in Writer, to carry out the same types of filtering and data display, including exports, as a public user in Browser.[1] This filtering will be carried out over the current DSS status, as visible to the logged-in user; it may therefore include unapproved DSS change proposals, i.e. work in progress.

4.1.1 User roles

User roles that will be used by each application:

- OTO
  - Enters proposals for DSS additions and modifications
  - Processes suggestions from the public
  - Creates and saves "recommended" (default) CCI context and UIP filters[2] ("presets"). Browser users are then shown, as a matter of preference, the default preset with the option of adding or removing properties in the given context and UIP based on their needs. Users can subsequently edit and save this preset. Changes proposed by users in this way are used for feedback on DSS settings (the submission of suggestions for DSS modifications).
- DSS administrator
  - Checks and approves OTO proposals
  - Approves the publication of the DSS version

[1] For this reason, these are not described in this section, but are covered in section 4.2 "*Functionalities required for Browser*"

[2] *In the combinations as per section 6 "Description of the structure of the link of the CCI context and* UIP prerequisites between the property and the template*"*

4.1.2 DSS publication in Browser

DSS is always published in full (not just in part). No change (even if approved) in Writer will feed into the published data other than by the publication of DSS in full.

4.2 Functionalities required for Browser

Many of the requirements are already met by the current version of Browser, such as the display of EN texts and CZ translations.

4.2.1 DSS publication in Browser

DSS is publicly released with a time stamp, a new version of DSS is created upon publication.

Browser must:

- enable a retrospective view of the database status for a specific version of DSS and enable full use of an older version of DSS
- provide a log of all changes between different DSS versions or show changes/differences between two different DSS versions

The administration of all code lists is subject to the same rules as DSS, e.g. approval, versioning. Code lists are an integral part of DSS publishing and versioning.

4.2.2 Presets

As described in 4.1.1 "*User roles*", the OTO creates and saves "recommended" (default) CCI context and UIP filters (presets). Browser users are then shown, as a matter of preference, the default preset with the option of adding or removing properties in the given context and UIP based on their needs. Users can subsequently edit and save this preset. Changes proposed by users in this way are used for feedback on DSS settings (the submission of suggestions for DSS modifications), which is handled separately from CoBuilder tools (the provision of feedback is not a CoBuilder requirement).

Users may save their own presets and manage saved presets.

Browser will allow external user authentication via NIA (www.identitaobcana.cz).

4.2.3 Data browsing

Data can be browsed freely without applying a filter. Browser will display all data.

Without applying a filter, users can browse data templates without restriction, i.e., the unification of all required properties in any combination of CCI context and UIP prerequisites. When a filter is applied, the request is restricted (refined) and thus the properties required for the user-specified combination of CCI context and UIP are reduced, as described below.

4.2.4 Template filtering

With Browser, templates can be filtered by metadata (see 3.4 "*Data template*"). The filter results in a list of templates.

4.2.5 Property filtering

There is therefore a separate filter for templates (the range of the required templates that are to be displayed is defined by a separate filter). The UIP facet filter does not affect the range of data templates found.

In each filter, multiple items can be selected, all items can be selected, or none can be selected.

Individual filters usually have a tree structure. Therefore, in order to select a filter, it is necessary to be able to select one value at any level, select the entire sublevel, and remove any subvalue of the sublevel. When selecting a parent tree item, it is necessary to distinguish whether this is the selection of all subordinate items or an individual selection of an item of this level. Even an individual item (that is not a leaf of a tree) may be subject to an xIR requirement as a separate construction element.

It will be possible to link predefined typical filters (presets) to certain parts of the definition of a CCI context in order to make the user interface more friendly for the user. For example, selecting a technical system pre-selects a relevant range of templates.

The CCI context may affect the recommended range of templates for xIR. The user interface should allow users to choose to apply a context to the template list (yes/no). In the “Yes” case, the template display will be controlled by the presence of at least one property in the defined (specific) property groups in relation to the selected CCI context. If no property is required in the template for the specified CCI context and UIP prerequisites, this data template should not appear in the filtering result.

4.2.6 Displaying properties in a data template

When a data template is displayed, its properties are shown to the user. Browser allows filtering via a CCI and UIP filter.

4.2.7 Filtering properties in displayed data templates

The setting of the CCI context will be an essential feature in the use of the template filter. Browser will only display data templates that match the required CCI context. If users subsequently apply a filter to UIP prerequisites, the system filters the properties in the data templates accordingly. This filter results in what are technically data templates, and in each template only the properties matching the filter. Users can view the filter results and export them to IDS, IfcXml and Microsoft Excel (xlsx) formats.

4.2.8 Exporting filtered data templates and properties

Filter-based exports in IDS, IfcXml, and Microsoft Excel (xlsx) formats.

Exports will also contain a unique template identification. The system makes it possible to retrieve and export the data of a data template according to its unique identification.

When requirements for information on a selected construction object are being compiled, those properties from the data template that are applicable to the combination of the particular UIP prerequisites and CCI facets will be selected. On the basis of the predefined selections (presets[3]) for the various combinations, end-users will be able to retrieve the requirements for information corresponding to their selected combination or combinations.

Depending on the selection of the CCI context and UIP prerequisites, multiple prescriptions for one construction object (represented by one or more data templates), structured according to the combination of the selected CCI context and UIP prerequisites, will be automatically created in the xIR requirements.
Example: if two technical systems contain the data: JA – gas distribution system and JB – water

[3] *In Link, this evidently corresponds to the term “Templates”. We do not want to use this term here so that it is not confused with data templates in the English translation of this text.*

distribution system, then one xIR basically results in 2 requirements for a Component – the construction object “valve” with different required properties – a valve for gas and a valve for water.

If no CCI context or UIP prerequisites are entered, the system will return all relevant breakdowns for all combinations of contexts and UIPs as a filtering result and include them in the export. The relevant ones will be derived from the specified CCI contexts and UIP prerequisites of the properties required in each data template.

The xIR breakdown is based on the selected level required, including facets extended to include identifications. That is, according to all the required context members (including identification) that are selected.

Additional requirements regarding user functions

The goal of the DSS user is to construct xIR requirements (information requests) for the user’s particular case based on a data dictionary that contains the unified minimum for all possible combinations of UIPs and contexts.

Users will be able to expand the proposed xIR requirement to include other properties they require, but only by selecting them from the DSS dictionary.

In Browser, users will be able to narrow the recommended proposed xIR requirement by removing/adding/editing certain requirements.

In cases where the context filter definition makes the classification of the construction object unambiguous, the general properties in xIR will be populated with the CCI classification value in the relevant facet. For example, if an xIR is created for a specific Technical System (e.g. JF), the construction objects in the xIR (or in the exported data) will have a pre-populated value of CCI4 = JF.

4.2.9 Public commenting on DSS

Public DSS commenting, and the collection of ideas and feedback from the public, will be handled separately from the CoBuilder tools.

However, the data dictionary will be managed within the Define tool. Here, there must be the possibility of entering change requests, e.g. in the form of comments, so that, for example, it is possible to enter a requirement for the existence of a property in a template, regardless of which property group it is to be added to. The disadvantage of property groups needs to be addressed: adding a property to a property group adds the property to all data templates that contain that property group. This is not always wanted. Suggestions are used by the OTO and the DSS administrator to create proposals for DSS modifications. The resolution of suggestions will be recorded and made available to users.

4.2.10 Fetching code lists via API

Writer will obtain at least the CCI classification and the CZ_MMR code list via API. Writer must reflect the changes made by code lists on the source side (the addition/removal/change of an item).

4.3 Functionalities required for data access via API

DSS accessibility via API with the possibility of entering a required UIP, CCI context and template filters according to metadata.

The query and API response will reflect Browser capabilities.

5 Data migration

Data migration is not addressed in this document; however, it is part of the work and will be covered in subsequent work once an analysis has been conducted.

6 Description of the structure of the link of the CCI context and UIP prerequisites between the property and the template

6.1 Introduction

Link is currently capable of filtering in combinations under this section only for UIP facets. CCI facets need to be added to the same behavioural model.

This section describes the details of the requirement "Presence of a property in an xIR requirement is determined by the **setting of the link** of the CCI context and UIP prerequisites between the property and the template".

This section covers the correct "**setting of the link**" mentioned in the preceding sentence.

The result of the filtering of properties in selected data templates depends on the combination of the CCI context and UIP filter options. So, for example, for one combination Milestone1 + Entity1 the property will be required, but for the combination Milestone2 + Entity1 the same property in the same template will not be required.

Important note on terminology: for the sake of clarity and to keep the text concise, we will make a deliberate simplification in this section: instead of using the exact term "property in the data template" or even better "linking entity between the property and the data template", we will use the abbreviated term "property".

The purpose of this section is to describe the property filtering requirements with respect to the interdependency of CCI and UIP facets. The section describes requirements for filtering properties in data templates when the public uses DSS via Browser. In order for Browser to be able to meet these requirements, additional data must be available. Therefore, this section also describes requirements relevant to the creation and management of the data dictionary in the part defining the data of properties in templates, i.e. the entering of properties in Writer.

The section does not describe searches for data template entities (i.e. template filtering). The document only describes the filtering of properties in templates.

The aim of this section is to describe only the requirements for "parameter combinations" that can be used to filter properties in templates.

The aim of the requirements described here is to enable the public, in Browser, to filter properties in templates by CCI (which specifies the classification context) and UIP (the level of information needs – actor, milestone, method of use) in such a way that the individual filter settings for each facet are interdependent (i.e. the occurrence of a property is determined by a combination of filter settings).

The aim of the change is described by the individual requirements set out in the points below:

6.2 Available facets

We envisage the following CCI facets:

Note: the numbering in the CCI facet names is not taken from CCI, but has proven effective for ease of communication.

- CCI facets
  - CCI 0 – Construction Complexes
  - CCI 1 – Construction Entities
  - CCI 2 – Built Space
  - CCI 3 – Functional Systems
  - CCI 4 – Technical Systems
- UIP facets (UIP = level of information needs)
  - Milestone
  - Use case
  - Actor

The facets that will be displayed to users for filtering

1. Browser must allow for the filtering of properties in templates by CCI and UIP facets
2. The scope of the CCI facets depends on the CCI level for which the data template is built – whether the data template is for a Construction Complex (CCI 0) or a Technical System (CCI 4). In general, if a data template is built for CCI n, then all facets CCI 0 to n-1 must be displayed to the user for filtering.

Therefore, Browser must be able to filter properties in templates (in addition to the relevant UIP facets) according to facets defined by CCI 0-n.

In a limiting case where a data template is built for a Component (i.e. CCI 5), the properties in templates need to be filtered according to all CCI- and UIP-defined facets, i.e. the 8 facets CCI 0,1,2,3,4+ (milestone, actor, method of use).

Examples:

- Where a template is built for a component, the filter will comprise the following facets: (CCI 0-4, i.e. 5 facets) + 3 UIP facets.
- Where a template is created for a built space, the filter will consist of the following facets: (CCI 0-1, i.e. 2 facets) + 3 UIP facets.

6.3 Choosing between multiple properties in templates and multiple data templates

Theoretical introduction

Several objects with similar functions can be labelled with the same name, each bearing a different set of properties.

The environment (building) in which an object is located, the system it fits into or is connected to, and the function it performs have a significant influence on what properties and what property values need to be known for a given object. This contextual classification can quite clearly narrow down the number of properties that need to be worked with (classification context). Even when the number of properties is narrowed down by the classification system, it will still be large, as they take into account the needs of all project participants throughout the entire life cycle of the building.

Therefore, it is necessary to correctly determine for whom, in what time period and for what purpose the values of the properties are required, i.e. the principle of the “level of information needs” must be applied. In contrast to the first narrowing of the list of properties moderated by the classification

system, the task of the level of information needs is to make the work with information more efficient and clearer for specific project participants.

Motivation for change

In the past, the Agency took the approach that for a single construction element (note that there are properties linked to a construction element) there were as many templates as there were possible contexts in which it could occur, due to the impossibility of filtering properties according to the CCI classification.[4] This approach proved unsustainable as it led to a significant expansion in the number of templates.

Therefore, the intention is to apply the "one construction element = one data template" approach.[5]

In order for this approach to be applied successfully, both Writer and Browser must allow:

- Property filtering by facets stated in CCI + UIP
  Property filtering by CCI facets should be handled in the same way as filtering by UIP facets; the user interface design should be approached in such a way that CCI and UIP together give a single set of facets, with the caveat that it is not possible to say the order of their importance (all facets are equally important for filtering), but it is possible to specify a recommended order in which the user should enter them, in the direction CCI → UIP.

- Property filtering in a coherent way (see section 6.4 "*Non-coherence and coherence of facets*")

With this approach, it is possible to have all the properties of a given construction object (e.g. a valve) in one data template (which will then have a considerable number of properties); this list can be filtered efficiently and no additional templates need be created artificially.

6.4 Non-coherence and coherence of facets

Non-coherence of facets

The term "non-coherence" in this approach means that the system filters the properties according to the user's selection entered in the individual facets (applying a simple logical product between facets). However, a characteristic of this approach is that if a facet contains a number of properties of a given construction object (e.g. a valve), the user may receive more results than required in a single query.

> We will use two examples to illustrate the situation:
>
> Example from the construction industry: a valve (fitting) can be described by a number of properties – nominal clearance, nominal pressure, material properties, and, for instance, radiation resistance and safety. The system is to return the property "radiation resistance" as a filtering result only if the valve is to be located in a nuclear power plant. Our system is to return the property "safety" as a filtering result if the valve is to be used in a drinking water supply, etc. If the valve is to be used in a drinking water supply in a nuclear power plant, the system is to return both of these properties during filtering.

[4] Not to be confused with the possibility of filtering the templates themselves according to the CCI classification: here we are referring to the filtering of properties stored in the templates. The filtering therefore results in a list of properties.

[5] A new data template is created when it is necessary to describe a construction element or a structure with a different composition of properties, even if it differs by a single property compared to an already existing data template, and where its presence cannot be controlled by classification (e.g. technological variants of the construction element) or by the level of information needs.

Illustrative and imprecise parallel from outside the construction industry: you want to choose a computer mouse. The filter offers you several facets: size (S, M, L, XL), colour (B, W, R, G, B), number of buttons (1,2,3,4) and manufacturer (Microsoft, Logitech, Genius). Even using the independent approach, of course, you are able to set up the filter and select the mouse, because as a human you will use a series of different queries and keep querying until you are satisfied. But what if you were limited to being able to ask the system only once? Then you are able to specify your requirement, for example, as S-W-2-Microsoft, or {S,M}-{W,R}-{2,3,4}-{Microsoft, Genius}, but you are not able to specify in one query that you would only accept a white mouse from Genius, and a red mouse only from Microsoft, assuming that the red mouse is size M, while you would only accept a white mouse with 3 buttons. Note: the parallel is, of course, inexact, but hopefully it captures the theme of the interplay between the various facets.

Coherence of facets

The coherence of facets in filtering can be characterized by the fact that the valid values of each individual facet depend on the combination of the values of all other facets. In a single query, users receive precisely the results they required (see the examples above). Public users enter the query via Browser in exactly the same way as in the previous case, where we described the independence of facets, by selecting data in lists or trees of individual facets.

The difference in approach is not **how the filter is entered** (in Browser), but **how the data** (in Writer) in which the filter runs the search **is structured**. This requirement therefore has implications for both Writer (which must allow data to be entered in the correct structure, see the following sections) and Browser, which must be able to process this structure correctly.

6.5 Selecting facet values for filtering

This filtering is made up of 2 aspects –

- in Browser, the user selects the CCI context
  in doing so, they limit the properties in the template that are visible to them solely to those that match the context
- at the same time, in Browser, the user selects the UIP prerequisites: for which actor, use case, and milestone they need to determine the properties
  in doing so, they further limit the properties in the template that are visible to them solely to those that match this information need

The selection of facet values in Browser follows common conventions:

- users can select the values available in each facet. “Selection” means choosing 1 to n values in the enumeration.
- users can select multiple values in each facet
- for the values selected in each facet, the system considers their logical conjunction (OR)
- for the individual facets among themselves, the system considers their logical disjunction (AND) when filtering
- for user convenience, an exception is introduced that if users do not select any value in a given facet in the filter, they are considered to have selected all values.

For all user-selected values in one facet in the filter, their conjunction (OR) applies. Disjunction (AND) is applied between individual facets, between individual subfilters for CCI level 0 to 4 (for a component) and all UIPs.

6.6 Explanation of facet dependency

When describing facet dependency in the section 6.4 "*Non-coherence and coherence of facets*" we noted that "the difference in approach is not how the filter is entered, but **how the data** in which the filter runs the search **is structured**."

Now, we describe this data structure.

6.6.1 General description

As long as we were considering **non-coherent facets**, for each property it was sufficient to define lists of the values of individual facets. If this list matches the filter, the property is returned as required by the user.

Example:

For properties in a data template describing a built space (CCI 2), the data stored in Writer will have the following structure:

Property	CCI 0	CCI 1	Milestone	Use	Actor
Compressive strength class	ABC	AB	DSP	<any>	designer
Other property	…	…	…	…	…

With **coherent facets**, this changes: for each property, all combinations of the individual facets must be entered; the property will be returned as required if the filter matches any of these combinations.

Example:

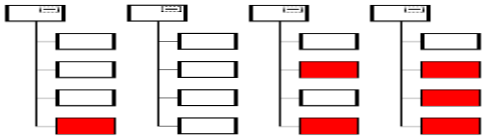
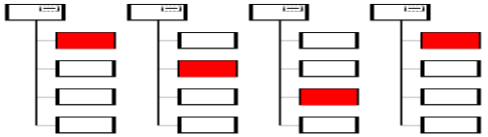
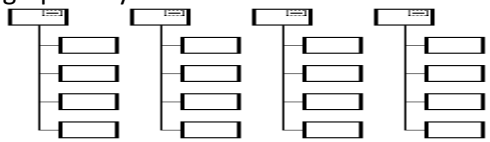
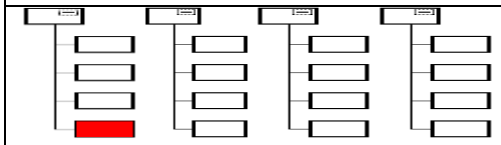
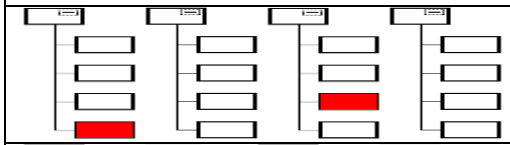
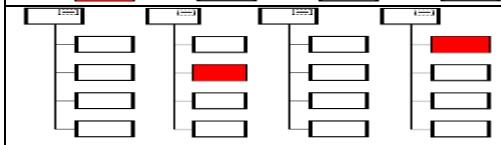
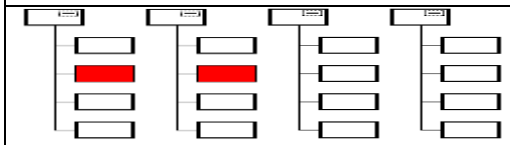
For properties in a data template describing a built space (CCI 2), the data stored in Writer will have the following structure:

Property	CCI 0	CCI 1	Milestone	Use	Actor	Note
Compressive strength class	ABC	AB	DSP	<any>	designer	*Let's call this Condition 1*
	AB	A	DPS	spatial coordination	<any>	*Let's call this Condition 2*
	…	…	…	…	…	*Let's call this Condition x*
Other property	…	…	…	…	…	

The table cells are references in the hierarchy of individual facets, to a generic element: the root of the hierarchy, any node or leaf. They need not be entered at all (in relation to UIP facets, though not CCI facets). Let's illustrate the expected behaviour of the system with examples.

6.6.2 Simple examples

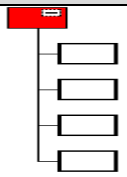
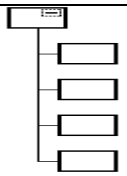
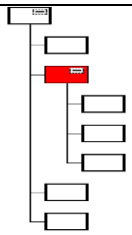
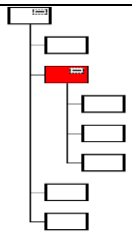
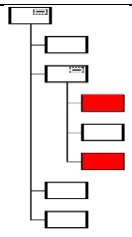
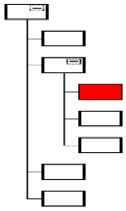
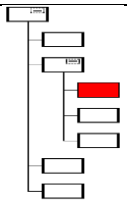
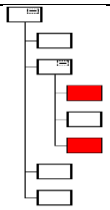
To give an example, let us dispense with the fact that the display conditions are entered in the form of a multiple selection of values for CCI0-n facets, milestones, uses, and actors. From a software perspective, there is generally a set of tree data structures defining an m-dimensional space (where m=n+3). For the sake of simplicity, only 4 facets (dimensions of m-dimensional space) are shown in the following example, and again, for simplicity, they are presented as flat lists (the hierarchy is degenerated into a one-level list):

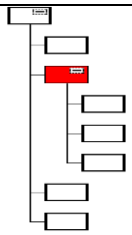
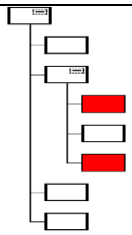
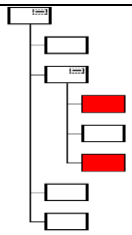
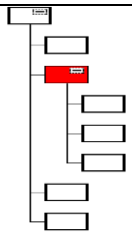
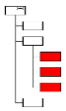
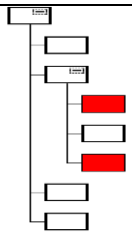
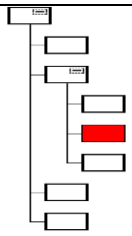
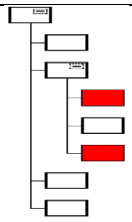
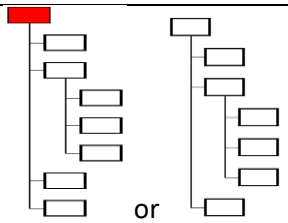
Facet hierarchy entered in Writer. The property references colour coded values	**Search criteria entered in Browser**	**Should the property be displayed? Why?**
The property is entered in the Writer application (see the table in the preceding paragraph for the example) by referring to the red-coloured elements of the individual trees: *(note: the conditions are organised in the manner indicated in the table in the preceding paragraph)* <u>*Condition 1*</u>  <u>Condition 2:</u> 	The user has not selected any filtering conditions (they are browsing DS, not filtering). We can express this graphically as: 	YES No condition is entered. **A value that is not entered is treated as a wildcard.**
		YES Conforms to Condition 1.
		NO Does not conform to any condition
		Conforms to Condition 2
		Does not conform to any condition

6.6.3 More complex examples

In 6.6.2 "*Simple examples*", we introduced the simplification that trees have only one floor and are therefore flat lists. We will now abandon this simplification, and the situation becomes curiously complicated. The aim of this section is to make sure that, even in this more complex case, we can specify under what conditions the property should be displayed and find a consensus on that.

We will go through the examples one by one and demonstrate them for one facet (any facet):

Facet hierarchy entered in Writer. The property references colour coded values	Search criteria entered in Browser	Should the property be displayed?
	Any. It is irrelevant what is entered.	YES Each filter entry in Browser corresponds to the fact that the property references the whole tree and therefore – according to this facet – should be found.
		YES This case is technically just the previous case expressed in a different way. From the perspective of the user, these cases are entirely analogous. The decision on how Writer implements this is entirely up to CoBuilder.
		YES The filter entry in Browser precisely matches the data entered in Writer
		YES The filter in Browser requires that the property in this facet reference the first leaf, and it does reference it.
		YES The filter in Browser requires that the property in this facet reference the first OR third leaf and it references the first leaf.

		YES The Browser filter requires that the property in this facet reference the first and third leaves. However, it references all sheets (on a legacy basis) – that is how it is entered in Writer.
		YES The legacy rules apply not only to data entered in Writer, but also to filter entries in Browser. Therefore, the filter basically looks like this:  The filter matches the data in the first and third requirements. One match alone would be enough for the property to be displayed (there are two matches here)
		NO Evident
	 or	YES These filter entries are equivalent. In the first case the legacy of the value occurs throughout the tree; in the second case the exception that an unselected value = the selection of all values is applied. The decision on how Browser implements this is entirely up to CoBuilder.

The examples indicate the algorithm that the property should be displayed (found, returned as the result of a filter operation) if, in the data entered in Writer:

- it references the given node directly, or
- it is contained anywhere in a subtree, or
- it is contained on the path to the root.

6.7 Alternative description

In order to supplement and elaborate on the information presented here, we also provide the following text describing this issue, written by another author (Jan Kolomazník):

Pre-setting specification of level of information need – a problem identified

Current known practice (e.g. CoBuilder) is pre-setting information requirements separately for each of the prerequisites (based on EN 17412-1) as follows:

Object O1	Purposes		Milestones		Actors	
Information In	P1	P2	M1	M2	A1	A2
I1	R	N	N	R	N	R
I2	R	R	R	N	R	R

R = requested, N = not requested

Example interpretation: For object O1 information I1 is requested for purpose P1 at milestone M2 and from actor A2. For object O1 information I2 is requested for purposes P1 and P2 at milestone M1 and from actors A1 and A2.

Problem: Information I2 is meant to be requested for purpose P1 only from actor A1 and for purpose P2 only from actor A2, both at milestone M1 – these combinations apparently can't be captured.

To capture such combinations a different table is needed, separately for each information:

Object O1	Purposes		Milestones		Actors	
Information I2	P1	P2	M1	M2	A1	A2
R	X		X		X	
N	X		X			X
N	X			X	X	
N	X			X		X
N		X	X		X	
R		X	X			X
N		X		X	X	
N		X		X		X

R = requested, N = not requested, X = combination marker

The table follows the following rules:

- exactly one of each prerequisite is marked in each line.

To calculate the number of combinations (nC; rows) the following formula seems to apply:

nC = nP x nM x nA

When 1 new prerequisite, e.g. a purpose, is added, it produces ÄC new combinations:

ÄC = 1 x nM x nA

To provide a relevant result when filtering according to a selection of prerequisites, the following rules need to apply:

- at least one of each prerequisite shall be selected;
- when none is selected, all apply (default value);
- when there are no values for selection for some of the prerequisites, an implicit value applies.

Interpretation of the result of filtering:

- when exactly one of each prerequisite is selected, the list applies to the selected combination;
- when multiple of any of the prerequisites are selected, the list applies collectively for all of them, again not capturing the distinct combinations.

Method of pre-setting the information requirements:

- by default, for no combination of prerequisites the information is marked as requested;
- by default, when a new prerequisite is added, for no new combinations the information is marked as requested;
- default settings can be manually overwritten by a choice to collectively select some or all combinations for which the information will be marked as requested.

7 Data templates and construction objects – how to proceed?

Plan for new CAS content

We are currently planning to create data templates for objects in a breakdown structure (as foreseen in EN 17412-1), namely CCI tables for construction components and technical systems (see https://cci-collaboration.org/ - DOWNLOAD - CCICore-02-...-...-ConstructionComponent-20221109 – sheets “CO Construction component” and “CT Technical system”) extended with some more precise custom object types identified as subclasses.

Both tables are based on definitions taken from international standards, namely IEC 81346-2:2019, table 3, and ISO 81346-12:2018, table A.2.

Issues with construction objects

Since the chosen breakdown structures follow standardised international definitions it seems redundant to search for existing (already approved) construction objects or to suggest potentially matching definitions from other sources.

It seems more logical to create and use a complete set of new construction objects following ISO/IEC 81346 series (RDS), but construction objects are currently a single flat list of items and naming conflicts would arise.

These naming conflicts could be resolved by applying some structured prefixes that would separate RDS based construction objects from other construction objects or by creating a separate named set of construction objects (or a combination of both).

Because the RDS breakdown structures are hierarchical, it would be practical to work with them in the form of trees, like it is now possible with classifications, ideally enriched with examples to cover more content from RDS.

NOTE: These issues became more important after seeing how IDS is compiled using Link where data template details are missing and construction objects combine some of their attributes instead. This lead to a finding that it is not practical to have a more generic construction object with several more precise data templates attached to it

How to proceed?

What would be your suggestion on how to deal with these issues and proceed?