D6.2 Ingest and Storage of 3D Objects in a Digital Preservation System

DURAARK

FP7 – ICT – Digital Preservation
Grant agreement No.: 600908

Date: 2015-01-31
Version 1.0
Document id. : duraark/2015/D.6.2/v1.0

Grant agreement number	:	600908
Project acronym	:	DURAARK
Project full title	:	Durable Architectural Knowledge
Project's website	:	www.duraark.eu
Partners	:	LUH – Gottfried Wilhelm Leibniz Universitaet Hannover (Coordinator) [DE] UBO – Rheinische Friedrich-Wilhelms-Universitaet Bonn [DE] FhA – Fraunhofer Austria Research GmbH [AT] TUE – Technische Universiteit Eindhoven [NL] CITA – Kunstakademiets Arkitektskole [DK] LTU – Lulea Tekniska Universitet [SE] Catenda – Catenda AS [NO]
Project instrument	:	EU FP7 Collaborative Project
Project thematic priority	:	Information and Communication Technologies (ICT) Digital Preservation
Project start date	:	2013-02-01
Project duration	:	36 months
Document number	:	duraark/2015/D.6.2
Title of document	:	D6.2 Ingest and Storage of 3D Objects in a Digital Preservation System
Deliverable type	:	Report
Contractual date of delivery	:	2015-01-31
Actual date of delivery	:	2015-01-31
Lead beneficiary	:	LUH
Author(s)	:	Michelle Lindlar `<michelle.lindlar@tib.uni-hannover.de>` (LUH), Michael Panitz `<michael.panitz@tib.uni-hannover.de>` (LUH), Ujwal Gadiraju `<gadiraju@l3s.de>` (LUH).
Responsible editor(s)	:	Michelle Lindlar `<michelle.lindlar@tib.uni-hannover.de>` (LUH).
Quality assessor(s)	:	Jakob Beetz <`J.Beetz@tue.nl`> (TUE), Martin Tamke `<martin.tamke@kadk.dk>` (CITA).
Approval of this deliverable	:	Stefan Dietze `<dietze@L3S.de>` (LUH) – Project Coordinator
Distribution	:	Public
Keywords list	:	OAIS integration, digital preservation, IFC, E57, metadata

Executive Summary

This deliverable defines the necessary steps for ingest and storage of 3D objects into an existing OAIS compliant digital preservation system. It discusses how the gaps, which were previously identified and presented in deliverable D6.6.1, have been addressed in the DURAARK project so far. Developed methods and tools will be run against the DURAARK test set. Lastly, the existing drafts of the metadata schemas *buildm* for descriptive information and *e57m* and *ifcm* as technical metadata schemas for E57 and IFC respectively, will be extended significantly and presented in a digital preservation context.

Table of Contents

1 Introduction

Digital preservation has reached a level of maturity, where the domain can rely on a number of existing standards, such as the OAIS reference model, and has a good understanding of how processes in the six OAIS entities - ingest, archival storage, administration, preservation planning and access - can be designed. As described in the previous WP6 deliverable D6.6.1 “Current state of 3D object digital preservation and gap analysis report”[1], these existing processes need to be adapted for each new content type on a bit, logical, semantic and organizational level. Previous work in the DURAARK project has put forth the following gaps, which are described in detail in D6.6.1 and shall only be summarized here:

Bit Preservation / Archival Storage: While no specific gaps were previously identified, the integration of the material type into the architecture of an existing digital preservation system has put forth discussion points regarding the structure at the Intellectual Entity (IE) and Archival Information Package (AIP) level of an archive. These issues are directly derived from the temporal and spatial dependencies of the digital objects as seen in their relation to the physical asset they describe. While not strictly being at the “bit level” of the object, a clear IE and AIP structure forms the basis of any other digital object and preservation layer and are at the heart of the archival storage entity. The findings on these issues are discussed in chapter 2.

Logical preservation: As the DURAARK consortium has chosen two open file formats for the material types point cloud scans and Building Information Models - namely E57 and IFC-SPF (Industry Foundation Classes - STEP Physical File Forma) - the gaps and further work on the logical preservation level are only focusing on these two encodings. The previously identified gaps pertaining to this layer were centered on four processes:

- *file format identification* with no information on the file formats being available in standard file format registries and no support through existing identification tools used by the community
- *technical metadata extraction* with no criterion and extraction tools available

[1] `http://duraark.eu/wp-content/uploads/2014/02/duraark_d6.6.1_final.pdf`

- *file format validation*, with the suitability of existing tools out of the buildingSMART community[2] / the libe57 reference implementation community[3] being unclear
- *risk analysis*, with no known in-depth analysis of the two file formats against potentially existing sustainability risks

The progress made in closing these gaps is discussed in chapter 3.

Semantic preservation: Semantic preservation is one of the least developed areas of digital preservation and the existing gaps are therefore not necessarily content specific, but fundamental ones. The only content / domain specific gap was indeed the lack of identified relevant knowledge base sources for 3D architectural data. Implementing methods to link and preserve changing semantic concepts, to monitor the used datasets for these changes and to calculate the impact of the changes in the data sets are indeed gaps which exist in a universal preservation landscape. These issues have been addressed by WP3 in the SDA+ (Semantic Digital Archive) prototype with its connected SDO (semantic digital observatory) layer. Chapter 4 will briefly summarize the completed and ongoing work in WP3 and discuss how it can be leveraged by existing digital preservation systems.

Metadata: Contextual and technical information about an intellectual entity form the backbone of any digital preservation process. While a number of existing descriptive sets for data relating architectural objects had been identified in D6.1, their suitability for plans and scans were unclear. On the level of technical metadata in a digital preservation context, no schemas for 3D scans or Building Information Models could be found. Furthermore, the feasibility of describing both types of objects in a single technical metadata set was unclear. As semantic preservation concepts pose a new area for digital preservation, the description of changing concepts over the course of time - the migration on the semantic representation level of the object's description - was identified as another area which needed to be explored further. Chapter 5 will describe the development and current status of the descriptive DURAARK metadata schema *buildm* and of the technical

[2]buildingSMART is the membership driven organization in charge of the IFC standards `http://www.buildingsmart.org/`
[3]libe57 offers a open source reference implementation and tools for the E57 file format `http://www.libe57.org/`

metadata sets *e57m* and *ifcm*. All schemas were introduced in D3.3.1[4], implemented in the first prototype as described in D2.4[5] and have been extended since. We built on this work by giving detailed element descriptions for all three schemas and by discussing the relevant connection of the technical metadata elements to a preservation environment. For *buildm*, we furthermore developed a mapping to other descriptive metadata schemas of the digital library / archival domain and are providing RDF and XML schemas in the Annex of this deliverable. For *e57m* and *ifcm*, XML schema definitions are also provided in the Annex. While XML still remains predominant metadata representation form in the digital library world, a slow shift towards RDF can be observed as the value of the semantic web is exploited more and more in the traditional digital library domain [13]. Delivering schemas for both - XML and RDF - shall bridge these two ends of the spectrum. As new methods and tools are developed throughout the entire course of the DURAARK project, the schemas are evolving in parallel to reflect new project results. Due to this, the status presented here is not a final version but reflects the current status of the schemas. However, all metadata sets presented here shall allow the user to understand information required of building representation information in a digital preservation context. While the definition of the metadata types are a prerequisite for an all-encompassing metadata wrapper such as METS[6], this identified gap is discussed in the integration assessment in chapter 6.

In addition to the groups of gaps described above, D6.6.1 identified gaps related to organizational roles in digital preservation. These are currently being analyzed in the use case studies taking place in WP7, in particular in D7.2 and D7.3. The outcome will form the basis for further work in connecting organizational requirements to digital preservation processes in the forthcoming WP6 task dealing with preservation planning. Those shortcomings will therefore be picked up in D6.3.

While chapters 2-5 discuss how the above mentioned gaps have been addressed in the project so far and show the results of applying the methods and tools against the DURRARK test set, chapter 6 discusses the integration possibilities of the DURAARK methods and tools. As the OAIS reference system is not a blueprint for a digital preservation system, many different forms of integration exist in organizations and their preservation workflows. Chapter 6 will list different integration forms, ranging from using the

[4] `http://duraark.eu/wp-content/uploads/2014/02/duraark_d3.3.1_final.pdf`
[5] `http://duraark.eu/wp-content/uploads/2014/08/DURAARK_D2_2_4.pdf`
[6] Metadata Encoding and Transmission Standard `http://www.loc.gov/standards/mets/`

DURAARK tools in their designed modularity to using the DURAARK integrated prototype. The discussion draws on experience with the integration into an OAIS compliant system so far and gives an outlook to the work which will be completed in forthcoming task 6.3 (M25-30), where the DURAARK pre-ingest workbench will act as an automatic depositor to the digital preservation system. Chapter 7 concludes this deliverable in discussing the impact and giving an outlook to further work of the project.

1.1 Methodology

Building on the state of the art study reported on in D6.1, the DURAARK consortium put efforts into closing the identified gaps. As part of work towards ingest and storage into a digital preservation system, the work which this deliverable reports on can be divided into three aspects with their own methodological background:

1. The tools and methods were used in experiments using the DURAARK test set[7], consisting of 117 IFC-SPF and 96 E57 files. The tests were run on regular desktop machines, accessing the objects in a local file system.
2. The first versions of the DURAARK metadata sets for IFC and E57, which were put forward in D3.3.1, were extended to meet the DURAARK requirements. For this deliverable, different versions were drafted and discussed within the consortium. This was amended by comparison to existing schemas, leading to a new version of the schemas for descripitive metadata - *buildm* - and technical metadata - *e57m* and *ifcm*.
3. Based on the results of the previous two points, the DURAARK tools and methods were analyzed for different integration options into digital preservation systems and workflows. This was amended by a desktop study and face-to-face discussions with preservation experts[8] to check feasibility in different preservation systems.

[7]The DURAARK data set was collected by WP7 and is reported on in D7.7.1. It has been extended slightly since the status reported on in D7.7.1. See D7.7.1 at: `http://duraark.eu/wp-content/uploads/2014/06/duraark_d7.7.1.pdf`

[8]e.g., in the context of conference attendance or working group participations

1.2 Risks

Risks associated with the process:

#	**Risk Description**	**Risk Assessment**	**Contingency Solution**	**Project Relevance**
1	The DURAARK test set is too homogeneous to serve as an adequate test set for the developed methods and tools.	**Impact**: High; **Probability**: Middle. As the test data stems from different producers with different requirements and different processes, a wide variety should be present.	DURAARK project partners either directly produce IFC and E57 files for research and/or collaborate with industry partners, who work with IFC and E57 files. If the current test set is too homogeneous, these resources will be tapped for more objects to be added to the data set.	High (Observed)
The problem occurred to a lesser degree with E57 files. While the DURAARK test set represents current practices in the domain well, the test set only included scans produced with FARO scanners. As a lack of export functionality in connection to the FARO SCENE software was detected, a second test set was used to verify that the problems are indeed only connected to this specific set up. For this, data examples from the libe57 site[9] were used. While these checks were used to verify the assumptions, the results presented in chapter 3 only describe the original DURAARK test set.				

[9]http://www.libe57.org/data.html

2	New aspects arise during the remainder of the project, which lead to necessary changes in the descriptive (*buildm*) and technical (*e57m* and *ifcm*) metadata sets.	**Impact:** Middle; **Probability:** High. As the methods and tools are in constant development throughout the remainder of the project, it is likely that new aspects will arise. However, thorough research into the use case and preservation background of the metadata sets have lead to a strong basis set. Additions and changes are expected to be minor ones.	Since the schemas have been presented and discussed in several cycles within the consortium, the structure is well known among all partners. New additions can be suggested by all partners and will be discussed within the consortium. There is a general consent, that slight modifications are to be expected and that the schemas are a work-in-progress for the remainder of the project. New versions will be incorporated into the remaining prototypes.	Middle (Observed)
Changes are proposed to the schema by various work packages on an ongoing basis. Proposed changes are discussed consortium wide and the decision to adapt the schema is always based on a consortium wide consensus.				

3	IFC-SPF and E57 will be considered not-suitable formats for design-to-construction processes or digital preservation by the respective communities.	**Impact:** High; **Probability:** Low. Over the course of the project so far a lot of work has been put into both, the evaluation of the suitability of E57 and IFC-SPF for preservation purposes (WP6) as well as the usage in the stakeholder communities (WP7). The probability of this risk is therefore very low.	The stakeholder communities will be monitored for new formats or indicators towards problems with the file formats. At the same time the DURAARK consortium will disseminate the project results to the stakeholder communities to present the usage and preservation suitability of the formats.	High (Not observed)

Table 1: **Risks identified and assessed.**

1.3 Technical Decisions

No technical decisions were made in the task associated with this deliverable. Instead, the work picked up on the following technical decisions made previously in the project:

- differentiation between file format identification, technical metadata extraction and file format validation, based on state-of-the-art processes in digital preservation, as described in D6.6.1
- differentiation between descriptive and technical metadata, based on state-of-the-art processes in digital preservation, as described in D6.6.1 and based on the first draft of metadata schema extensions presented in D3.3.1
- serializations of metadata in both, XML and RDF, as proposed in the aforementioned deliverables D6.6.1, D3.3.1 as well in the first integrated prototype described in deliverable D2.4
- processes on the observatory and archive levels of the Semantic Digital Archive (SDO/SDAS), as presented in deliverable D3.3[10]

The outcome of this work is integrated into the WP6 task 6.3, where the generated objects ingested into the existing digital preservation system hosted by TIB - the German National Library of Science and technology. The motivation for using Rosetta was described in deliverable D2.2.3[11].

[10]http://duraark.eu/wp-content/uploads/2014/08/DURAARK_D3_3_3.pdf
[11]http://duraark.eu/wp-content/uploads/2014/02/duraark_d2.2.3_final.pdf

2 Bit Preservation and Archival Storage

The following chapter briefly presents the implementation options and choices regarding the structuring of information objects from the PREMIS-based intellectual entity model as well as from the OAIS-based information package view. It discusses the feasibility for 3D architectural objects in general and the DURAARK objects in particular.

2.1 Intellectual Entity

The concept of the Intellectual Entity (IE) has been driven by the PREMIS standard, in which it is one of the five entities. The PREMIS standard describes the IE as "a set of content that is considered a single intellectual unit for purposes of management and description" [20]. The standard carries on to say that intellectual entities can be nested and that each intellectual entity may have multiple digital representations, with a representation being a "complete and reasonable rendition of the Intellectual entity". Each representation may consist of one or more files. Each file consists of a bitstream - if the bitstream itself can be transformed into a meaningful file by only using minimal transformation on the decoding level, e.g., in form of extracting the embedded png 2D image sections of an E57 file, the bitstream is called a filestream. [20]

Figure 1 shows an example for an Intellectual Entitiy in the context of architectural data. Here, the intellectual unit is a plan, which was created at a specific point in time and describes a section of a building. Separate representations of the plan exist. The first representation is the proprietary AutoCAD object, consisting of the .dwg file as well as a font file .ttf and color information for plotting in a .ctb file. Besides the native and proprietary AutoCAD file, an export into the Drawing Exchange Format (.dxf) as well as a PDF/E representation for preview purposes may be available. While all representations may be derived from the native AutoCAD file, they all serve representation purposes in their own right - i.e., the dxf representation as a 2D exchange format supported by almost all CAD packages and the PDF/E representation as an easily accessible preview rendition. Since thumbnails are stored in a reserved section within the DXF file, they could be captured and described at the bitstream level, if desired.

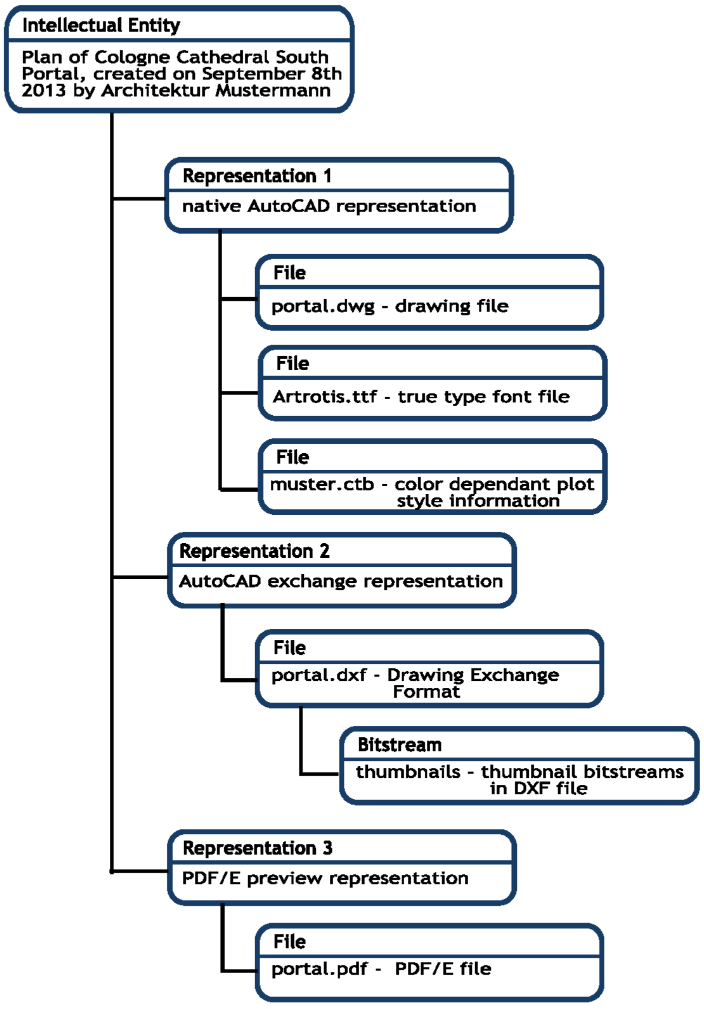

Intellectual Entity
Plan of Cologne Cathedral South Portal, created on September 8th 2013 by Architektur Mustermann
Representation 1
native AutoCAD representation
File
portal.dwg - drawing file
File
Artrotis.ttf - true type font file
File
muster.ctb - color dependant plot style information
Representation 2
AutoCAD exchange representation
File
portal.dxf - Drawing Exchange Format
Bitstream
thumbnails - thumbnail bitstreams in DXF file
Representation 3
PDF/E preview representation
File
portal.pdf - PDF/E file

Figure 1: Description of non-nested IE structure for a 3D plan with different representations

When putting the IE in Figure 1 in relation to the physical asset or building it describes, the IE becomes a representation in itself with temporal and spatial dependencies. A different approach to the simple IE structure presented in Figure 1 is a nested intellectual entity structure, where the top-level IE is a description of the physical asset itself and all plans, scans or other digital objects representing the asset are sub-IEs. These sub-IEs describe the physical asset, either partially or in its entirety, as it was - or is envisioned to be as in the case of future plans - at a given point in time. Such a nested-IE structure is shown in Figure 2.

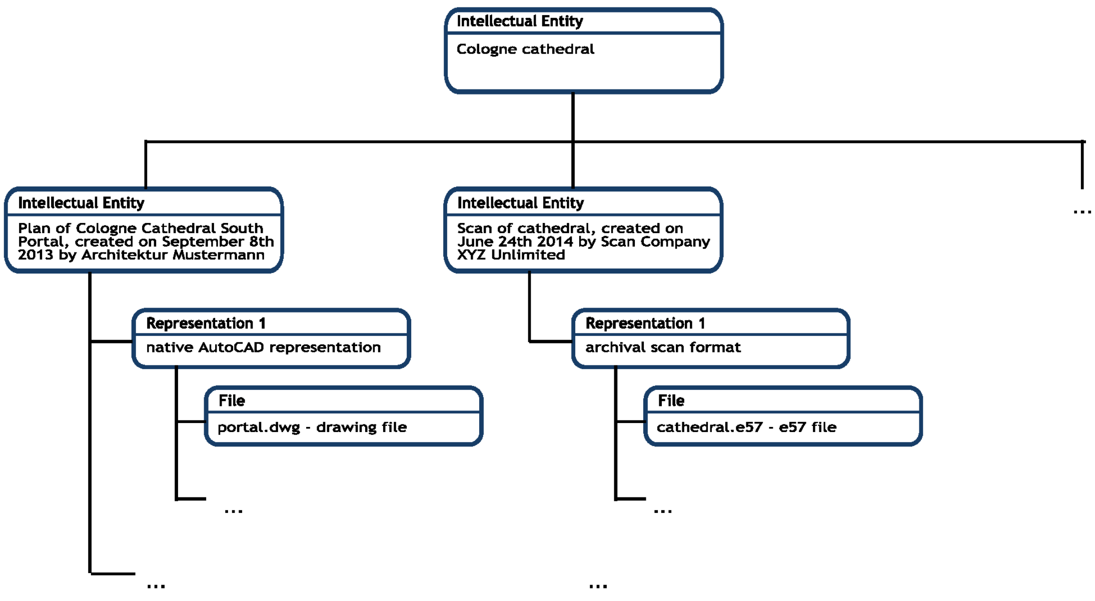

Intellectual Entity
Cologne cathedral
Intellectual Entity
Plan of Cologne Cathedral South Portal, created on September 8th 2013 by Architektur Mustermann
Intellectual Entity
Scan of cathedral, created on June 24th 2014 by Scan Company XYZ Unlimited
...
Representation 1
native AutoCAD representation
Representation 1
archival scan format
File
portal.dwg - drawing file
File
cathedral.e57 - e57 file
...
...
...
...

Figure 2: Description of nested IE structure with the physical asset at the top-level and a 3D plan and 3D scan as sub-entities

2.2 Information Packages

While all OAIS-compliant digital preservation systems package the preservation objects - consisting of the information object, the content information and the preservation information - as Submission Information Packages (SIP), Archival Information Packages (AIP) and Dissemination Information Packages (DIP), the make-up and internal structure of the packages may differ greatly. [6]

Submission Information Packages are typically tailored towards the requirements of the producer and transfer processes. Here, it is important to keep in mind that the archive may restructure the SIP by, e.g., creating multiple AIPs from a single SIP which contained several files. The EU-funded E-ARK[12] project is currently analyzing the requirements which archives have regarding exchange and transfer processes of information packages.

[12]http://www.eark-project.com/

A first analysis of SIPs from European archives of various sizes put forth that while package structures are most often influenced by METS[13], PREMIS[14] and EAD[15], a wide variety of logical structures exist [8].

2.2.1 Archival Information Packages

Implementation choices within OAIS compliant archives may follow either a physical or a logical AIP structure. In physical AIPs all components are merged into one single physical file. This is either achieved by packaging content and preservation information alongside the digital object(s) into an archive format such as .tar or .zip containers or by including all content and preservation information in a single XML wrapper format which furthermore includes the digital objects as embedded base64 encoded binary data. While the single XML file is not feasible for large data objects such as 3D scans, the bitstream robustness needs to be considered when choosing container formats such as tar. Especially for files with transparent data streams, such as ASCII encoded files, or self-controlling mechanisms such as CRC checks within the E57 file, a compression of the bitstream is a costly trade-off. Smaller file size and a self-contained "file" come at the expense of loss in resilience against bit-flips. The logical structure, on the other hand, allows for an arbitrary location of the digital objects as long as a stable linking can be ensured. This allows for easy preservation of complex hierarchies of AIPs and for rule-based decisions, e.g., to push less-frequently used objects to offline-storage devices. The trade-off here is between flexibility and transfer - while any form of structuring is possible, the transfer of AIPs to another archive, or even to a different infrastructure, will always require pre-processing steps where the information packages are assembled from their different storage locations.

The OAIS describes two different types of archival information packages (AIPs) which can contain multiple information objects: the Archival Information Unit (AIU) and the Archive Information Collection (AIC). While the AIU contains different information objects, which are not broken down into separate AIPs and are described through a single content information instance and a single preservation description object, the AIC contains separate AIPs with their corresponding content information and preservation

[13]Metadata Encoding & Transmission Standard - `http://www.loc.gov/standards/mets/`
[14]Preservation Metadata - `http://www.loc.gov/standards/premis/`
[15]Encoded ARchival Description - `http://www.loc.gov/ead/`

description objects. Put into the context of the intellectual entity, it can be said that a simple IE structure, as shown in Figure 1, is an AIU, while a nested IE structure, as shown in Figure 2, is an AIC. Both AIP structures support access, however, AICs allow for an internal organization of the sub-AIP in a thematic hierarchy, which allows for a higher degree of flexibility in retrieval and access scenarios on the consumer side. The nested IE structure which was chosen in the DURAARK context leads to the organization of AIPs as AICs [6].

3 Logical Preservation

The gaps associated with the logical preservation level covered four different processes: file format identification, technical metadata extraction, file format validation and risk extraction. Of these processes, technical metadata and risk extraction are closely related, as risks associated with the file format's technical make-up or contextual content of the digital object need to be extractable to allow for scalable detection. Due to this, these two processes are covered in section 3.2, which presents extracted results for the DURAARK test set and discusses some problematic values.

3.1 File Format Identification

File format identification is a crucial part of any digital preservation strategy, as an object's logical structure is the starting point and potential benchmark for any preservation action. File format identification is typically an automated process, where only unidentifiable results may be treated manually. As described in Deliverable D6.6.1., standard digital preservation file identification tools such as DROID[16] or the Unix file utility may use a combination of intrinsic characteristics such as format signature patterns and extrinsic characteristics such as file extension information to derive identification results. Figure 3 shows a typical file format identification workflow in a digital preservation system. The output of the identification process typically describes the file format through its PUID (PRONOM Unique Identifier) and/or the file format in clear text, e.g. "fmt/20 PDF 1.6".

[16]http://digital-preservation.github.io/droid/

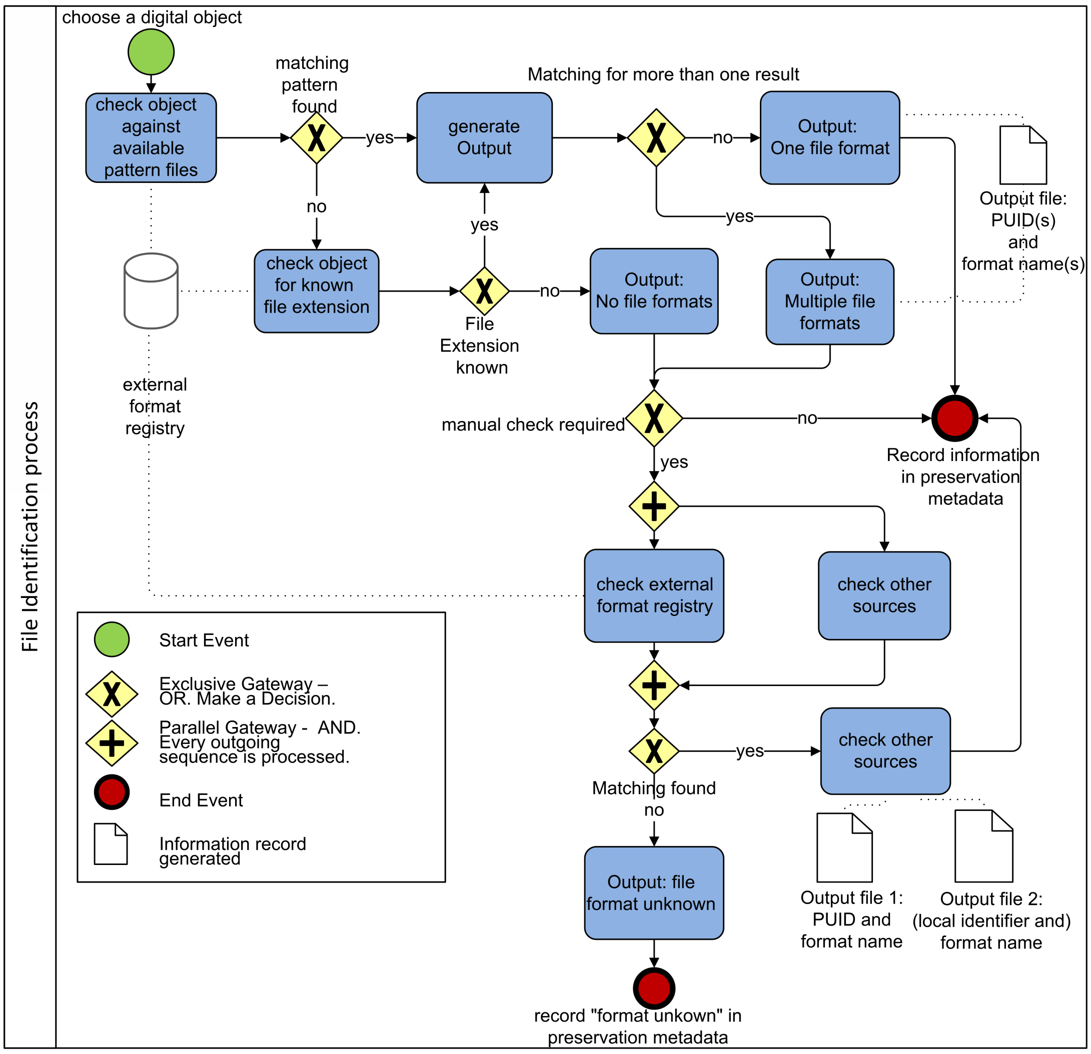

File Identification process
choose a digital object
check object against available pattern files
matching pattern found
yes
no
generate Output
Matching for more than one result
no
Output: One file format
yes
Output file: PUID(s) and format name(s)
check object for known file extension
File Extension known
no
Output: No file formats
Output: Multiple file formats
external format registry
manual check required
no
yes
Record information in preservation metadata
check external format registry
check other sources
Matching found
yes
no
check other sources
Output file 1: PUID and format name
Output file 2: (local identifier and) format name
Output: file format unknown
record "format unkown" in preservation metadata
Start Event
Exclusive Gateway – OR. Make a Decision.
Parallel Gateway - AND. Every outgoing sequence is processed.
End Event
Information record generated

Figure 3: Typical file format identification workflow in a digital preservation environment. File format identification tools may have no, one or multiple tentative hits as their output. The manual check process is optional and not found in every digital preservation setting.

Signature patterns of identification tools are often linked to registry information. These file format information knowledge bases, such as PRONOM[17], UDFR[18] or the Archiveteam's file format site[19] may furthermore be used for manual identification processes in the problematic objects as well as for preservation planning information. In closing the file format identification-related gaps, the two file formats E57 (fmt/643) and IFC-SPF (fmt/659)

[17]http://apps.nationalarchives.gov.uk/PRONOM
[18]http://www.udfr.org/
[19]http://fileformats.archiveteam.org

were in a first step registered in the file format registry PRONOM. In addition to IFC-SPF, IFCXML (fmt/663) has been included in the registry and the STEP physical file format is currently a pending candidate to be included in the PRONOM registry. In a second step, DROID signature patterns were written for the two formats. By using DROID signature patterns, at least three identification tools which are frequently used in the digital preservation domain are supported: DROID, fido[20], and the relatively new Siegfried[21] development. The signature patterns use one or more intrinsic characteristic to identify the file format. The E57 signature pattern looks for the "ASTM-E57" (byte sequence: Absolute `BOF 41 53 54 4D 2D 45 35 37`) sequence which is per ASTM E57 standard the mandatory beginning of every E57 file [3].

Signature patterns for binary formats are often more straight-forward than for textual formats. For plain text formats, such as IFC-SPF, the question "what (of the options) is the file format" often arises. At the encoding level, the digital object is ASCII text, at the format-family level, IFC-SPF is a STEP Physical file, fully conformant to ISO 10303-21. However, IFC itself is not just a STEP application profile, but instead a standard in itself - ISO 16739:2013 which specifies "an exchange file format for Building Information Model (BIM)"[17][22], thus presenting a third file format layer. A last layer exists in form of the schema version, i.e., IFC2x3, IFC4, etc. PRONOM addresses this in two ways: classification groups such as Text (Structured), Text (Mark-up) or Image (Vector) allow for a basic file format classification and relationships such as "has lower priority than", "has priority over" or "is superclass of" allow to describe the relationships between different formats / PUIDs. In line with this, a relationship between IFC-SPF and the STEP physical file format shall be described within PRONOM. While binary formats are most often described on a file format version level, i.e., with separate PUIDs for PDF 1.6, 1.7, 1.8 and so forth, structured or mark-up text objects are only rarely captured on the schema level. The Geography Markup Language, for example, is captured as PUID x-fmt/227, however, no separate PUIDs exist for the different schematic versions (current status 3.3)[23].

The common denominator of all STEP physical files is the beginning of the file (BOF)

[20]`https://github.com/openpreserve/fido`

[21]`http://openpreservation.org/knowledge/blogs/2014/09/27/siegfried-pronom-based-file-format-identification-tool/`

[22]At this level, three format variants exist: IFC-SPF, IFCXML and IFCZIP.

[23]`http://www.opengeospatial.org/standards/gml`

with “ISO-10303-21;” (byte sequence: absolute `BOF: 49 53 4F 2D 31 30 33 30 33 2D 32 31 3B`). As the ISO standard explicitly ignores line breaks, they can in theory occur at any point in the file. While we have observed different forms of line breaks in the objects of the test set, these did not occur within keywords or even data entities. We therefore did not include the option of different linebreaks after every binary value in the patterns. As opposed to IFC files with the .ifc extension, most application protocols use .stp as the file format extension.

Besides the different extension, one other intrinsic criteria is used to identify IFC-SPF files: the “FILE_SCHEMA’((IFC” substring, which can be found at a variable position within the header. As no regulation regarding spacing and linebreaks exists in the standard, different variants are possible. In the test corpus we have found two variants of the notation - one as stated above and one with an extra space before the first parenthesis: “FILE_SCHEMA ((’IFC”. The developed pattern recognizes both variants. Tests with larger test corpora will have to prove if other variants exist and the pattern needs to be adapted, if necessary. As per the “Implementation Guide for IFC Header Section” [15], the attribute of the file_schema entity shall be a string consisting of the name of the schema in upper case letters. Existing IFC schemas have been the now deprecated IFC1.0, IFC1.5, IFC1.5.1 and IFC2.0 as well as the still operative IFC2x, IFC2x-Add-1, IFC2x2, IFC2x2-Add-1, IFC2x3, IFC2x3-TC1, IFC4 and IFC4-Add-1. IFC5 is currently in the planning stage[24]. The naming convention for the schema has therefore followed a more or less intuitive numbering pattern.

The DURAARK test set was identified using the process described in Figure 3 and using DROID v6.1.3 and fido v1.3.1, both with a slightly modified signature file v79[25]. The two tools reached the same result: out of 96 E57 files 95 were identifiable via pattern and 1 via extension, out of 117 IFC files 116 were identifiable via pattern and 1 was identified as a zip file per pattern with a file extension mismatch to IFC. Closer investigation proved this to be correct - the file was actually an .ifczip, which is a regular .ifc file in a PKzip 2.04g container, as per buildigmsart definition of .ifczip[26]. Here, the file format identification tools acted as expected - since no separate pattern for ifczip exists, the objects were

[24]For a regularly updated list of IFC releases, see `http://www.buildingsmart-tech.org/specifications/ifc-releases`

[25]The modified file contained slight updates to a previously submitted pattern to TNA, addressing problems with line breaks in IFC file format identification

[26]`http://www.buildingsmart-tech.org/specifications/ifc-overview`

identified as zip containers (x-fmt/263) and, if recursive identification through containers was activated, the packaged file identified correctly as an ifc by pattern. Since in a pattern based identification process ifczip objects will always be identified as the PKzip containers that they are, archives may decide to automatically extract the information. In this case, an ifzip object entering an archive will always result in an IFC-SPF file in the AIP. The advantage is a direct transparency to file format analysis tools, such as the IFC metadata extraction tool, as well as higher robustness to bit flips due to the uncompressed storage of the object. Closer investigation of the one failed pattern identification for E57 turned out to be a corrupt E57 file, which could not be rendered and had been damaged during transfer to the FTP server. It was replaced with a backup copy and the identification process was rerun on this file, leading to successful identification by pattern. After the analysis of the two problematic files which were only identifiable via extension, a 100% identification rate by signature pattern could be reached for the DURAARK test corpus.

3.2 Technical Metadata Extraction

While the current state of the technical metadata sets *e57m* and *ifcm* is described in chapter 5, this section presents overarching criteria for technical metadata and the associated extraction process and presents and discusses the results of the extraction process for the test gset.

Technical metadata is often considered the first line of defense in maintaining access to a digital object over the long-term [21]. It captures information required to understand the creation process and metrics of the object to allow for access even if a rendering application on a current machine / operating system is no longer available for the object. Technical metadata is also used in risk-assessment processes, either when comparing the technical criteria of a digital object against an institution's digital preservation policy which, e.g., states that no compressed objects may be archived, or when comparing files against known global risks, such as, e.g., faulty export routines for an archival format in a specific creation software. Lastly, it allows to verify the success of preservation action such as migration, by e.g. comparing pre-migration to post-migration criteria such as resolution [25]. The current technical metadata sets for E57 and IFC are presented in chapter 5. Capturing technical metadata is therefore an essential process in digital preservation. Doing so manually, however, is a time-consuming process and usually only possible for limited information sets such as the creation process [21]. The solution for this problem is automation through metadata extractors.

The functionality and technical basis of the E57 and IFC metadata extractors were introduced and presented in D2.4. While the E57 extractor[27] is a DURAARK development based on libE57 and already extracts a wide variety of parameters from the digital objects, the DURAARK IFC extractor[28] is an ongoing development, with new parameters being added throughout the project. This is mainly due to the high level of semantic richness found in IFC objects as opposed to a limited information width found in E57 objects. In identifying technical metadata candidates, we focused on the following areas, in-line with common recommendations and practices for other content types [25] [21]:

Creation Process
Information about the creation process includes details about creation time and author as well as about hardware and software used during the creation process. For E57 and IFC alike this includes creation date, author and file format version, as well as the schema version for IFC. Information relative to the E57 creation process furthermore includes the scanner make and model as well as the accompanying firmware version, the software used to process the object as well as environmental influences during the scanning process, such as the ambient temperature or humidity which may influence the scanning outcome for outside captures. For IFC, software information about the creation process can furthermore be divided into the native software in which the original BIM object was designed, the software used to export the native object into the IFC object and the platform on which this was conducted as well as the current “owning software” of the object. It needs to be noted that this information is not exhaustive, as often several IFC exporter plug-ins exist for a CAD tool and as configuration decisions of the export workflow have direct implications on the outcome[29].

Parametric information
Parametric information describes the system in which the data is to be interpreted, such as measurement scales. The possible values for the respective elements can usually be

[27]`https://github.com/DURAARK/e57Extract`

[28]`https://github.com/DURAARK/pyIfcExtract`

[29]See for example the information which Revit provides on the webpage regarding exporting to IFC: `http://knowledge.autodesk.com/support/revit-products/learn-explore/caas/CloudHelp/cloudhelp/2015/ENU/Revit-DocumentPresent/files/GUID-6EB68CEC-6C17-4B16-A509-30537F666C1F-htm.html`

described through fixed sets. For E57 this includes information such as how the points are organized and which coordinate system is used (spherical or cartesian). For IFC this includes information about units such as length, area or monetary units.

Content extent
Content extent information describes the digital object in its information width and depth. This includes information on how large the information extent is as, e.g., described through number of pages or records, as well as how much detail is captured as, e.g., described through resolution. For E57 this includes information such as the number of 3D scans and 2D images which are included in the digital object as well as information about the bounding region of the respective coordinate system. For IFC this includes information such as the number of floors, the total square meters or the number of entities used in a model. Here, it needs to be noted that criteria such as the number of floors or square meters may be considered descriptive instead of technical metadata - within the DURAARK project it was decided that it is indeed both. A comparison can be made to technical criteria such as natural language a textual object is written in (e.g., English) [24] or source information in still images [1] as those criteria are as well information carrying features but can also be used to described the content extent. The content extend in form of floors, square meters or numbers of certain entities becomes relevant not only in automated comparison of objects post-migration, but also in measuring the correct display of the digital object when, for example, testing a new rendering software.

3.2.1 Results for E57 Extraction

E57 creation process
Table 2 presents a distribution of the scans of the test set by their creation date. The creation date is based on the creation_datetime values in the root section of the E57 object. However, it needs to be noted that this is the date at which the E57 file was exported from the processing software, e.g., FARO SCENE. It is not the date at which the scanning was conducted - as per standard the scanning date is to be captured in the acquisition_start and acquisition_end elements of the scans. However, for all E57 objects in the test set the acquisition_start and acquisition_end elements were set at 1980-01-06T00:00:00. Further investigation put forth that if the acquisition_start and end values are empty, either because the scanner did not capture them or because the processing software did not

export them, the libE57 reference implementation instantiates the acquisition_start und acquisition_end elements with this value, if no other value is present [30].

Date	**Number of objects**
May 2013	1
June 2013	2
August 2013	2
September 2013	1
October 2013	1
November 2013	54
March 2014	3
August 2014	22
September 2014	9
December 2014	1[31]

Table 2: Distribution of E57 test dataset objects by creation date

Regarding hardware and software, all but 8 objects had extractable information, with 5 out of the 8 - coming from the same producer - having vendor information present, while the remaining 3 objects, coming from different producer, had no hardware and software information whatsoever. It is unknown if the information was deleted deliberately or simply not included by the scanner or the processing software. One file of these files included information on the processing software in the description field ("this file is generated by Autodesk ReCap"). All of the remaining 88 objects were created using FARO Focus 3D S 120 scanners with iQLib firmware. As far as the processing software is concerned, only the version and not the name is captured. While for the FARO Focus 3D scanner the software can be assumed as FARO SCENE[32] software, the lack of processing software information may be more problematic for other scanner vendors or for open output formats which are supported by a multitude of processing software. While information about hard- and software may be captured, E57 has no explicit fields for information about the digital object's creator.

The investigation showed that information can be either not captured at the point of scanning, as the scanner may not support certain features, but may also be lost at the

[30]GPS week counter 0 starts on January 6th 1980

[32]This is based on the software version information extracted, all pointing to versions between numbering sequences 5.0.0.28079 and 5.1.1.30791, which are FARO SCENE versions common at the point of creation

point of E57 generation by the processing software. Along those lines, FARO SCENE claims to export the following information to the E57 file [11]:

- xyz-coordinates
- RGB and/or intensity values
- corresponding index of row and column
- meta information like scan name, UID, software version for export

A full list of the "meta information" is not given, while it is clearly stated that SCENE does not support the export of 2D images into E57.[11]

Due to the limited export functionalities of FARO SCENE, a lot of information which was most likely present in the raw scan output is not available in the E57 file anymore. This includes all weather data - while the FARO Focus 3D scanner is capable of capturing information about the atmospheric pressure, the SCENE software does not export it to the E57 file. However, all of our test data extracted the same value for temperature, relative humidity and atmospheric pressure (3.4028234663852886e+038). Further investigation put forth that this is another value created by the libE57 reference implementation libraries, which instantiates the value for the element with the maximum possible value for float / double [11] [10] [3]. The value can therefore be neglected, as it is faulty.

E57 parametric information

The E57 standard contains a coordinate metadata field, which shall describe the Coordinate Reference System using the well-known text (WKT) specification as described in the Open Geospatial Consortiums's (OGC) specification for Coordinate Transformation Services. The WKT allows the 3D and 2D data stored in the file to be referenced in a standardized coordinate reference system [3]. However, the element is not filled in any of test data set's objects. It is unclear if this is due to the information not being captured by the scanner or due to FARO Scene or other processing software not exporting the value.

E57 content extent

As mentioned above, FARO SCENE does not export 2D images, and accordingly, none can be found in the test data. The number of scans included in the E57 objects range between 0-139. The object with 0 scans was investigated further and turned out to be an empty, but valid E57 file. Since the header information and framework was parsed correctly, it passed file format identification and was not noticed prior to the extraction process. About 30% of the available E57 files in the test set consist of 1 scan, 22% consist of more than 20 scans - the information width found in the test set can be hence considered rather heterogeneous. No object in the test set included original_guid information and it is unclear if this is due to FARO SCENE not having exported this value or due to no scans having been merged as part of processing. An indicator for information depth is furthermore color, which can be divided between the color range for RGB channels, as well as the color intensity range. The test set of 96 scans included 47 files with color information (RGB range >0). In all cases the 0-255 scale was used. For intensity, only one object used a full 0-255 scale, all other objects had a 0-0 or 0-1 scale, indicating no meaningful use of color intensity.

3.2.2 Results for IFC Extraction

Creation process
Table 3 presents a distribution of the IFC objects by their creation year. Compared to the E57, the creation dates go further back in time - mostly due to the fact that IFC as a file format has been in use for much longer than E57, which was only adopted in 2011. However, besides the longer standing tradition of creating IFC objects, the availability of older files may also be considered a sign for the archival value and processes behind the objects. As pointed out in deliverable D7.7.1, the use of scans is relatively new to the building profession. Until now, these objects are mainly created and used by land surveyors and construction companies, who create 2D and 3D objects for stakeholders as a business model - hence having little interest in long-term archival of these objects. When scans do enter the building profession, they are currently still largely seen as an auxiliary file type in the BIM generation process.
The creation date for the evaluation was extracted from the time_stamp attribute in the FILE_NAME element of the IFC header. All objects in the IFC dataset contained a time_stamp - according to the STEP Header definition, the time_stamp shall correspond to the ISO 8601 date format convention [15]. However, 3 sample files of the dataset contained a non-ISO 8601 conformant time_stamp (in this case, DD/MM/YYYY HH:MM:SS). All 3 files came from the same producer and were created in 2012 with Graphisoft's ArchiCAD-64 16.0.0 with the "preProc - EDM 5.0 preprocessor". A search for objects created with the same application and preprocessor showed a correctly formed

time_stamp[33], it can therefore be assumed that the problem is not a generic one to the creating software but possibly associated with the producer's machine or local settings. For IFC objects, the schema is relevant information, as every schema introduces updates and changes to the specification.

Date	**Creation Year**	**Schema**
2003	4	IFC2x2
2004	1	IFC2x2
2005	2	IFC2x2
2006	5	IFC2x2
2007	5	IFC2x2
2008	4	1 object IFC2x2, 4 objects IFC2x3
2009	7	IFC2x3
2010	10	IFC2x3
2011	22	IFC2x3
2012	23	IFC2x3
2013	32	IFC2x3
2014	2	IFC2x3

Table 3: Distribution of IFC test dataset objects by creation year and schema version

Table 3 shows the extracted schema and creation year information for the DURAARK IFC test set. While the IFC2x3 schema was formally introduced in February 2006[34], the earliest "real-world" objects contained in the test data set were created in 2007. This is inline with what we are seeing for IFC4 today - while the new schema was released in March 2013, it has not been adopted by software vendors and/or consumers yet and is not represented in the DURAARK dataset. Notation of the schema found did not include information about the addendums, e.g., IFC2x3-TC1. It is uncertain if this is due to the TC1 version of the IFC2x3 not being implemented in the respective software or due to

[33]see, for example: `http://www.deltafaucet.com/media/CAD/1159LF\%20Trinsic\%20Single\%20Handle\%20Kitchen\%20Faucet.txt`

[34]For IFC schema release information, see `http://www.buildingsmart-tech.org/specifications/ifc-releases`

the applications not differentiating between major schema versions and addendums when writing the schema into the header[35]

Further information on the provenance of the file include the name of the object, the author, the organization and information on who authorized the transfer process. An evaluation of these values in the dataset showed what seems to be the weakness of most IFC generation processes - the values are filled automatically from standard values of the software implementation, such as the software vendor for “organization”, which should instead describe the organization that the author is associated with, or the user name value for the author, which often leads to meaningless information such as initials or values like “anonymous user”. Figure 4 gives a brief overview of the percentages of meaningful data found for the aforementioned values within the test set. Contrary to the provenance information, creating application information is usually well documented, however, not always in a structured manner. Following the STEP specification, the IFC header differentiates between preprocessor and originating system, but not all exporting software use these values the same. Nevertheless, the creating application could be determined for 87% of the objects, with over half of them being created by Autodesk Revit[36].

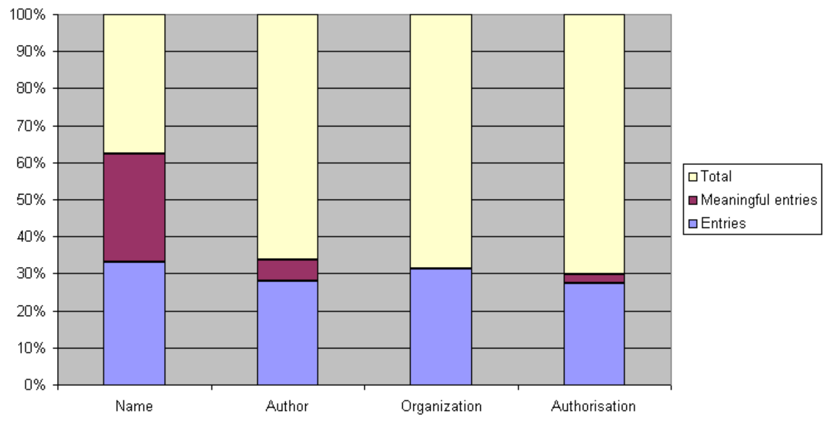


Figure 4: Meaningful *Name* entries are information other than placeholder values like “project number” or “xxx”; meaningful *Author* entries consist of a first and last name, meaningful *Organization* entries do not include the software vendor as an organization, meaningful *Authorisation* values include a first and last name or an organization.

[35]An exception are the IFC2x2 files, for which the schema is stated as “IFC2x2_FINAL”. As “final” does not describe a known addendum, it is unsure what the suffix means.

[36]Various versions of Revit were used. Other frequently used software included DDS CAD and Nemetschek Vectorworks

In addition to the creating application information taken out of the header, the IFC schema itself has an IfcApplication entity, which includes other software components which were involved in creating the BIM file.

Besides the poor implementation of IFC by different software vendors, other reasons for the lack of meaningful data in the test set exist. The data may simply be lacking because it was not deemed necessary - either because the objects stem from a research / education background, where the provenance was of no relevance - or because the business process connected to the object within a company may require the respective information to be stored elsewhere, e.g., in a database system such as bimsync[37], which keeps track of the authors and organizations.

Parametric information

In the IfcSiUnit entity, each IFC file defines the units which apply to different measures used in the object. Units found in the test set included those for area, length, electric current, electric resistance, electric voltage, energy, force, frequency, luminous flux, luminous intensity, mass, volume, pressure, plane angle, solid angle, power, thermodynamic temperature and time. Applied units differed for length and mass, but also for factors such as luminous intensity with possible values such as Lumen or Candela.

Within an IFC file, one or several defined representation contexts (IfcGeometricRepresentationContext) allow for a classification of shapes regarding context (either model or plan with according subgroups, see Figure 5), the numeric precision which is applicable to the geometric representations and the TrueNorth attribute or an offset from the project coordinate system. The precision describes the maximum distance between two points to be still considered identical. In the surveyed test data, precision values ranged between 1.0E-9 and 1.0E-5.

[37]https://bimsync.com/

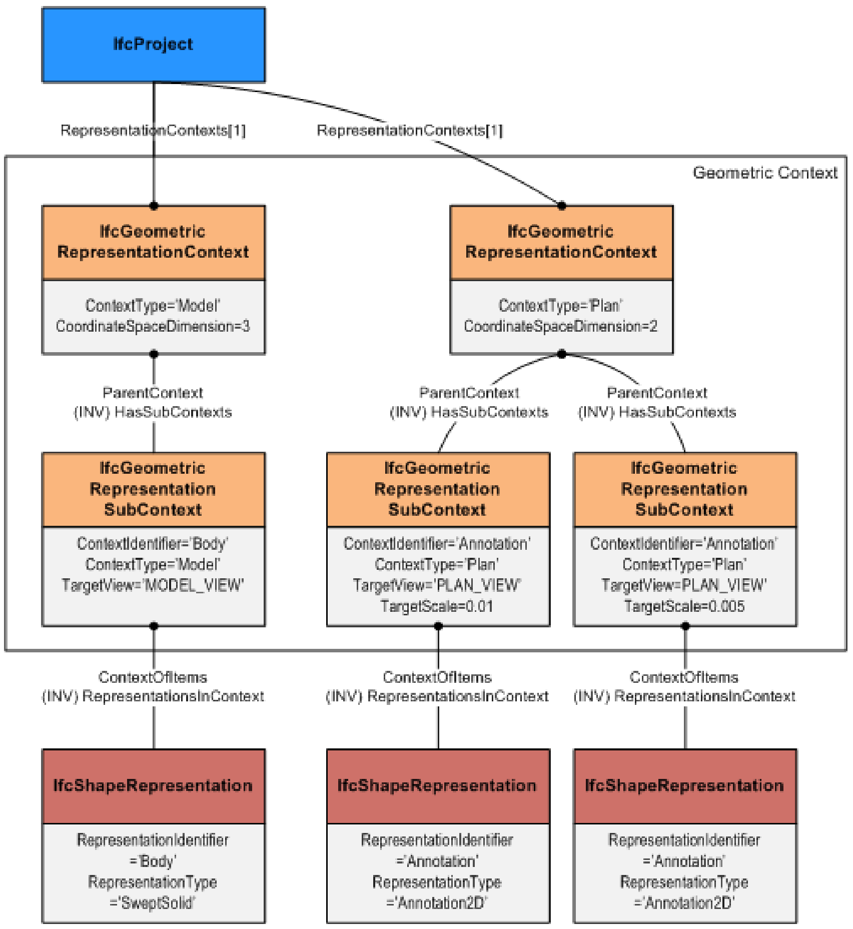
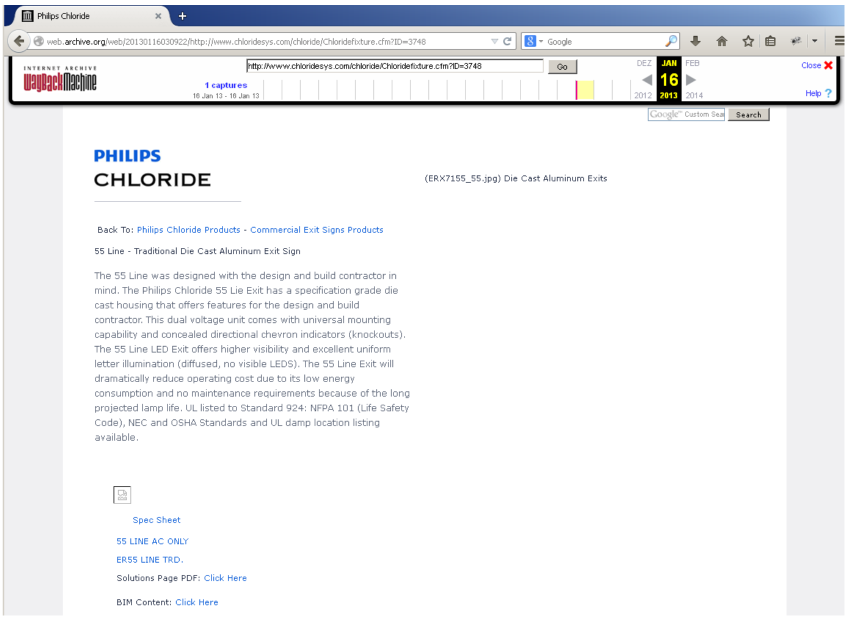

IfcProject
RepresentationContexts[1]
RepresentationContexts[1]
Geometric Context
IfcGeometric RepresentationContext
ContextType='Model'
CoordinateSpaceDimension=3
IfcGeometric RepresentationContext
ContextType='Plan'
CoordinateSpaceDimension=2
ParentContext (INV) HasSubContexts
ParentContext (INV) HasSubContexts
ParentContext (INV) HasSubContexts
IfcGeometric Representation SubContext
ContextIdentifier='Body'
ContextType='Model'
TargetView='MODEL_VIEW'
IfcGeometric Representation SubContext
ContextIdentifier='Annotation'
ContextType='Plan'
TargetView='PLAN_VIEW'
TargetScale=0.01
IfcGeometric Representation SubContext
ContextIdentifier='Annotation'
ContextType='Plan'
TargetView=PLAN_VIEW'
TargetScale=0.005
ContextOfItems (INV) RepresentationsInContext
ContextOfItems (INV) RepresentationsInContext
ContextOfItems (INV) RepresentationsInContext
IfcShapeRepresentation
RepresentationIdentifier ='Body'
RepresentationType ='SweptSolid'
IfcShapeRepresentation
RepresentationIdentifier ='Annotation'
RepresentationType ='Annotation2D'
IfcShapeRepresentation
RepresentationIdentifier ='Annotation'
RepresentationType ='Annotation2D'

Figure 5: Different context groups of shapes for the IfcGeometricRepresentationContext entity [5]

Content extent

A straightforward way to describe the information width of the IFC file is to give the total area of the modelled object or to give overall counts of different building parts, such as floors / storeys, walls, rooms, windows, doors, pipes or columns. The latter has clear advantages in the case of migration or when re-importing objects into native CAD systems, as the count allows for an easy metric to check against. Recent research has shown that these factors are on the rise, as the building profession is realizing the potential this information brings by, e.g., allowing a comparison of different models based on their size or state [23]. The information depth of an IFC file can be described via an attribute metric, i.e., by assessing how many optional attributes are used, how many entities were used or how many actors are captured in the object. In the case of the DURAARK dataset, only 14 files listed actors at all - and here typically only between 1-4 different actors.


Philips Chloride
web.archive.org/web/20130116030922/http://www.chloridesys.com/chloride/Chloridefixture.cfm?ID=3748
Google
INTERNET ARCHIVE
WaybackMachine
http://www.chloridesys.com/chloride/Chloridefixture.cfm?ID=3748
Go
1 captures
16 Jan 13 - 16 Jan 13
DEZ
JAN
FEB
16
2012
2013
2014
Close
Help
Google™ Custom Sea
Search

PHILIPS
CHLORIDE

(ERX7155_55.jpg) Die Cast Aluminum Exits

Back To: Philips Chloride Products - Commercial Exit Signs Products

55 Line - Traditional Die Cast Aluminum Exit Sign

The 55 Line was designed with the design and build contractor in mind. The Philips Chloride 55 Lie Exit has a specification grade die cast housing that offers features for the design and build contractor. This dual voltage unit comes with universal mounting capability and concealed directional chevron indicators (knockouts). The 55 Line LED Exit offers higher visibility and excellent uniform letter illumination (diffused, no visible LEDS). The 55 Line Exit will dramatically reduce operating cost due to its low energy consumption and no maintenance requirements because of the long projected lamp life. UL listed to Standard 924: NFPA 101 (Life Safety Code), NEC and OSHA Standards and UL damp location listing available.

Spec Sheet

55 LINE AC ONLY

ER55 LINE TRD.

Solutions Page PDF: Click Here

BIM Content: Click Here

Figure 6: Capture of a no longer available resource as viewed via the Internet Archive's WaybackMachine

A content extent with a big impact on digital preservation methods is the dependency on external information, e.g., linked web data resources. The DURAARK dataset includes a total of 6246 links to external web data resources via a URL, which are used as classification references but also as product links[38]. With a total of 4917 in only 23 files, the majority of the URLs point to `http://www.csiorg.net/uniformat`, however, that resource is not available and the correct link to the website of the Construction Specification Institute, who is responsible for the UniFormat standard is `http://www.csinet.org/uniformat`. While that link are rather generic, a more specific example is one file linking to `http://www.chloridesys.com/chloride/Chloridefixture.cfm?ID=3748` for a lighting fixture description. Neither the resource behind the full URL nor the top level domain leads to an available resource. The resource is, however, available via the Internet Archive's WaybackMachine (see Figure 6), where a January 16th 2013 capture of the

[38]See the following URL for an example of a linked product sheet: `http://www.bradleycorp.com/products/techdata/8825.pdf`

website links to site describing "55 Line - Traditional Die Cast Aluminium Exit Sign"[39]. While the site contains a brief description and allows us to identify the object, the spec sheet which the archived site links to has unfortunately not been archived and is thus not available via the WaybackMachine.

3.3 File Format Validation

As mentioned in D6.6.1, the libE57 reference implementation includes a validation tool, the suitability of which shall be tested towards validation in a digital preservation workflow. The E57 validator checks for:

- unknown bounds type
- bounds minimum greater than maximum
- all values against expected data types, declared bounding regions, "suspiciously large values" (e.g., for integer <1e12), "suspiciously small" values (e.g., for blob with byte count >4)
- invalid version numbers
- correctly defined coordinates (either as XYZ or as RAE)
- quaternion norm for plausibility

The output of the validator differentiates between "Error", "Warning", "Suspicious" and "Informational" level messages. Of the 96 E57 files in the DURAARK test set, 4 objects failed the validation process using the libE57 validation tool. 1 of the objects actually crashed the validator - further investigation showed that the object could not be rendered but had shown no problem with the file format identification and technical metadata extraction processes before. According to the extraction process, the object should have one scan, but none could be rendered. The other 3 objects could be rendered without a problem, but reported respectively 4 (E57 object containing 4 scans), 4889 (E57 object containing 19 scans) and 1777004643 (E57 object containing 139 scans) errors during the validation process. Further investigation needs to be put towards the reported error messages.

[39]For the archived resource in the WaybackMachine, see `http://web.archive.org/web/20130116030922/http://www.chloridesys.com/chloride/Chloridefixture.cfm?ID=3748`

While the validation tool seems to be reliable, the performance may prove to be a problem. In a few instances, the validation process took up to an hour. Figures 7 and 8 show the correlation between file size and duration as well as number of scans in the object and duration for the DURAARK test set, showing that the larger the object in bytes - regardless of the number of scans contained within - the longer the validation process with the libE57 validation tool takes.

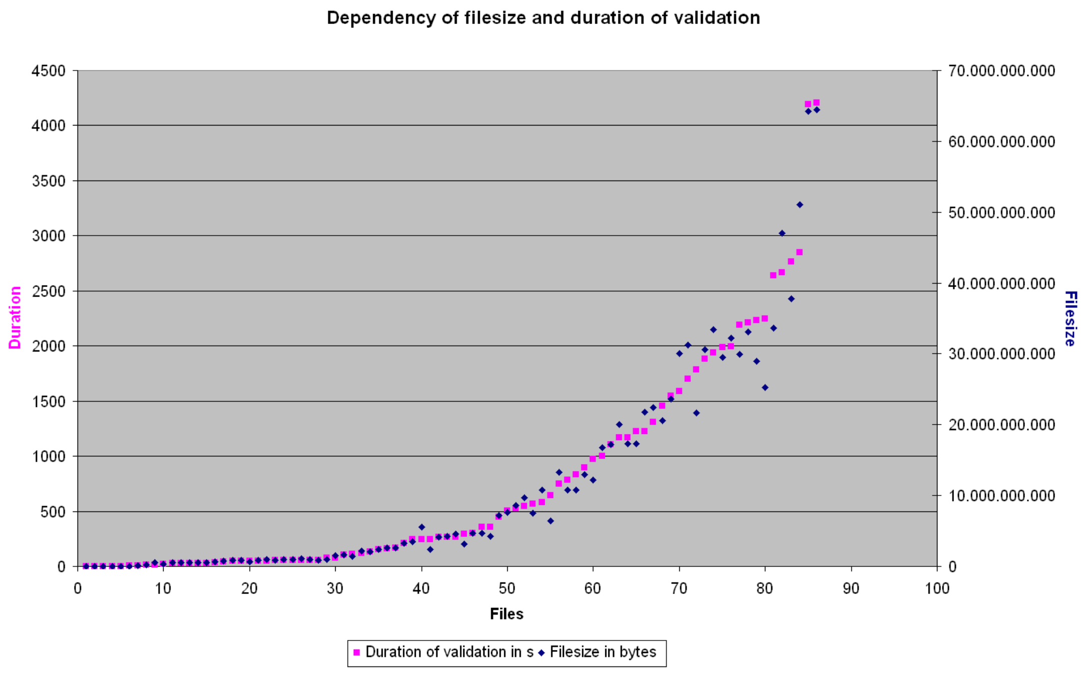


Figure 7: Correlation between validation process duration and filesize in bytes

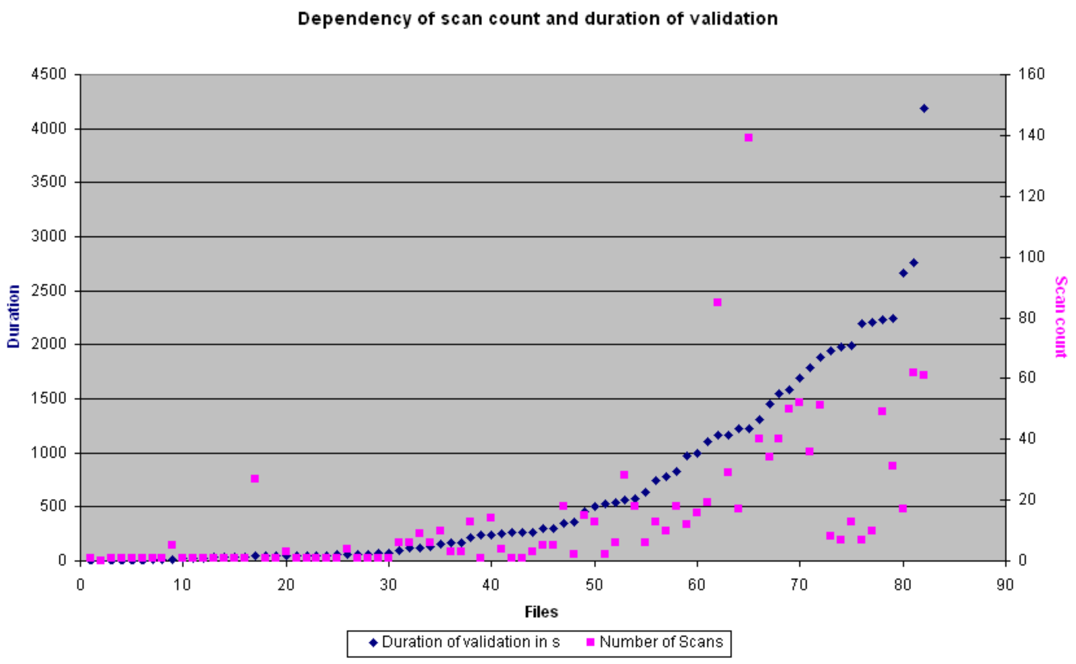
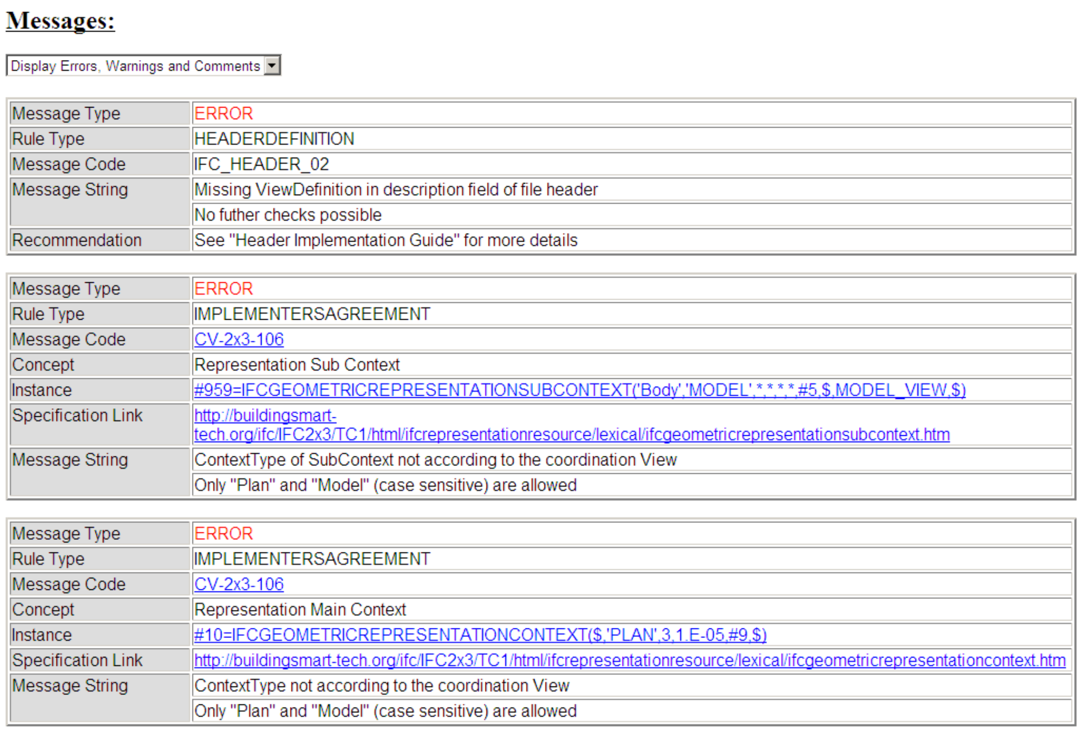


Figure 8: Correlation between validation process duration and number of scans in object

For IFC-SPF, the "Global Testing and Documentation Server" (GTDS)[40] and the corresponding KIT Karlsruhe stand-alone version ifcCheckingTool_Lite was introduced in deliverable D6.6.1. These tools check against the "largest" model view definition, so to speak - against the Coordination View. The KIT tool and GTDS check against Coordination View v1.0, however, the model view definition has been outdated since 2010, when it was superseded by Coordination View v2.0. With the release of IFC4, the work on the next version of the Coordination View has started.[41]. Running the test data - where supported - against the ifcCheckingTool_Lite returned no file of the test set as being error free. Error numbers returned ranged between 5-1608 errors. About 10% of the test data caused to tool to crash or malfunction, i.e., not returning the validation results. As the tool checks against all IFC2X2 / IFC2X3 dependencies, the granularity of errors varies accordingly. Figure 9 shows an example of errors returned via the tools reporting function.

[40]`http://gtds.buildingsmart.com/`

[41]See information on model view definitions at the buildingSmart website:`http://www.buildingsmart-tech.org/specifications/ifc-view-definition`



Messages:

Display Errors, Warnings and Comments

Message Type	ERROR
Rule Type	HEADERDEFINITION
Message Code	IFC_HEADER_02
Message String	Missing ViewDefinition in description field of file header
	No futher checks possible
Recommendation	See "Header Implementation Guide" for more details

Message Type	ERROR
Rule Type	IMPLEMENTERSAGREEMENT
Message Code	CV-2x3-106
Concept	Representation Sub Context
Instance	#959=IFCGEOMETRICREPRESENTATIONSUBCONTEXT('Body','MODEL',*,*,*,*,#5,$,MODEL_VIEW,$)
Specification Link	http://buildingsmart-tech.org/ifc/IFC2x3/TC1/html/ifcrepresentationresource/lexical/ifcgeometricrepresentationsubcontext.htm
Message String	ContextType of SubContext not according to the coordination View
	Only "Plan" and "Model" (case sensitive) are allowed

Message Type	ERROR
Rule Type	IMPLEMENTERSAGREEMENT
Message Code	CV-2x3-106
Concept	Representation Main Context
Instance	#10=IFCGEOMETRICREPRESENTATIONCONTEXT($,'PLAN',3,1.E-05,#9,$)
Specification Link	http://buildingsmart-tech.org/ifc/IFC2x3/TC1/html/ifcrepresentationresource/lexical/ifcgeometricrepresentationcontext.htm
Message String	ContextType not according to the coordination View
	Only "Plan" and "Model" (case sensitive) are allowed

Figure 9: Example for reporting returned by ifcCheckingTool_Lite tool

With IFC constantly growing in complexity to serve more needs of interoperability, data exchange and design-to-construction-to-maintenance activities of a building, validation becomes more and more complex. Furthermore, the desired level of information is highly dependent on the archival use case.

4 Semantic Preservation

4.1 Linking Semantic Concepts

As discussed in deliverable D6.6.1, prior to the DURAARK project, relevant sources of knowledge bases for 3D architectural data had not been identified or standardized. In addition, there were no existing standard methods to link changing semantic concepts. As a part of the DURAARK objectives, these issues have been addressed in the previous deliverables.

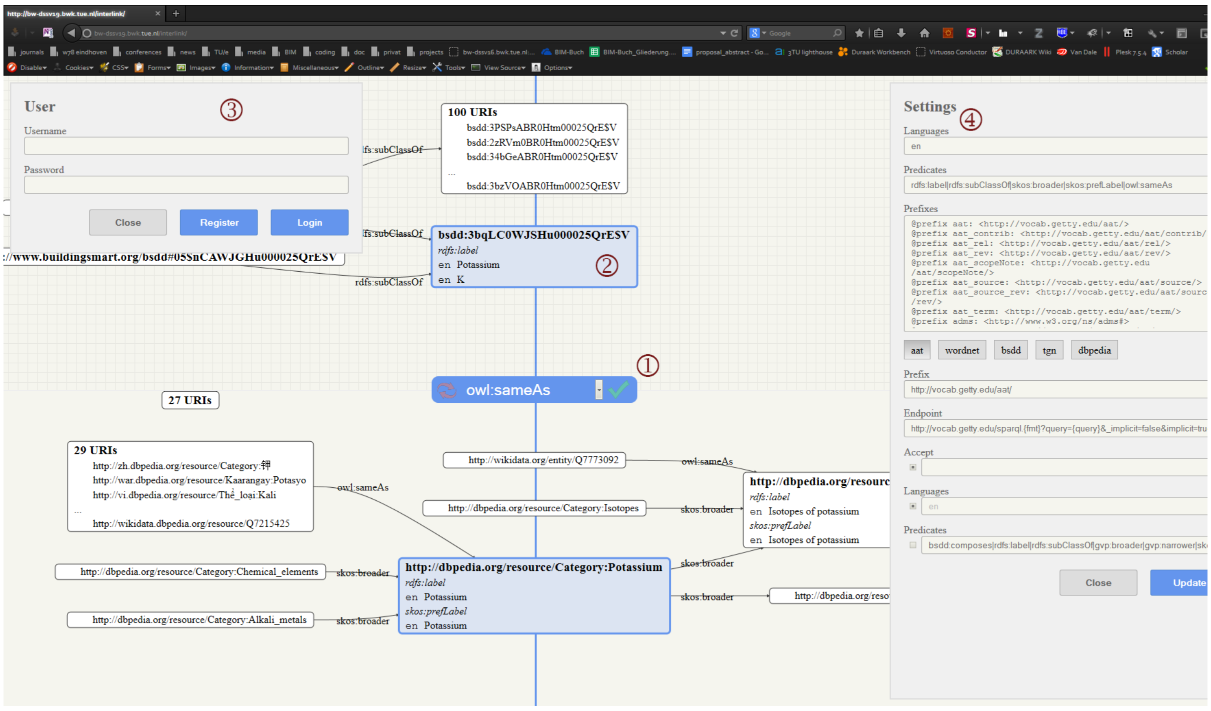
In the deliverable D3.3.1, existing metadata vocabularies and specification were examined and metadata vocabularies (*buildm*, *ifcm*, and *e57m*) were proposed to address the requirements identified. The further work conducted on these metadata vocabularies since is presented in chapter 5. Furthermore, D3.3.1 identified and presented knowledge bases and vocabularies which are suitable sources for the semantic enrichment of architectural archival data.

In deliverable D3.3.2, the DURAARK project presented the Semantic Digital Archive (SDA), a component that enables the storage of temporal snapshots of linked datasets and the restitution of archived versions of semantically enriched Building Information Models (BIMs) in digital preservation life-cycles. Alongside this, to cater for the evolution of the datasets used for semantic enrichment, a subcomponent called the Semantic Digital Observatory (SDO) was introduced by the project. The SDO profiles, monitors and updates the datasets used for enrichment.

In deliverable D3.3, the first prototype of the DURAARK SDA component was presented, consisting of the SDO and the storage component, the Semantic Digital Archive Storage (SDAS). The SDO facilitates the discovery of suitable relevant datasets as well as the their retrieval in order to provide structured metadata corresponding to the available datasets. The *dataset crawler module* and the *dataset profiling module* support the SDO in achieving these functionalities.

Upcoming Work
With an aim to improve the enrichment of archival data, link and align various re-

sources relevant to the architectural domain (from vocabularies such as BSDD[42] and Getty AAT[43]), the DURAARK project is currently developing an approach that utilizes the *Interlink* tool for link prediction, and relies on experts as well as crowdsourced judgments. This work will be elucidated in deliverable D3.6[44]. Figure 10 presents the interface of the tool presented in deliverable D3.4.



Languages: en

Predicates: rdfs:label|rdfs:subClassOf|skos:broader|skos:prefLabel|owl:sameAs

Prefixes:

```
@prefix aat: <http://vocab.getty.edu/aat/>
@prefix aat_contrib: <http://vocab.getty.edu/aat/contrib/
@prefix aat_rel: <http://vocab.getty.edu/aat/rel/>
@prefix aat_rev: <http://vocab.getty.edu/aat/rev/>
@prefix aat_scopeNote: <http://vocab.getty.edu
/aat/scopeNote/>
@prefix aat_source: <http://vocab.getty.edu/aat/source/>
@prefix aat_source_rev: <http://vocab.getty.edu/aat/sourc
/rev/>
@prefix aat_term: <http://vocab.getty.edu/aat/term/>
@prefix adms: <http://www.w3.org/ns/adms#>
```

Prefix: http://vocab.getty.edu/aat/

Endpoint: http://vocab.getty.edu/sparql.{fmt}?query={query}&_implicit=false&implicit=tru

Accept:

Languages: en

Predicates: bsdd:composes|rdfs:label|rdfs:subClassOf|gvp:broader|gvp:narrower|sk

Figure 10: Screenshot of the Manual Interlinking Prototype (delivered as of D3.4) showing two concepts being linked. 1) Relation widget with vertical alignment axis. 2) Node widget showing the labels of the selected node. 3) Register/Login dialog. 4) Configuration Dialog

4.2 Observing Changes in Linked Datasets

As identified in deliverable 6.6.1, current semantic preservation efforts mainly target the question of semantic enrichment and the preservation of the datasets themselves, the

[42]buildingSMART Data Dictionary `http://http://bsdd.buildingsmart.org/`

[43]Getty Research Institute Art & Architecture Thesaurus `http://www.getty.edu/research/tools/vocabularies/aat/`

[44]D3.6- Semantic Digital Interlinking and Clustering Prototype II.

question of how to trace changing concepts which have been captured in enriched datasets has not really been addressed.

In order to close this gap and to monitor changes in linked datasets that are used for the semantic enrichment of the archival objects, a Linked Iterative Crawler (presented in deliverable D3.3[45] has been developed).

The crawler plays an important role in the SDO, and has been designed in order to accommodate methods for assessing the temporal evolution of linked datasets. A dataset which has not been crawled earlier will thereby be crawled completely and all corresponding data is stored in a relational database. However, in case a dataset has already been crawled, the differences between the previously crawled state of the dataset and the current state are determined on-the-fly. These differences alone, called deltas or diffs, are then stored in the SDAS. Therefore, for any dataset that has been crawled multiple times at different crawl-points, by storing all data in the first crawl and consequently storing only the deltas, we are able to reconstruct the state of the dataset at any of the given crawl-points. A detailed account of how the evolution of datasets in the Linked Open Data (LOD) cloud can be monitored can be found in deliverable D3.3.

Upcoming Work
The project's objective to extend the crawler such that it attains the functionality of a *Focussed Crawler* has been presented and discussed in deliverable D3.4[46]. Currently ongoing work in WP3 is addressing methods to optimize the relevance of the crawled data with respect to the context of the archival object and thereby improve the overall quality and performance of the crawler.

4.3 Measuring and Describing the Impact of Changes on Enriched Data

Deliverable D6.6.1 also highlighted the existing dearth in methods to measure the impact which changes corresponding to one entity within a chosen dataset, has on other entities (in a domain specific setting).

With the help of the SDO provisions have been developed to both detect as well as

[45]D3.3- Semantic Digital Archive Prototype.
[46]D3.4- Semantic Digital Interlinking and Clustering Prototype.

monitor changes in external datasets that are used for semantic enrichment purposes. Further information on the importance of ensuring persistent correctness of the archival information as well as a detailed discussion of the computation of differences (i.e., the *diffs*) is provided in deliverable D3.3. In order to maintain persistent accuracy in the archived and enriched metadata, frequent crawling of relevant datasets will be carried out to allow for the capture possible changes in the form of *insertions, deletions, or updates* to the content.

Upcoming Work
Within WP3, methods that can cater to handling changes arising in the datasets are currently under development. These shall be used for the semantic enrichment of archived objects within an OAIS compliant digital preservation system.

5 Metadata

This chapter presents the current status of the DURAARK descriptive metadata set *buildm* and the technical metadata sets *e57m* and *ifcm*. As the schemas are evolving over the course of the project to include the ongoing development of methods and tools, the status presented here is not a final one, but an intermediate status. However, the current status of the schemas can be considered as a solid basis for further versions - it allows us to understand the necessary differentiation between the *Physical Asset* and the *Digital Object* layer at the descriptive level (see section 5.1) and the different technical criteria defined for E57 and IFC objects, leading to two necessary technical metadata sets.

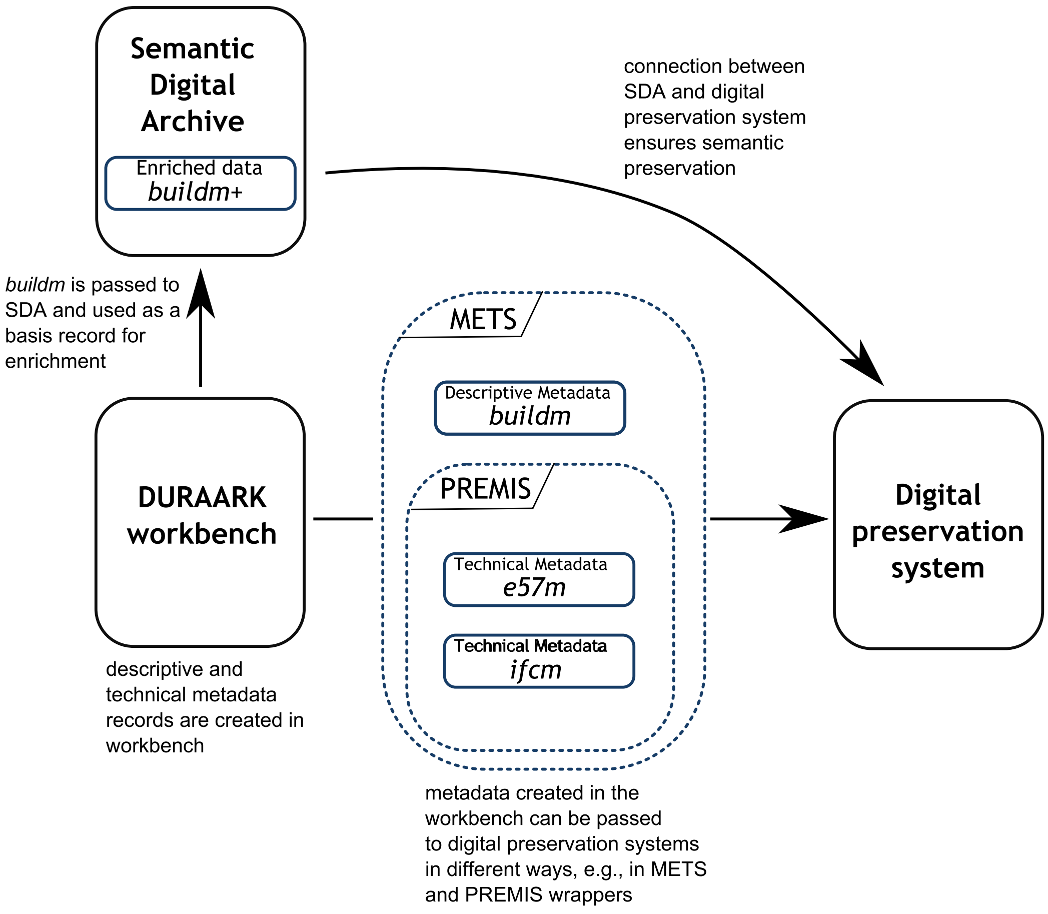

Semantic Digital Archive
Enriched data
buildm+
connection between SDA and digital preservation system ensures semantic preservation
buildm is passed to SDA and used as a basis record for enrichment
METS
Descriptive Metadata
buildm
PREMIS
Technical Metadata
e57m
Technical Metadata
ifcm
DURAARK workbench
Digital preservation system
descriptive and technical metadata records are created in workbench
metadata created in the workbench can be passed to digital preservation systems in different ways, e.g., in METS and PREMIS wrappers

Figure 11: DURAARK metadata sets *buildm*, *ifcm* and *e57m* and their role in the DURAARK architecture

Figure 11 shows at which points the metadata sets are leveraged within the DURAARK architecture. While the metadata is gathered through extraction and user-input in the DURAARK workbench, descriptive and technical metadata is passed to the digital preservation system, optionally via wrapper formats such as METS and/or PREMIS. Furthermore, the descriptive metadata set is passed to the SDA, where it gives a description of the archived object without the need to store the digital object itself. Within the SDA the *buildm* metadata set is extended through enrichment processes, as presented in deliverable D3.3.4, leading to an extended *buildm* set (*buildm+*). The preservation of the enriched objects is given through a connection between the SDA and the digital preservation system.

The separation between the *buildm* passed to the digital preservation system and the *buildm* passed to the SDA, where it is enriched and becomes the *buildm+*, is required due to different reasons. As an authoritative and clearly defined set of data, the *buildm* allows a clear description of the Physical Asset and the Digital Object over time. Through the accompanying description of the elements it can be ensured that the descriptive metadata can be understood over time, while the mapping to other schemas such as Dublin Core[47] allows for an integration of the descriptive metadata set into different preservation systems as well as digital libraries. Simultaneously, the *buildm+* allows for flexible enrichment, extending the descriptive set as required in user specific workflows and requirements. *Buildm+* thus becomes a living object not only on the vertical, but also on the horizontal level of content - updating original descriptive information as well as constantly introducing new elements to describe and capture the content and context of the physical and digital object alike.

This chapter focuses on the metadata sets directly passed to the digital preservation system upon initial ingest - namely, descriptive metadata in *buildm* and technical metadata in *e57m* and *ifcm*, as seen in Figure 11.

5.1 Descriptive Metadata for Architectural Data Objects

The DURAARK *buildm* descriptive metadata set was first proposed in deliverable D3.3.1. Based on this first suggestion, the schema was extended by comparing several existing metadata schemas for architectural 3D assets to the DURAARK requirements. Here, a

[47]See Annex for *buildm* mapping

focus was put on metadata schemas with a certain foothold in the digital library or digital preservation domains, while building-domain specific metadata schemas are currently only touched upon in a rudimentary way. It is expected that the draft of the *buildm* schema as proposed in this deliverable will be extended throughout the course of the project, as both - best practises amongst building-domain institutions as well as technical capabilities of automatic extraction and enrichment - are explored further. For the draft put forward in this deliverable, in particular the following existing schemas were analyzed[48]:

- CARARE v2.0[49]: The CARARE metadata stems from the Europeana associated Carare project, which aims to bring cultural heritage data to the digital library Europeana. The metadata schema is based on CIDOC-CRM[50], MIDAS Heritage[51], POLIS DTD[52] and LIDO[53]. CARARE differentiates between information about four main areas: collection, digital resource, heritage asset and activity [12].
- MIT Facade PIM [22], which focuses on the architectural project, including descriptive metadata on the project and on the digital preservation object, but not explicitly on the physical asset.
- The Recommendations from the “Historic Buildings and Monuments Commission for England”[18], which focus on 3D scans, recommending a minimal set of descriptive information about the digital object, accompanied with the monument name or ID.
- PROBADO3D MD Core [4], which only describes the digital resource.
- Dublin Core, which only describes the digital resource and is not an explicit schema for architectural 3D data, but a common schema for descriptive metadata on all types of digital objects. It can be considered the most generic and interoperable description.

For the analysis, the first *buildm* candidates from D3.3.1 as well as all elements from the schemas above were listed and matched against each other. This way, redundancies could

[48]All but CARARE were candidates identified as important resources in D6.6.1. Via CARARE the D6.6.1 candidate CIDOC-CRM and other relevant schemas were covered.
[49]`http://carare.eu/slk/Resources/CARARE-Documentation/CARARE-2.0-schema`
[50]`http://www.cidoc-crm.org/index.html`
[51]`http://www.english-heritage.org.uk/publications/midas-heritage/`
[52]`http://www.ics.forth.gr/CULTUREstandards/paradotea.htm`
[53]`http://network.icom.museum/cidoc/working-groups/data-harvesting-and-interchange/what-is-lido/`

be avoided while identifying the focus of the different schemas to be evaluated towards suitability for the DURAARK context. The mapping and the resulting DURAARK *buildm* element set is included in Annex 8.1. The DURAARK use cases described in D2.2.1 show that the project puts a strong focus on re-use scenarios from a facility maintenance and building owner point of view. At the same time, descriptive information about the digital object is crucial for digital preservation as well as digital library processes. Hence, extensive information about the physical asset as well as about the digital object must be included in the descriptive metadata schema. Physical asset and digital object may share the same elements for description, e.g., creator, but fill the element with different values. To ensure that both objects are described correctly, the DURAARK *buildm* schema must therefore differentiate clearly between *Physical Asset* and *Digital Object*. Figure 12 shows a high-level overview of the *buildm* schema description.

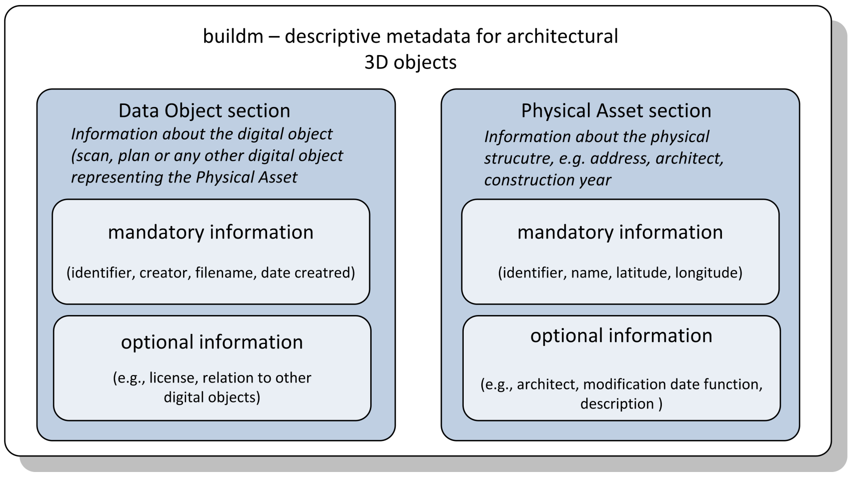

buildm – descriptive metadata for architectural 3D objects
Data Object section
Information about the digital object (scan, plan or any other digital object representing the Physical Asset
mandatory information
(identifier, creator, filename, date creatred)
optional information
(e.g., license, relation to other digital objects)
Physical Asset section
Information about the physical strucutre, e.g. address, architect, construction year
mandatory information
(identifier, name, latitude, longitude)
optional information
(e.g., architect, modification date function, description)

Figure 12: Conceptual view of the *buildm* schema and its components

As seen in Figure 12, both physical asset and digital object contain mandatory and optional elements. This differentiation is included from the get-go to allow for a low-level entry for data producers and consumers, who might otherwise be overwhelmed by the amount of schema elements. The mandatory fields contain the minimal information which is needed to understand the context of both the digital object as well as the physical asset it describes. They furthermore allow at least minimal support of the DURAARK use cases defined and described in D2.2.1. Within the scope of the DURAARK project the digital object in *buildm* may be a BIM object or a 3D scan, however, the section of the schema can accommodate any other digital representation of a physical asset, such as a 2D CAD plan or a photograph.

In the next sections each element of the *buildm* schema is described, using the following information:

Path notation of the element, in the order level:element, where level is either the PhysicalAsset or the DigitalObject

Label:	Label of the element.
Use:	May contain the following values: "Required"/"Optional" indicating if an object is mandatory, "Repeatable"/"Not Repeatable" if the element is repeatable within the level a buildm record. As a single buildm record may describe one physical asset but multiple digital objects, the separate digital objects will be described in their own instantiation of a digitalObject section. Usage here hence applies to one level describing one digital object.
Data Type:	Indicates the data type of the element, can be string, URL, boolean, integer, float/double or dateTime (according to ISO 8601).
Description:	Brief description of the element.
Example:	Example of how the element is used.

5.1.1 Minimal Requirement for PhyiscalAsset Information

Unlike with creative works such as publications, descriptive elements of buildings may change over time. Ownership may change when buildings are sold and the building area may change when additions are planned and constructed. Even address information may change over the course of time, as streets may be re-named or postal codes may be re-assigned when cities or communities merge. In the light of this, it is necessary to find a stable way to describe physical assets. Identifiers for buildings are common in both facility management as well as in cultural heritage. In addition to an identifier, a name shall be supplied - typically names of buildings are created by indicating the function and the location or owner, such as "Cologne Cathedral", "Apartment Building Southside" or "Science Building University". The last minimal set of information is the geolocation, consisting of longitude and latitude. While there is no guarantee that these criteria remain stable over time - the identification system may change, the name may change as the function changes and even the geolocation may change if, for example, the building is moved due to preservation reasons to a museum - the minimal set allows an easy identification of the physical asset and contains comparatively stable elements.

PhysicalAsset:identifier

Label:	identifier
Use:	Required, repeatable
Data Type:	String
Description:	A non-ambiguous reference of the physical asset within a given context.
Example:	UTS:CB71.02.04

PhysicalAsset:name

Label:	name
Use:	Required, repeatable
Data Type:	String
Description:	Title or name of the building, usually consisting of a combination of function and location.
Example:	Cologne Cathedral

PhysicalAsset:latitude

Label:	latitude
Use:	Required, not repeatable
Data Type:	String
Description:	The latitude of the physical asset's location in decimal degrees.
Example:	50.9411111

Physical Asset:longitude

Label:	longitude
Use:	Required, not repeatable

Data Type:	String
Description:	The longitude of the physical asset's location in decimal degrees.
Example:	6.95805555

5.1.2 Minimal requirement for DigitalObject Information

A physical asset can be described by many digital objects. As for the physical asset, the digital object shall be described by at least a minimal set of descriptive information to provide a rudimentary context of the object. While the identifier shall serve as a persistent reference to the digital object within a formal system or a repository, the name may be extracted from the digital object, be provided in form of the file name or be supplied by the user at the time of the deposit. The creator allows minimal information about provenance of the object while the date created relates the digital object to a point in time in the physical asset's lifecycle and also serves as important information for digital preservation actions when assessing an object's creation path.

DigitalObject:identifier

Label:	identifer
Use:	Required, repeatable
Data Type:	String
Description:	A non-ambigious reference of the digital object within a given context. Where possible, formal identification systems should be used.
Example:	ark:/12148/btv1b8448858m

DigitalObject:creator

Label:	creator
Use:	Required, repeatable
Data Type:	String
Description:	Creator/author of the digital object.
Example:	John Smith

DigitalObject:name

Label:	name
Use:	Required, not repeatable
Data Type:	String
Description:	The name of the digital object. This may be the file name or a name reflecting on the data described in the object.
Example:	Scan 8.Sept.2013 Cologne Cathedral

DigitalObject:dateCreated

Label:	dateCreated
Use:	Required, not repeatable
Data Type:	dateTime (according to ISO 8601)
Description:	The date the digital object was created.
Example:	2013-09-08T00:35:00

5.1.3 Information about size, function and ownership the PhysicalAsset

Further descriptive information about the physical asset shall include its owner, but also structural information such as total number of floors, rooms or overall area. This is important information which may regularly be revisited for various use cases, including, e.g., rental costs but also for the comparison of a digital object's spatial relationships to the entire physical asset. An example for this is evaluating whether a scan which just includes one floor is a partial or a complete interior description of a physical asset.

PhysicalAsset:owner

Label:	owner
Use:	Optional, repeatable
Data Type:	String
Description:	Person or organization who owns the physical asset. The element may be repeated to described different owners.
Example:	Hohe Domkirche zu Köln (corporate body under public law)

PhysicalAsset:buildingArea

Label:	buildingArea
Use:	Optional, not repeatable
Data Type:	String
Description:	Total floor area inside the building, including the measuring unit.
Example:	7914 sqm

PhysicalAsset:floorCount

Label:	floorCount
Use:	Optional, not repeatable
Data Type:	Integer
Description:	Total number of floors of the physical asset.
Example:	6

PhysicalAsset:numberOfRooms

Label:	numberOfRooms
Use:	Optional, not repeatable
Data Type:	Integer
Description:	Total number of rooms of the physical asset.
Example:	36

PhysicalAsset:function

Label:	function
Use:	Optional, repeatable
Data Type:	String
Description:	Current or intended use of the Physical Asset.
Example:	Religious building; cathedral

PhysicalAsset:architecturalStyle

Label:	architecturalStyle
Use:	Optional, repeatable
Data Type:	String
Description:	Architectural Style of the Physical Asset.
Example:	Gothic; neoclassicism

PhysicalAsset:description

Label:	description
Use:	Optional, repeatable
Data Type:	String
Description:	A description of the physical asset, e.g., to give historical background or further describe the status.
Example:	The Cologne Cathedral is Germany's most visited landmark. Construction started in 1248, paused between 1473 and 1842 and was completed in 1880.

5.1.4 Address Information:

While the location should be described through longitude and latitude (see section above), further information about the physical asset's location may be included. Properties describing the location further may be partial or detailed address information, such as the street address, city, region and country - but also a more general description of the location. Besides the benefit of direct human interpretability of a street address, the completeness-level of the street address may be an indicator as to how the geolocation was derived. In some cases, only the country and city of an object may be known, which limits the geolocation to a descriptor of the vicinity level instead of an exact location, e.g., by including 50.9364, 6.9528 as the geonames.org location for the city of Cologne as opposed to the correct geolocation of the Cologne cathedral itself, i.e., 50.9411111, 6.95805555.

PhysicalAsset:location

Label:	location
Use:	Optional, not repeatable
Data Type:	String
Description:	A general description of the Physical Asset's location.
Example:	Domhügel

PhysicalAsset:streetAddress

Label:	streetAddress
Use:	Optional, not repeatable
Data Type:	String
Description:	The street address corresponding to the Physical Asset.
Example:	Domkloster 4

PhysicalAsset:postalCodeStart

Label:	postalCodeStart
Use:	Optional, not repeatable
Data Type:	String
Description:	The postal code which corresponds to the location of the physical asset. If the physical asset is in a range of postal codes, for example if it describes a large apartment complex which spans over several postal codes, then this element marks the starting value of the postal code range.
Example:	50667

PhysicalAsset:postalCodeEnd

Label:	postalCodeEnd
Use:	Optional, not repeatable
Data Type:	String

Description:	If an address of a physical asset spans over several postal codes, e.g. for a large apartment complex, the starting postal code of the range is noted in postalCodeStart and the end of the range is noted in postalCodeEnd. In the case of addresses which just have one postal code, this value is empty.
Example:	50668

PhysicalAsset:postOfficeBoxNumber

Label:	postOfficeBoxNumber
Use:	Optional, not repeatable
Data Type:	String
Description:	If the mailing address of the physical asset includes a post office box number, this number is described in this field.
Example:	1234

PhysicalAsset:addressRegion

Label:	addressRegion
Use:	Optional, not repeatable
Data Type:	String
Description:	The region corresponding to the location of the physical asset, such as a state, province or area designation.
Example:	NRW - North Rhine-Westphalia

PhysicalAsset:addressLocality

Label:	postalLocality
Use:	Optional, not repeatable
Data Type:	String
Description:	The town / locality corresponding to the location of the physical asset.
Example:	Cologne

5.1.5 Construction process information:

Construction information can be divided into two main categories: the initial construction and (re-occurring) renovation / refurbishing. While information regarding the initial construction process is unique, reconstruction information can be repeated, describing the course of several remodelling projects. Properties such as architects or contributors span over initial as well as re-occurring work and may therefore be repeated.

PhysicalAsset:architect

Label:	architect
Use:	Optional, repeatable
Data Type:	String
Description:	The architect(s) of the physical asset.
Example:	Meister Gerhard; Barbara Schock-Werner

PhysicalAsset:contributor

Label:	contributor
Use:	Optional, repeatable
Data Type:	String
Description:	A person who contributed to the construction of the physical asset, e.g. the structural engineer or stone mason.
Example:	Gerhard Richter

PhysicalAsset:startDate

Label:	startDate
Use:	Optional, not repeatable
Data Type:	Integer
Description:	Year when the construction phase of the physical asset began.
Example:	1248

PhysicalAsset:completionDate

Label:	completionDate
Use:	Optional, not repeatable
Data Type:	Integer
Description:	Year when the construction phase of the physical asset was completed.
Example:	1880

PhysicalAsset:constructionTime

Label:	constructionTime
Use:	Optional, not repeatable
Data Type:	Integer
Description:	Duration of the construction phase of the physical asset in days.
Example:	230888

PhysicalAsset:rebuildingDate

Label:	rebuildingDate
Use:	Optional, repeatable
Data Type:	dateTime (according to ISO 8601)
Description:	Year when the rebuilding date of the physical asset began.
Example:	1944

PhysicalAsset:modificationDetails

Label:	modifciationDetails
Use:	Optional, repeatable
Data Type:	String
Description:	Explanation of the modification of the physical asset.
Example:	During 1944-1956 repairs of the damages which occurred during World War II were undertaken.

PhysicalAsset:cost

Label:	cost
Use:	Optional, repeatable
Data Type:	Double
Description:	Financial efforts which were needed for realizing the construction of the physical asset, in USD.
Example:	100.000.000,00

PhysicalAsset:rightsDetails

Label:	rightsDetails
Use:	Optional, repeatable
Data Type:	String
Description:	Information about rights, such as copyrights, license information or regulatory requirements related to the Physical Asset.
Example:	The Cologne Cathedral is a UNESCO World Heritage site. Special construction requirements apply.

5.1.6 Relationship to other Digital Objects

Digital objects represent physical asset with temporal and spatial dependencies, meaning that a digital object may only describe a part of a physical asset as it existed / was intended to exist at a given point in time. For example, a series of point cloud scans may describe an office building as it was on 2014-10-05, with each file representing a separate floor of the building. Along the same lines, a series of plans of the same office building may describe a remodelling as planned in 2013-06-20, with separate files being created for different model views of the building.

DigitalObject:isPartOf

Label:	isPartOf
Use:	Optional, repeatable
Data Type:	String

Description:	Links the digital object to an overarching digital object it is a part of, e.g., in the case of plans for different floors the object may link to an overall plan view of all the physical asset's rooms. The corresponding overarching object shall be identified through its identifier.
Example:	ark:/12148/btv1b8448858m

DigitalObject:hasPart

Label:	hasPart
Use:	Optional, repeatable
Data Type:	String
Description:	Links the digital object to child objects it may be related to, e.g., in the case of scans of the different floors which the overarching building representation may link to. The children objects shall be referenced through their identifiers.
Example:	ark:/12148/btv1b8448858m

5.1.7 Type of Digital Object

Descriptive information on the file format / document type level gives an understanding of the basic make-up and the rendering requirement of the digital object.

DigitalObject:format

Label:	format
Use:	Optional, not repeatable
Data Type:	String
Description:	The media type format of the digital object. Recommendation is to use the mime type to fill this value, where available.
Example:	Undefined

DigitalObject:hasType

Label:	hasType
Use:	Optional, not repeatable
Data Type:	String
Description:	The type of this digital object, e.g., plan or scan.
Example:	Scan

DigitalObject:hasFormatDetails

Label:	hasFormatDetails
Use:	Optional, not repeatable
Data Type:	String
Description:	Additional information out the digital object, e.g., it's encoding or compression information.
Example:	binary; E57

5.1.8 Provenance and Content Information of Digital Object

Further information about a digital object's lifecycle and content may be beneficial in providing more information about the content and usability for access and preservation reasons. An extent description may give necessary background information regarding the object's creation or processing before it entered the archive / collection. Information about provenance and license may indicate how the object may be re-used. Lastly, information about the unit code(s) used in the digital object and the level of detail in which the digital object is describing the physical asset can aid a user in making a decision about the object's content for usability scenarios.

DigitalObject:description

Label:	description
Use:	Optional, repeatable
Data Type:	String

Description:	A description of the digital object, e.g. to give information of how and why the object was created.
Example:	This scan was created in July 2014 to document the sculptures along the main portal of the cathedral.

DigitalObject:provenance

Label:	provenance
Use:	Optional, repeatable
Data Type:	String
Description:	A statement of any changes in ownership and custody of the digital object since its creation that are significant for its authenticity, integrity, and interpretation.
Example:	The scans were created by the company xyz on behalf of the Domkapitel.

DigitalObject:license

Label:	license
Use:	Optional, not repeatable
Data Type:	URL
Description:	A link to the license information to the digital object.
Example:	https://creativecommons.org/publicdomain/zero/1.0

DigitalObject:unitCode

Label:	unitCode
Use:	Optional, not repeatable
Data Type:	String
Description:	The unit of measurement given using the UN/CEFACT Common Code (3 characters). This determines in which unit properties corresponding to the Digital Object are entered.
Example:	mm

DigitalObject:levelOfDetail

Label:	levelOfDetail
Use:	Optional, not repeatable
Data Type:	String
Description:	The level of detail / level of development (LOD) in which the physical asset is described / captured in the digital object. If a standard reference system is used to describe the LOD, the system shall be named with the value.
Example:	CityGML LOD 3

DigitalObject:event

Label:	event
Use:	Optional, repeatable
Data Type:	String
Description:	Information for what the digital object was used for, such as in the case of digital objects created for presentations or competitions.
Example:	CAD Fair 2014

5.2 Technical Metadata for 3D Scans and Plans

The pivotal role which technical metadata plays in various tasks in digital preservation has been described in chapter 3.2. As in the case of *buildm*, first candidates for the technical metadata sets *e57m* and *ifcm* were also presented in deliverable D3.3.1. This work was extended through the development of technical metadata extractors, which were first presented in D2.4. The technical metadata set for E57 is directly based on the standard and elements which are already accessible through the libE57 reference implementation libraries. Since the main content of the scans are essentially points to be arranged in a coordinate system, the differentiation between content and technical information about the content is fairly straightforward. For IFC this is not necessarily the case, as the format specification is filled with semantically rich entities. The typical information classes for technical metadata - information about the creation process, parametric information and information about the content extent - has served as a guide to identify further elements for the *ifcm* schema.

The following subchapters will list all identified elements and give a brief description and example for each. It shall serve as a guide for archives to better interpret the output of the technical metadata extractors, to understand the technical information and the impact it may have on their preservation decisions and as a schematic description of information to be embedded as technical metadata in preservation metadata such as PREMIS.

The list of elements are described in the order of the respective schema structures as shown in figures 13 and 14. Each element is also listed in the path notation schema:schemaGroup:(schemaSubGroup:)element - here, it is important to note that elements can be assigned at a group or a subgroup / container level. If elements directly depend on each other, they are described in a single description box and noted as schemaGroup:schemaSubGroup:[element1, element2, element3, . . .], e.g. E57scan:cartesian_bounds:[x_minimum, x_maximum, y_minimum, y_maximum, z_minimum, z_maximum]. This notation does not follow RDF syntax, but shall ease the readability of the single element descriptions to the subgroup- and group-levels. As in the case of the *buildm* description, each element or group of elements is described as follows:

Path notation of the element(s), in the order group:[optional - subgroup]:element[s], where multiple elements are listed in brackets and separated by commas

Label:	label of the element, multiple elements are separated by semicolons
Use:	May contain the following values: “Required”/“Optional” indicating if an object is mandatory, “Repeatable”/“Not Repeatable” if the element can be instantiated multiple times within its respective group. Please note that for the technical metadata elements can be non-repeatable within a group but still exist multiple times in the overall schema, as the group itself is repeatable, e.g. the “scan” group.
Data Type:	Indicates the data type of the element(s), can be string, boolean, float, integer, dateTime (according to ISO 8601) or undefined if the standard or extractor makes no assumption on the content of the element.

Description:	Brief description of the element(s).
Example:	Example of how the element is used.

5.2.1 e57m schema elements

The following section lists and explains all elements for the *e57m* schema. For a sample output in JSON format, please refer to the Annex 1.4 of D2.4. From the extracted version proposed in D2.4, there have been some changes in form of combining values or changing the parsing, e.g., combining minor_version and major_version to version and parsing the date vectors to ISO8601 time_stamps. Furthermore, the following subchapter is the first description of the metadata elements for preservation purposes.

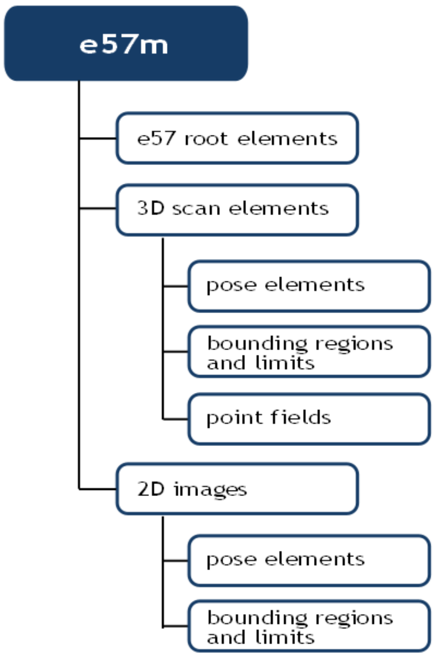


Figure 13: Basic overview of *e57m* structure

e57m - Root Level Information

E57-root level metadata describes basic attributes about the entire file, such as the version number of the file format or the number of 3D scans or 2D images contained in the file.

E57root:guid

Label:	guid
Use:	Required, not repeatable
Data Type:	String
Description:	A globally unique identification (GUID) String for the current version of the file. The standard does not prescribe the format of the GUID. Suggestions listed in the standard include IETF RFC4122 UUIDs or combinations of make/model/serial number of the scanner plus creationdatetime for non-networked equipment.
Example:	{7E3C7C9C-EFCB-4F5A-9A6F-98A08F72FB1B}

E57root:version

Label:	version
Use:	Required, not repeatable
Data Type:	String
Description:	The version of the file format. The current version (as of November 2014) is 1.0.
Example:	1.0

E57root:creation_datetime

Label:	creation_datetime
Use:	Optional, not repeatable
Data Type:	dateTime (according to ISO 8601)
Description:	Date and time the file was created.
Example:	2011-11-03T19:05:35

E57root:coordinate_metadata

Label:	coordinate_metadata
Use:	Optional, not repeatable
Data Type:	String

Description:	Describes the Coordinate Reference System using the well-known text (WKT) specification as described in the Open Geospatial Consortiums's (OGC) specification for Coordinate Transformation Services.The WKT allows the 3D and 2D data stored in the file to be referenced in a standardized coordinate reference system.
Example:	undefined

E57root:scan_count

Label:	scan_count
Use:	Required, not repeatable
Data Type:	Integer
Description:	Number of single scans the file aggregates. Within the e57 file, a scan is described by a data3D object.
Example:	5

E57root:image_count

Label:	image_count
Use:	Optional, not repeatable
Data Type:	Integer
Description:	Number of 2D images the file contains. Within the e57 file, each 2D image is described by a Image2D object.
Example:	5

E57root:scan_size

Label:	scan_size
Use:	Optional, not repeatable
Data Type:	String
Description:	Describes each scan by the following parameters: scanIndex number, number of Rows, number of Column, number of Points, number of groups, number of points per group, a boolean flag indicating whether the idElementName is “columnIndex”.

Example:	19

E57root:image_size

Label:	image_size
Use:	Optional, not repeatable
Data Type:	String
Description:	Describes each image by the following parameters: imageIndex number, image projection type (e.g., spherical, pinhole), image encoding type, image width, image height, number of bytes in the image, image mask type , (e.g., for PNG masks), image visual type.
Example:	285

e57m - scan level

E57scan:guid

Label:	guid
Use:	Required, not repeatable
Data Type:	String
Description:	Global unique identifier for each scan element.
Example:	{F0B3C105-325B-4FC9-9E01-3130153F9800}

E57scan:name

Label:	name
Use:	Optional, not repeatable
Data Type:	String
Description:	A name for the 3D data as assigned by the user.
Example:	parking000

E57scan:original_guids

Label:	original_guids

Use:	Optional, repeatable
Data Type:	String
Description:	If the object has been modified, i.e., is a result of merging muliple scans during processing, this element contains the guid of one or more scansElement which the data originated from.
Example:	{G1C4D216-436C-4FC9-9E01-4241264G0911}

E57scan:description

Label:	description
Use:	Optional, not repeatable
Data Type:	String
Description:	A description of the scans as supplied by the user.
Example:	Part1 Scan of 1st floor hallway.

E57scan:sensor_vendor

Label:	sensor_vendor
Use:	Optional, not repeatable
Data Type:	String
Description:	Manufacturer of scanner/sensor used to capture the scan.
Example:	FARO Scanner Production GmbH

E57scan:sensor_model

Label:	sensor_model
Use:	Optional, not repeatable
Data Type:	String
Description:	Model name of scanner/sensor used to capture the scan.
Example:	FARO Focus 3D S 120

E57scan:sensor_serial_number

Label:	sensor_serial_number

Use:	Optional, not repeatable
Data Type:	String
Description:	Serial number of scanner/sensor used to capture the scan.
Example:	XXY0714889234T

E57scan:sensor_hardware_version

Label:	sensor_hardware_version
Use:	Optional, not repeatable
Data Type:	String
Description:	Hardware version of scanner/sensor used to capture the scan.
Example:	3.001.124

E57scan:sensor_software_version

Label:	sensor_software_version
Use:	Optional, not repeatable
Data Type:	String
Description:	Data collection version running on scanner/sensor used to capture the scan.
Example:	4.8.0.23502

E57scan:sensor_firmware_version

Label:	sensor_firmware_version
Use:	Optional, not repeatable
Data Type:	String
Description:	Firmware version running on scanner/sensor.
Example:	“iQLib” 5.0.6.30068

E57scan:temperature

Label:	temperature
Use:	Optional, not repeatable

Data Type:	Float
Description:	Weather data. Ambient temperature as measured by the sensor of the camera at the point of capture, in degree Celsius.
Example:	18.4

E57scan:relative_humidity

Label:	relative_humidity
Use:	Optional, not repeatable
Data Type:	Float
Description:	Weather data. Relative humidity as measured by the sensor of the camera at point of capture, in percent.
Example:	23

E57scan:atmospheric_pressure

Label:	atmospheric_pressure
Use:	Optional, not repeatable
Data Type:	Float
Description:	Weather data. Atmospheric pressure as measured by the sensor of the camera, at point of capture, in Pascal.
Example:	101325

E57scan:acquisition_start

Label:	acquisition_start
Use:	Optional, not repeatable
Data Type:	dateTime (according to ISO 8601)
Description:	Start date and time of scans acquisition, as an ISO8601 date.
Example:	2011-11-02T13:45:21

E57scan:acquisition_end

Label:	acquisition_end

Use:	Optional, not repeatable
Data Type:	dateTime (according to ISO 8601)
Description:	End date and time of scans acquisition, as an ISO8601 date.
Example:	2011-11-02T14:34:32

e57m - Scan level - Pose

Pose information is necessary when the 3D scan data is stored in a local coordinate system, which is relative to the sensor, as opposed to a file-based coordinate system. The rotation and translation information captured in pose allows for the transformation of the 3D data to a file-based coordinate system.

E57scan:pose:rotation:[w;x;y;z]

Label:	w;
	x;
	y;
	z

Use:	The group is optional and not repeatable. If the group is present all values *[w;x;y;z]* are required and not repeatable.
Data Type:	Float; Float; Float; Float
Description:	These elements describe the scalar part of the quaternion (w), which shall be nonnegative, as well as the i (x), j (y) and k (z) coefficients of the quaternion.
Example:	0.99996960774189081; -0.0074585516927261801; -0.0022701539983015365; 0

E57scan:pose:translation:[x;y;z]

Label:	x;
	y;
	z

Use:	The group is optional and not repeatable. If the group is present all values *[x;y;z]* are required and not repeatable.
Data Type:	Float; Float; Float
Description:	These elements describe the x, y and z coordinates of the translation, in meters.

Example:	89.951072690000004;
	1.8420018;
	2e-008

e57m - Scan level - Bounding regions and limits

Within an E57 file various bounding region limits are set, such as the index bounds, depending on the type of index used, the bounding regions, depending on the type of coordinate system used, the color limits and the intensity limits, if color is used.

E57scan:index_bounds:[**row_minimum; row_maximum; col_minimum; col_maximum; return_minimum; return_maximum**]

Label:	row_minimum;
	row_maximum;
	col_minimum;
	col_maximum;
	return_minimum;
	return_maximum

Use:	The group is optional and not repeatable. If the group is present all values *[row_minimum; row_maximum; col_minimum; col_maximum; return_minimum; return_maximum]* are optional and not repeatable.
Data Type:	Integer; Integer; Integer; Integer; Integer; Integer

Description:	The points in of the scans can be organized in rows, columns or by return numbers. For the respective organization form in use, minimum and maximum values need to be defined to describe the valid indices for the bounds.

Example:	0;
	3470;
	0;
	8213;
	0;
	0

E57scan:cartesian_bounds:[**x_minimum; x_maximum; y_minimum; y_maximum; z_minimum; z_maximum**]

Label:	x_minimum;
	x_maximum;
	y_minimum;
	y_maximum;
	z_minimum;
	z_maximum

Use:	The group is optional and not repeatable. If the group is present all values *[x_minimum; x_maximum; y_minimum; y_maximum; z_minimum; z_maximum]* are required and not repeatable.
Data Type:	Float; Float; Float; Float; Float; Float
Description:	E57 may use two different coordinate systems: cartesian or spherical. If the cartesian coordinate system is used, the minimum and maximum values for the bounding region needs to be defined: the min. and max. extend in the x direction, min and max extend in the y direction and min. and max. extend in the z direction, in meter.

Example:	-68.432470999999993; 57.134830999999998; -59.897230999999998; 70.512130999999997; -2.0202709999999997; 3.779801

E57scan:sphericalbounds:[range_minimum; range_maximum; elevation_minimum; elevation_maximum; azimuth_minimum; azimuth_maximum]

Label:	range_minimum;
	range_maximum;
	elevation_minimum;
	elevation_maximum;
	azimuth_minimum;
	azimuth_maximum

Use:	The group is optional and not repeatable. If the group is present some values *[range_minimum; range_maximum; elevation_minimum; elevation_maximum]* are required and not repeatable and some values *[azimuth_minimum; azimuth_maximum]* are optional and not repeatable.
Data Type:	Float; Float; Float; Float; Float; Float
Description:	E57 may use two different coordinate systems: cartesian or spherical. If the points are represented in spherical coordinates, the bounding region needs to be defined: min. and max. (range) extend in the r direction in meters t, min. and max. (elevation) extend in the phi direction in radians and min. and max. (azimuth) extend in the theta direction in radians.

Example:	1.6562939999999999;
	90.929899999999989;
	-1.0909121353667537;
	1.5701933463079427;
	0;
	-6.4112263142845904e-007

E57scan:intensity_limits:[intensity_minimum; intensity_maximum]

Label:	intensity_minimum; intensity_maximum
Use:	The group is optional and not repeatable. If the group is present all values *[intensity_minimum; intensity_maximum]* are required and not repeatable.
Data Type:	Integer
Description:	The intensity describes the strength of the signal for a point. The intensity limits need to be defined if the points in the pointRecord of the scans contain intensity information and if the sensor/scanner is has defined minimum and maximum values for the intensity which can be produced by the scanner/sensor. The standard does not define a unit for this element.
Example:	0; 1

E57scan:color_limits:[color_red_minimum; color_red_maximum; color_green_minimum; color_green_maximum; color_blue_minimum; color_blue_maximum]

Label:	color_red_minimum;
	color_red_maximum;
	color_green_minimum;
	color_green_maximum;
	color_blue_minimum;
	color_blue_maximum

Use:	The group is optional and not repeatable. If the group is present all values *[color_red_minimum; color_red_maximum; color_green_minimum; color_green_maximum; color_blue_minimum; color_blue_maximum]* are required and not repeatable.
Data Type:	Integer
Description:	PointRecords can be optionally described by color elements. These elements describe the limit of values for RGB, which the scanner is capable of producing. The standard does not prescribe a unit.

Example:	0;
	255;
	0;
	255;
	0;
	255

E57scan:pointsSize

Label:	pointsSize
Use:	Required, not repeatable
Data Type:	Integer
Description:	Returns the number of records found in the respective scan.
Example:	26602731

e57m - Scan level - Point Fields

The point fields check the structure of the point records for the availability of standardized fields as well as the meaningfulness for some of the content.

E57scan:point_fields:cartesian_fields:[**cartesian_x_field; cartesian_y_field; cartesian_z_field; cartesian_invalid_state_field**]

Label:	cartesian_x_field;
	cartesian_y_field;
	cartesian_z_field;
	cartesian_invalid_state_field

Use:	The group is optional and not repeatable. If the group is present all values *[cartesian_x_field; cartesian_y_field; cartesian_z_field; cartesian_invalid_state_field]* are required and not repeatable.
Data Type:	Boolean; Boolean; Boolean; Boolean
Description:	Checks for the existence of a pointer to a buffer with the x, y and z coordinates, returning true for the respective element. The cartesian_invalid_state_field indicates whether the content of the x, y and z fields are not meaningful.
Example:	true; true; true; false

E57scan:point_fields:spherical_fields:[**spherical_range_field; spherical_elevation_field; spherical_azimuth_field; spherical_invalid_state_field**]

Label:	spherical_range_field;
	spherical_elevation_field;
	spherical_azimuth_field;
	spherical_invalid_state_field

Use:	The group is optional and not repeatable. If the group is present all values *[spherical_range_field; spherical_elevation_field; spherical_azimuth_field; spherical_invalid_state_field]* are required and not repeatable.
Data Type:	Boolean; Boolean; Boolean; Boolean
Description:	Checks for the existence of a pointer to a buffer with the x, y and z coordinates, returning true for the respective element. The spherical_invalid_state_field indicates whether the content of the x, y and z fields are not meaningful.
Example:	true; true; true; false

E57scan:point_fields:point_range:[point_range_minimum; point_range_maximum; point_range_scaled_integer]

Label:	point_range_minimum;
	point_range_maximum;
	point_range_scaled_integer

Use:	The group is optional and not repeatable. If the group is present all values *[point_range_minimum; point_range_maximum; point_range_scaled_integer]* are required and not repeatable.
Data Type:	Float; Float; Float
Description:	Returns the values of point_range min. and max. values. The point_range_scaled_integer field either indicates that the point records should be configured as a scaled integer node, returning 1, or returns the float node.
Example:	268.435454999999999; -268.43545499999999; 9.9999999999999995e-007

E57scan:point_fields:angles:[angle_minimum; angle_maximum; angle_scaled_integer]

Label:	angle_minimum;
	angle_maximum;
	angle_scaled_integer

Use:	The group is optional and not repeatable. If the group is present all values *[angle_minimum; angle_maximum; angle_scaled_integer]* are required and not repeatable.
Data Type:	Float; Float; Float
Description:	For scans in a spherical coordinate system, these values return the min. and max. angle which may be used. The angle_scaled_integer either returns a 1, indicating that the point record should be configured as a scaled integer node, or returns the float node.
Example:	0; 0; 0

E57scan:point_fields:index_fields:[row_index_field; row_index_maximum; column_index_field; column_index_maximum; return_index_field; return_count_field; return_maximum]

Label:	row_index_field;
	row_index_maximum;
	column_index_field;
	column_index_maximum;
	return_index_field;
	return_count_field;
	return_maximum

Use:	The group is optional and not repeatable. If the group is present all values *[row_index_field; row_index_maximum; column_index_field; column_index_maximum; return_index_field; return_count_field; return_maximum]* are required and not repeatable.
Data Type:	Boolean; Integer; Boolean; Integer; Boolean; Boolean; Integer

Description:	The index fields indicate if the respective index fields for the type of index (row, column, return) are active. For the return index, the count field is checked against being active as well. The maximum fields return the maximum values for the respective index, the minimum is assumed to be set to 0.

Example:	true;
	4294967295;
	true;
	4294967295;
	false;
	false;
	0

E57scan:point_fields:time_fields:[time_stamp_field; is_Time_Stamp_invalid; time_Maximum]

Label:	time_stamp_field;
	is_Time_Stamp_invalid;
	time_Maximum

Use:	The group is optional and not repeatable. If the group is present all values *[time_stamp_field; is_Time_Stamp_invalid; time_Maximum]* are required and not repeatable.
Data Type:	Boolean; Boolean; Float
Description:	Checks for the presence and validity of a time_stamp in the file, returning true when time_stamp exists.Time_Maximum indicates a maximum values which point record time_stamp fields should be configured with.
Example:	true; true; 0

E57scan:point_fields:intensity_color_fields:[intensity_field; is_intensity_invalid_field; intensity_scaled_integer; color_red_field; color_green_field; color_blue_field; is_color_invalid_field]

Label:	intensity_field;
	is_intensity_invalid_field;
	intensity_scaled_integer;
	color_red_field;
	color_green_field;
	color_blue_field;
	is_color_invalid_field

Use:	The group is optional and not repeatable. If the group is present all values *[intensity_field; is_intensity_invalid_field;* *intensity_scaled_integer; color_red_field; color_green_field;* *color_blue_field; is_color_invalid_field]* are optional and not repeatable.
Data Type:	Boolean; Boolean; Float; Boolean; Boolean; Boolean; Boolean
Description:	Checks for the presence and use of intensity and color schemes in the respective fields, returning true when found. The intensity_scaled_integer field indicates either that the point record intensity fields should be configured as a scaled integer node (returning 1) or returns the float node or the integer node. The is_color_invalid_field indicates whether the content of the color fields for red, green and blue are not meaningful.

Example:	true;
	false;
	0.00048851978505129456;
	true;
	true;
	true;
	false

e57m - Image level

Information at the images level describes each 2D image contained within the E57 file.

E57image:name

Label:	name
Use:	Optional, not repeatable
Data Type:	String
Description:	A name for the image as assigned by the user.
Example:	parking000image

E57image:guid

Label:	guid
Use:	Required, not repeatable
Data Type:	String
Description:	Global unique identifier for each image element.
Example:	{76BD148C-D22A-4FE3-8CB2-0FB01F96698B}

E57image:representation

Label:	representation
Use:	Optional, not repeatable
Data Type:	String
Description:	Describes the representation type of the image. The representation type can either be defined by the spherical, cylindrical or pinhole camera projection model - or follow no camera projection model, in which case the image shall serve as a visual reference only.
Example:	spherical

E57image:description

Label:	description
Use:	Optional, not repeatable
Data Type:	String

Description:	A description of the image as supplied by the user.
Example:	This is a jpeg image taken at the same time as the 3D capture.

E57image:acquisition_datetime

Label:	acquisition_datetime
Use:	Optional, not repeatable
Data Type:	dateTime (according to ISO 8601)
Description:	Timestamp of the acquisition time of the image, as an ISO8601 date.
Example:	2011-11-02T14:34:32

E57image:associated_data3D_guid

Label:	associated_data3D_guid
Use:	Optional, not repeatable
Data Type:	String
Description:	Lists the guid of the 3D scan image, which this image corresponds with.
Example:	{F0B3C105-325B-4FC9-9E01-3130153F9800}

E57image:sensor_vendor:

Label:	sensor_vendor
Use:	Optional, not repeatable
Data Type:	String
Description:	Manufactures of the sensor which was used to capture this image.
Example:	FARO Scanner Production GmbH

E57image:sensor_model

Label:	sensor_model
Use:	Optional, not repeatable
Data Type:	String

Description:	The model name of the sensor which was used to capture this image.
Example:	FARO Focus 3D S210

E57image:sensor_serial_number:

Label:	sensor_serial_number:
Use:	Optional, not repeatable
Data Type:	String
Description:	The serial number of the sensor which was used to capture this image.
Example:	XXXY0714889234T

e57m - Image level - Pose

Pose information is necessary when the image data is stored in a local coordinate system, which is relative to the sensor, as opposed to a file-based coordinate system. The rotation and translation information captured in pose allows for the transformation of the Image data to a file-based coordinate system.

E57image:pose:rotation:[w;x;y;z]

Label:	w;
	x;
	y;
	z

Use:	The group is optional and not repeatable. If the group is present all values *[w;x;y;z]* are required and not repeatable.
Data Type:	Float; Float; Float; Float
Description:	These elements describe the scalar part of the quaternion (w), which shall be nonnegative, as well as the i (x), j (y) and k (z) coefficients of the quaternion.

Example:	0.70283815264201144;
	0.077691052038131911;
	0.088163333920523737;
	0.070127669443617587

E57image:pose:translation:[x;y;z]

Label:	x;
	y;
	z

Use:	The group is optional and not repeatable. If the group is present all values *[x;y;z]* are required and not repeatable.
Data Type:	Float; Float; Float
Description:	These elements describe the x, y and z coordinates of the translation, in meters.

Example:	89.951072690000004;
	1.8420018;
	2e-008

e57m - Image level - Representation

Further technical metadata is captured depending on the respective representation type of the image.

E57image:visual_ref_representation:[**jpeg_image_size; png_image_size; image_mask_size; image_width; image_height**]

Label:	jpeg_image_size;
	png_image_size;
	image_mask_size;
	image_width;
	image_height

Use:	The group is optional and repeatable. If the group is present some values *[image_width; image_height]* are required and not repeatable and some values *[jpeg_image_size; png_image_size; image_mask_size]* are optional and not repeatable.
Data Type:	Integer; Integer; Integer; Integer; Integer
Description:	These elements are used, if an image is to be used as a visual reference. The image is then described with the size of the image data (either jpeg or png) found in the blob node, if used - the size of the image mask in the blob node as well as the image width and height in pixels.

Example:	0;
	23551883;
	0;
	8187;
	3471

E57image:spherical_representation:[**jpeg_image_size; png_image_size; image_mask_size; image_width; image_height; pixel_width; pixel_height**]

Label:	jpeg_image_size;
	png_image_size;
	image_mask_size;
	image_width;
	image_height;
	pixel_width;
	pixel_height

Use:	The group is optional and repeatable. If the group is present some values *[image_width; image_height; pixel_width; pixel_height]* are required and not repeatable and some values *[jpeg_image_size; png_image_size; image_mask_size]* are optional and not repeatable.
Data Type:	Integer; Integer; Integer; Integer; Integer; Float; Float

Description:	These elements are used for images which are mapped to 3D using the spherical camera projection model. The image is then described through the size of the image data (either jpeg or png) found in the blob node, if used - the size of the image mask in the blob node, the image width and heights in pixel as well as by the pixel's width and height in radians.

Example:	0;
	23551883;
	0;
	8187;
	3471;
	0.0076745772193576565;
	0.007666681778157584

E57image:pinhole_representation:[jpeg_image_size; png_image_size; image_mask_size; image_width; image_height; focal_length; pixel_width; pixel_height; principal_point_x; principal_point_y]

Label:	jpeg_image_size;
	png_image_size;
	image_mask_size;
	image_width;
	image_height;
	focal_length;
	pixel_width;
	pixel_height;
	principal_point_x;
	principal_point_y

Use:	The group is optional and repeatable. If the group is present some values *[image_width; image_height; focal_length; pixel_width; pixel_height; principal_point_x; principal_point_y]* are required and not repeatable and some values *[jpeg_image_size; png_image_size; image_mask_size]* are optional and not repeatable.
Data Type:	Integer; Integer; Integer; Integer; Integer; Float; Float; Float; Float; Float
Description:	These elements are used for images which are mapped to 3D using the pinhole camera projection model. The image is then described with the size of the image data (either jpeg or png) found in the blob node, if used - the size of the image mask in the blob node, the image width and height in pixel, the camera's focal length in meters, the pixels' width and height, both in meters, as well as the x and y coordinates of the principal point found in the image, in pixels.

Example:	23551883;
	0;
	0;
	8187;
	3471;
	1.5;
	0.0076745772193576565;
	0.007666681778157584;
	4576;
	1735

E57image:cylindrical_representation:[**jpeg_image_size; png_image_size; image_mask_size; image_width; image_height; pixel_width; pixel_height; radius; principal_point_y**]

Label:	jpeg_image_size;
	png_image_size;
	image_mask_size;
	image_width;
	image_height;
	pixel_width;
	pixel_height;
	radius;
	principal_point_y

Use:	The group is optional and repeatable. If the group is present some values *[image_width; image_height; radius; pixel_width; pixel_height; principal_point_y]* are required and not repeatable and some values *[jpeg_image_size; png_image_size; image_mask_size]* are optional and not repeatable.
Data Type:	Integer; Integer; Integer; Integer; Integer; Float; Float; Float; Float

Description:	These elements are used for images which are mapped to 3D using the cylindrical projection model. The image is then described with the size of the image data (either jpeg or png) found in the blob node, if used - the size of the image mask in the blob node, the image width and height in pixel, the camera's focal length in meters, the pixels' width in radians and height in meters as well as the radius of the cylinder in meters and the y coordinates of the principal point found in the image, in pixels.

Example:	23551883;
	0;
	0;
	8187;
	3471;
	1.5;
	0.0076745772193576565;
	0.007666681778157584;
	1.25

5.2.2 ifcm schema elements

The following section lists and explains all elements for the current version of the *ifcm* schema. The list of elements has been extended significantly since the first proposal in D3.3.1. Figure 14 shows a basic overview of the *ifcm* structure - while the header elements and ifcParameters section contain elements whose values are directly extracted from the corresponding entities in the IFC file, count objects and information metric calculate values by counting existing entities within the IFC file. The dependencies group contains an element which aggregates information from several IFC entities - in this specific case, links to web resources.

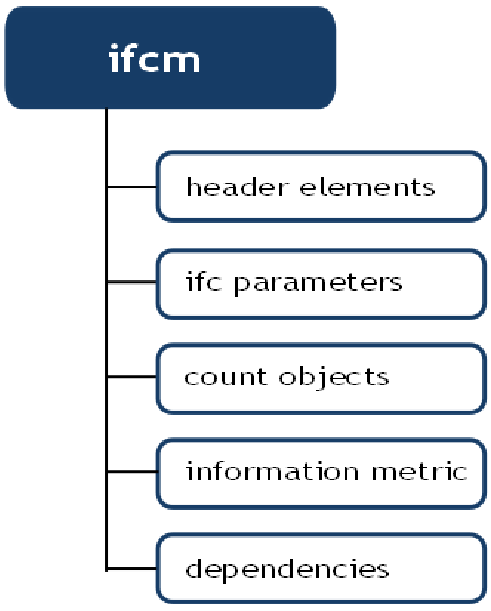


Figure 14: Basic overview of the *ifcm* structure

ifcm - header elements

The following elements are extracted from the STEP physical file format header portion of the IFC file. They contain information about the creation process of the file as well as about the schema and view make-up of the digital object.

ifcm:header:name

Label:	name
Use:	Required, not repeatable
Data Type:	String
Description:	Name of digital object as used for data exchange, as taken from the *name* attribute in IFC header's *FILE_NAME* entity.
Example:	Plan Hartnett Hall, Minot State University Campus

ifcm:header:creationDate

Label:	creationDate
Use:	Required, not repeatable
Data Type:	dateTime (according to ISO 8601)
Description:	Creation date of IFC file in ISO 8601 format, as taken from the *time_stamp* attribute in IFC header's *FILE_NAME* entity.
Example:	2013-05-28T10:23:19

ifcm:header:author

Label:	author
Use:	Optional, not repeatable
Data Type:	String
Description:	Creator of IFC file, as taken from the *author* attribute in IFC header's *FILE_NAME* entity.
Example:	John Doe

ifcm:header:organization

Label:	organization
Use:	Optional, not repeatable
Data Type:	String
Description:	Organization, which the creator of the IFC file is associated with, as taken from the *organization* attribute in IFC header's *FILE_NAME* entity.

Example:	Architecture and Construction Ltd.

ifcm:header:preprocessor

Label:	preprocessor
Use:	Optional, not repeatable
Data Type:	String
Description:	Name and version of the toolbox used to create the IFC file, as taken from the *preprocessor_version* attribute in IFC header's *FILE_NAME* entity.
Example:	DDS-IFC v2.0

ifcm:header:originatingSystem

Label:	originatingSystem
Use:	Optional, not repeatable
Data Type:	String
Description:	Name and version of the CAD system or other application used to generate the IFC file, including the build information, where possible. As taken from the *originating_system* attribute in IFC header's *FILE_NAME* entity.
Example:	DDS-CAD Version 8.0 Win32 build 29/6-2012

ifcm:header:authorization

Label:	authorization
Use:	Optional, not repeatable
Data Type:	String
Description:	Name and address of the person / organization who authorized the transfer of the IFC file, as taken from the *authorization* attribute in IFC header's *FILE_NAME* entity.
Example:	Fred Miller, Architecture and Construction Ltd.

ifcm:header:fileSchema

Label:	fileSchema
Use:	Required, not repeatable
Data Type:	String
Description:	Schema version of the IFC file, as taken from the *FILE_SCHEMA* entity in the IFC header file.
Example:	IFC2x3

ifcm:header:viewDefinition

Label:	viewDefinition
Use:	Required, not repeatable
Data Type:	String
Description:	Describes one or more model view definitions of the IFC File, as taken from the *view_defintion* attribute of the *FILE_DESCRIPTION* entity in the IFC header file. Values for view definitions have to be formally agreed upon by building-SMART.
Example:	FMHandOverView

ifcm:header:exportOptions

Label:	exportOptions
Use:	Optional, not repeatable
Data Type:	String
Description:	Describes export options or additional filters used for the IFC generation, as taken from the *FILE_DESCRIPTION* entity in the IFC header file.
Example:	Build Number of the IFC_2X2 interface: 02550 (26-09-2003)

ifcm - ifcParameters

The ifcParameters are values from corresponding IFC entities. For transparency reasons, the name of the *ifcm* elements are congruent with the name of the corresponding IFC entity.

ifcm:ifcparameters:ifcApplication

Label:	ifcApplication
Use:	Optional, repeatable
Data Type:	String
Description:	Describes software involved in the IFC/BIM process of the object, as taken from the ifcApplication enitity. The notation includes application developer, version, application full name and identifier.
Example:	ArchiCAD 7.0

ifcm:ifcparameters:IfcGeometricRepresentationContext

Label:	IfcGeometricRepresentationContext
Use:	Optional, repeatable
Data Type:	String
Description:	Describes the context in which applies to a group of geometric shapes in the file including the representation context group (model or plan with respective subgroups), the dimension count of the coordinate space the representation is modelled in, the precision or tolerance with which two points are assumed to be identical to each other and the TrueNorth or offset from the project's coordinate system (optional).
Example:	('Model',3,1.00000000000000E-6)

ifcm:ifcparameters:IfcSiUnit

Label:	IfcSiUnit
Use:	Optional, repeatable
Data Type:	String

Description:	Describes the units which apply to different measure in the IFC file, stating the dimension to be measured and the unit applied. This does not only apply to length, area and mass units but to all subjects included in the BIM process such as electricity, light, thermo dynamics or time.
Example:	(LENGTHUNIT,METRE)

ifcm - CountObjects

This group contains counts for various basic entities within the ifc object.

ifcm:countObjects:floorCount

Label:	floorCount
Use:	Optional, not repeatable
Data Type:	Integer
Description:	Describes the number of floors / storeys which are modelled in the IFC file.
Example:	4

ifcm:countObjects:roomCount

Label:	roomCount
Use:	Optional, not repeatable
Data Type:	Integer
Description:	Describes the number of rooms which are modelled in the IFC file.
Example:	64

ifcm:countObjects:wallCount

Label:	wallCount
Use:	Optional, not repeatable
Data Type:	Integer
Description:	Describes the number of walls which are modelled in the IFC file.
Example:	420

ifcm:countObjects:windowsCount

Label:	windowsCount
Use:	Optional, not repeatable
Data Type:	Integer
Description:	Describes the number of windows which are modelled in the IFC file.
Example:	105

ifcm:countObjects:doorCount

Label:	doorCount
Use:	Optional, not repeatable
Data Type:	Integer
Description:	Describes the number of doors which are modelled in the IFC file.
Example:	72

ifcm:countObjects:pipeCount

Label:	pipeCount
Use:	Optional, not repeatable
Data Type:	Integer
Description:	Describes the number of pipes which are modelled in the IFC file.
Example:	34

ifcm:countObjects:columnCount

Label:	columnCount
Use:	Optional, not repeatable
Data Type:	Integer
Description:	Describes the number of columns which are modelled in the IFC file.
Example:	60

ifcm:countObjects:numberOfComponents

Label:	numberOfComponents
Use:	Optional, not repeatable
Data Type:	Integer
Description:	Describes the total number IfcProduct subtypes which have been instantiated in the object (counts al doors, windows, roofs, walls, etc.).
Example:	11423

ifcm:countObjects:numberOfRelations

Label:	numberOfRelations
Use:	Optional, not repeatable
Data Type:	Integer
Description:	Describes the total number relations - internal as well as external - which are used in the IFC file.
Example:	5.218

ifcm:countObjects:numberOfActors

Label:	numberOfActors
Use:	Optional, not repeatable
Data Type:	Integer
Description:	Describes the number of actors which are listed in the IFC file.
Example:	5

ifcm - InformationMetric

This group contains elements which are based on counted and/or calculated values of entities.

ifcm:InformationMetric:numberOfEntityTypesUsed

Label:	numberOfEntitiesUsed
Use:	Optional, not repeatable

Data Type:	Integer
Description:	Describes the number of different entities which are used in the IFC file. Each different entity is only counted once, regardless of how many times it is instantiated in the file.
Example:	364

ifcm:InformationMetric:numberOfTotalEntitiesUsed

Label:	numberOfTotalEntitiesUsed
Use:	Optional, not repeatable
Data Type:	Integer
Description:	Gives the total count of entities which are used in the IFC file.
Example:	2954

ifcm:InformationMetric:optionalAttributes

Label:	optionalAttributes
Use:	Optional, not repeatable
Data Type:	Integer
Description:	Describes the percentage of OPTIONAL schema-level attributed which are provided with values in this file.
Example:	57

ifcm - Dependencies

This group contains information about internal or external dependencies of the IFC object. The element contained within is an aggregate of various entities.

ifcm:Dependencies:webResourceLink

Label:	webResourceLink
Use:	Optional, repeatable
Data Type:	String

Description:	Describes every web resource (URL) which the IFC links to, including from which entity type in the IFC file it was referenced from and how many times it was linked from that type.
Example:	(`http://www.armaturjonsson.no/novus/upload/article/rorprodukter/PDFror/classicweb.pdf`,IfcLabel,6); (`http://www.cpic.org.uk`,IfcClassificationReference,34)

6 Integration into an OAIS Compliant Digital Preservation System

While the previous sections focused on the methods and tools in an isolated manner, this chapter briefly discusses how the DURAARK methods and tools can be integrated into an existing OAIS compliant digital archive.

6.1 Information Packages between the OAIS and the Semantic Digital Archive

A first thing to consider regarding the integration into a digital preservation system is the package structure - different options have been presented in chapter 2, where for the DURAARK context an organization in a nested IE structure and Archival Information Collections (AICs) is favored.

While PREMIS has outlined the possibility of nested intellectual entities since its conception, no reference implementation of such a structure is known [19]. Furthermore, METS, which is frequently used for preservation system implementations in the library and archive world, provides complex and non-intuitive mechanisms to model nested IE structures. In the DURAARK context, we proposed a nested IE-structure as shown in Figure 2, where the “top-level IE” is the descriptive information of the physical asset itself. Values to describe the physical asset were presented in the physical asset elements of the *buildm* schema of chapter 5.1. It is to be expected that descriptive information about a building may change over time - the overall area changes when the building is extended as part of a modification, the city in the address may change when communities are combined and new city limits are formed, and even the geolocation can change when, e.g., historical buildings are moved from their original location to a museum. These changes of the descriptive metadata may be conducted manually or automatically and can be regarded a migration at the semantic level. Within the preservation context this is a preservation action, information about which needs to be captured. However, neither does METS foresee metadata about metadata to be captured, nor does PREMIS in v2.3 allow for a direct linking of agents and events to the intellectual entity layer. Currently, the metadata would have to be treated as a separate representation / file in order to allow preservation metadata to be captured about semantic migration processes [7]. This implementation detaches processes from the object itself, favoring a predominantly static and object-centric view of the digital artifact to be preserved. The relevance of this gap does not only pertain to process-driven objects such as representations of evolving architectural structures - take, for example, a digitized book whose physical representation is lost in a fire, now making the digital representation the only remaining version; or a 14th century fresco painting, which has been restored several times, putting digital photographic representations of the object in direct relation to the re-occurring restoration activities of the analogue original; or changing business processes, where data output needs to be put into direct relation to the procedural environment it stems from. Capturing these dependencies in the future will be improved via PREMIS v3.0 treating the intellectual entity as an object type, therefore allowing events and rights, and through these agents to link to the IE. In further support of this, version 3.0 will define semantic units for the IE object type [7]. An outlook to the DURAARK elements in the context of the PREMIS v3 data model can be seen in Figure 15. The METS Editorial Board is also currently proposing a new version of the standard - METS 2.0. The early stage proposal for community feedback includes a closer integration with PREMIS as well as moving descriptive metadata directly to the structural / fileGroup / file sections [14].

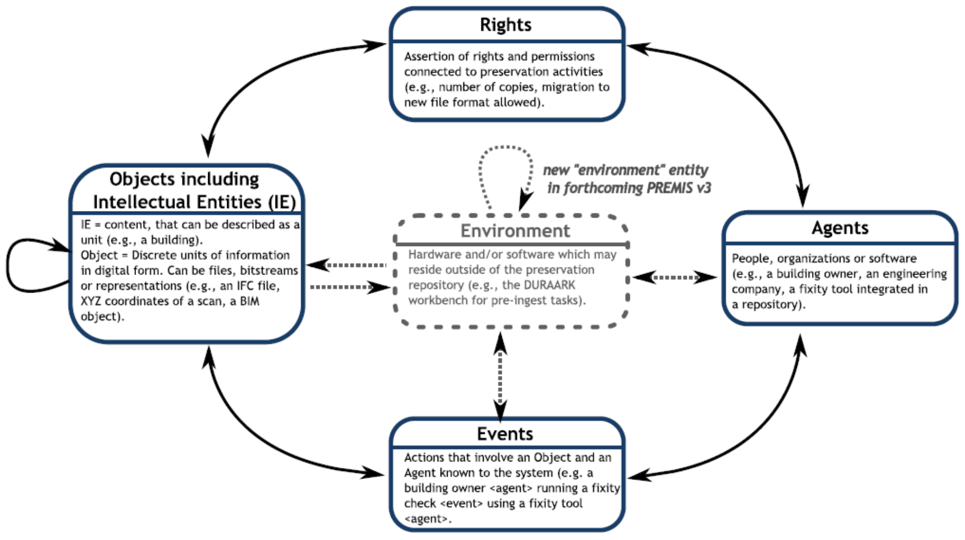

Rights
Assertion of rights and permissions connected to preservation activities (e.g., number of copies, migration to new file format allowed).
new "environment" entity in forthcoming PREMIS v3
Objects including Intellectual Entities (IE)
IE = content, that can be described as a unit (e.g., a building).
Object = Discrete units of information in digital form. Can be files, bitstreams or representations (e.g., an IFC file, XYZ coordinates of a scan, a BIM object).
Environment
Hardware and/or software which may reside outside of the preservation repository (e.g., the DURAARK workbench for pre-ingest tasks).
Agents
People, organizations or software (e.g., a building owner, an engineering company, a fixity tool integrated in a repository).
Events
Actions that involve an Object and an Agent known to the system (e.g. a building owner <agent> running a fixity check <event> using a fixity tool <agent>.

Figure 15: PREMIS v3 data model [7], put into the DURAARK context

The organization of archival information objects in archival information collections is the implementation favored by most off-the-shelf preservation systems. Rosetta, Archivematica and Preservica support the grouping of AIPs in collections. However, while some of the solutions version content / descriptive and preservation metadata captured at the AIP level, none versions the information at the collection level, regarding it rather as fixed and stable information. In-line with the AIC implementation, the aforementioned systems only support 1 IE per AIP [16][2][9]. While Rosetta, Archivemetica and Preservica all use XML structures to capture content and preservation information, the implementation choice between logical and physical AIP differs. Rosetta and Preservica both implement logical structures, while Archivematica implements a physical structure, packaging the objects in a 7z file. While the recommendation towards an AIC structure is directly derived from the role of the nested-IE structure to allow for an adequate capturing of the physical asset information, the choice between logical or physical structure remains one to be made by each institution, regardless of the content type which is being archived.

As off-the-shelf digital preservation systems currently do not support the extensive information capturing and versioning of the "top-level IE" as in the sense of the DURAARK physical asset entity, this needs to be addressed in a different way. The DURAARK semantic digital archive (SDA) stores a copy of each *buildm* object. Within the DURAARK workbench, the SDA can be leveraged to enrich the *buildm* with missing information using linked data information, for example a missing geolocation via geonames, if the address is known. To support arbitrary enrichment cases, a second rdf file stores non-*buildm*-relevant enriched information and may be passed on to the SIP created in the workbench and via the SIP to the digital preservation system and the AIP as well. This EnrichedData.RDF file allows the DURAARK workbench and the SDA to have a flexible approach towards user needs, while still having a stable set of basic description elements via the *buildm* schema. But what if changes to aspects of the building, such as its address or its area size, occur? The descriptive *buildm* information can be updated in three ways:

1. A user updates the *buildm* information directly.
2. A user deposits a new SIP to the archive, which matches the physical asset and shows updated elements in the *buildm* description.
3. *Buildm* elements which are connected to linked data sets, which in return are monitored through the SDA, are updated via the SDA as new information becomes available and is deemed relevant (e.g., address information).

To allow for these update cases to work, a connection between the SDA, the SIP and the AIP is given through a DURAARK physical asset ID contained in the *buildm*. In the AIP of the physical asset/digital object AIC, the OAIS may choose to keep the original *buildm* which was first deposited to the archive. However, as part of the integration between the OAIS and the SDA regular snapshots of the SDA are pulled as SIPs into the archive at either event-based occurrences (e.g., n number of relevant updates) or at fixed intervals (e.g., every 3 months). Since the SDA stores the *buildm* records for every deposit and is (a) aware of updates through linked data sources and is (b) contacted at every deposit process and is thus aware of user steered updates to the *buildm*, it always keeps an “up-to-date” copy of the descriptive metadata for the physical asset. The OAIS, on the other hand, stores the beginning of the audit trail, so to speak, directly at the top AIC level and can trace changes to this entity by comparing the stored information to the corresponding subsets of the SDA snapshots.

6.2 Logical Preservation of IFC-SPF and E57 in an Archival Setting

Tools to support the logical preservation of digital objects are the backbone of any digital preservation system. Ideally, tools should allow for as much automation as possible without loosing in granularity. (Semi-)automatic processing of digital objects in preservation workflows start with the file format identification process, which subsequently allows to route the objects to the correct tools for technical metadata extraction and, where desired, file format validation or risk extraction. Due to this pipeline file format identification tools shall support a wide variety of file formats. In adding E57 and IFC-SPF (as well as STEP-21 files) to the list of formats which are identifiable through standard digital preservation tools like DROID or fido, a logical preservation workflow for these two file formats has become more feasible, moving them from the “unknown long tail” to known entities in a preservation context. While granularity down to the file format version level is easy to achieve for E57, capturing structured text based file formats at the schema level is not necessarily common practice. The decision to capture a file format at the schema version level should be based on the benefit for subsequent preservation tasks. In the case of IFC, new entities are introduced with new schema versions and old entities may become deprecated, which in return may influence technical metadata extraction.

Furthermore, model view definitions typically depend on the IFC schema version, which in return may influence file format validation, if choosing to validate beyond the STEP-21 header. This clearly speaks for an identification at the schema instance level. In the current DURAARK setup, IFC-SPF is recognized on a super-level, agnostic of the schema, which becomes known at the technical metadata extraction layer. For the current definition of technical metadata candidates this solution works fine, however, the mid-term goal should be to turn the IFC-SPF PUID into a superclass for IFC-SPF schema PUIDs. Here, it needs to be considered if new descriptions for each addendum version of the IFC format are necessary, or if top-level versions, such as IFC2x3 for IFC2x3-TC1 suffice.

Technical metadata extraction shall deliver flexible output to allow the subsequent workflows to parse the information as required and to, e.g., use it to populate part of the significant properties. At the same time, technical metadata extraction is only as good as the description of the metadata set and as the data pool to extract from. The technical metadata extraction put forth the fact, that all E57 members of the DURAARK test data set are created with a FARO scanner. While FARO is undeniably one of the leading players in the global 3D scanning community, a wider sample set including other vendors may be beneficial, e.g., to check if other vendors export 2D images or weather data all the way through to the E57 file. While the IFC dataset was far more heterogeneous than the E57 dataset, more data shall prove helpful here as well. Especially regarding different actors involvement in the BIM object we would expect objects with a much higher information depth. Currently some IFC technical metadata candidates, such as IfcOwner, are put on hold because representative test data is missing. Nevertheless, significant progress have been made in technical metadata extraction - and as a by-product, several candidates were identified which can be extracted from the digital object and be used to populate the descriptive metadata elements of the digital and physical asset (see chapter 5).

Since the first analysis in D6.6.1, the libE57 reference implementation's e57validate tool has proven to be a valid choice for file format validation. The only potential problem lies in the speed of the validation process, especially for large files (as previously shown in Figure 7) - however, the experiment should be run in different environments to further determine if setup mechanisms can improve the performance. Unfortunately the situation is not as bright on the IFC-SPF front. The validation against the main model view definition does not seem to be practical. Here, two scenarios could be possible in an integration scenario: a validation against the STEP physical file format requirements seems a valuable choice from an archival viewpoint to verify a minimum renderability of the object as per the intact header and overarching structure. However, a suitable tool which allows for IFC-SPF validation against EXPRESS / SPF still remains to be found or developed. A second option is the definition of an archival IFC/A view definition, as planned in earlier stages of the project. Along these lines it needs to be noted that archival use cases differ widely and the question of the feasibility of a single "IFC/A" to fulfill all needs shall be monitored closely.

Simultaneously, the trend in the validation discussion amongst the digital preservation community seems to lead more and more from the original idea of validation meaning to validate against a standard - towards - validation against institutional / use case requirements[54].

A generic validation approach for IFC-SPF objects can be to validate against ISO10303-21 Step Physical File Format Requirements in a first step and towards respective MVDs, if available, in a second step. Unfortunately no STEP validator could be found so far which supports IFC files.

6.3 The Role of the Pre-Ingest Workbench

The DURAARK workbench, whose first prototype has been presented in D2.4, can be used in different integration options with an OAIS. Each implementation scenario brings implications about trust and control. There are three main scenarios:

1. It can fully reside on the OAIS side, being hosted and controlled by the respective institution. In that case, all preservation processes within the workbench, e.g., technical metadata extraction, can be considered trustworthy. The SIP produced by the workbench is equal to a SIP being produced by the OAIS, as the workbench becomes a fixed part of the OAIS.

2. It can fully reside on the producer side, with tasks like file format identification or technical metadata taking place and being made transparent to the user for control and correction. Upon transfer to the OAIS, the OAIS may choose to rerun the preservation processes, possibly based on institutional decisions depending on the

[54]See, e.g., recent efforts such as SCAPE's Flint `http://flint.openpreservation.org/` or the Swiss KOST-ECO'S KOST-Val tool `https://github.com/KOST-CECO/KOST-Val`

original data producer or other criteria. Some tools integrated into the workbench therefore run on both sides.

3. Instead of using the workbench, the OAIS has taken the DURAARK workbench's microservices and integrated those directly into the OAIS, therefore bypassing the workbench all together.

While combinations of these three integration options are possible, they are not to be explicitly listed here. The output as described in the previous chapters remains the same - regardless of the integration level of the Pre-Ingest Workbench.

7 Conclusion and DURAARK Objectives

This deliverable has shown how the gaps identified in D6.6.1 have been addressed throughout the project so far. For logical preservation, using the test set it could be proven that the file format identification processes work well. The progress of the extractors was described based ib using the tools to extract metadata from the DURAARK test set, simultaneously leading to a solid data basis to discuss the availability and structure of the technical metadata contained within the objects. Here, especially the E57 part of the dataset turned out to be very homogeneous, leading to a current identification process of further suitable test objects, ideally from non-FARO scanners to cover a wider base of sensor vendors. In the case of FARO it was particularly interesting to note that the company does not export available information sets, including any 2D images, into the E57 file.

Regarding the semantic gaps, the work taking place in WP3 was briefly described. While the *buildm* schema, described as part of the metadata gaps, is an important building block of the SDA and the entire semantic preservation aspect, a connection between the SDA tools and the OAIS will be integrated into upcoming prototypes. This deliverable furthermore described the efforts and results of the extension of the metadata schemas *buildm*, *ifcm* and *e57m*. A full description of the elements shall aid digital curation and preservation workers in their work with IFC and E57 files. Lastly, integration possibilities into an existing OAIS compliant digital preservation system were briefly discussed.

7.1 Impact and Outlook

The overall goal of the deliverable was to describe the current state of the DURAARK processes for ingest and storage into an existing OAIS compliant archive. This information is required for WP6 Task 6.3 (project month 25-30), which will conduct the ingest starting from the SIP generation in the DURAARK workbench into the existing digital preservation system Rosetta, which is a part of the OAIS implementation in place at TIB - the German National Library of Science and Technology. For the integration, the nested-IE structure will be realized through Rosetta's AIC function, accompanied through the SDA snapshot archiving, as presented in chapter 6. File format identification, technical metadata extraction and file format validation will take place inside the digital preservation system - however, Rosetta will save the original PREMIS output of the DURAARK toolbench as source metadata via the DURAARK workbench METS file. Standard processes, such as fixity checksuming and virus checking will also be conducted. Lastly, the end-to-end process will be demonstrated via search&retrieval use cases, which query the SDA for objects and request the objects from the digital preservation system via a deep link.

In addition to outlining the work for the ingest and storage process, this deliverable has identified a number of issues for the project to address in the near future. For file format identification, the question of whether PUIDs shall be assigned at a schema instance level needs to be explored further and discussed with the PRONOM maintainers The National Archives, UK (TNA). This includes turning the current IFC-SPF PUID into a superclass for further IFC schema dependent PUIDs. Regarding technical metadata extraction, some non-FARO files should be run against the E57 technical metadata extractor to see the support of various elements in other vendor's export routines. Regarding the lack of export of weather data and, more importantly, 2D data in the E57 objects, a request could be sent to FARO to include this information future export routines. The awareness of vendors towards both, the open file format standard as well as towards digital preservation requirements in the data should be raised early on. Simultaneously, minor issues in the E57 technical metadata extractor, which are inherited from the libe57 reference implementation, need to be addressed. The identification of candidates for *ifcm* also remains an ongoing task within the project. As new test datasets become available, and especially as IFC4 hits the stakeholders' systems, new objects should be evaluated towards their requirements and information depth. This does not only serve technical metadata extraction, but also semantic enrichment use case scenarios. These factors show that for both, IFC and E57 alike, a larger test data set is beneficial for two reasons: to gather further proof that assumptions made by the project so far are indeed correct and to continuously monitor the direction in which the use of 3D laser scans and building information models are evolving. Regarding descriptive metadata, the *buildm* development has been largely fed through experience with existing archival standards. With parallel research in WP7 is currently evaluating the user expecations from the facility maintenance domain in long-term archival, the state of *buildm* needs to be benchmarked against those expectations and experiences as well. Validation of both formats remains a challenging task. While the validation rules for E57 are straight-forward and clear canonical rules exist, the file size remains a challenging factor in leveraging validation processes in a scalable manner. For IFC-SPF, on the other hand, not file size but rather the unlimited complexity poses a problem in even achieving a canonical rule-set. Here, the idea of either on overarching or an institution specific IFC/A MVD as an archival guideline can be seen as an effective, yet laborsome task.

Lastly, all knowledge gathered about the sustainability and usability context of IFC-SPF and E57 so far - on the technical as well as on the organizational side - will be included in the analysis of significant property candidates, leading to the preservation planning study for the two digital formats.

References

[1] American National Standards Institute. Data dictionary - technical metadata for digital still images. Technical Report ANSI/NISO Z39.87-2006, ANSI/NISO, 2006.

[2] Artefactual. *Dataset preservation.* `https://www.archivematica.org/wiki/Dataset_preservation`.

[3] ASTM International. Standard specification for 3d imaging data exchange, version 1.0, 2013.

[4] I. Blümel. Documentation of PROBADO3D metadata for OAI-PMH interface. `http://3d.probado.igd.fraunhofer.de/Probado3DOAI`, October 2012.

[5] BuildingSmart Model Support Group. Industry Foundation Classes IFC4 Official Release. `http://www.buildingsmart-tech.org/ifc/IFC4/final/html/`, 2013.

[6] CCSDS. Reference Model for an Open Archival Information System (OAIS) - Magenta Book, 2012.

[7] A. Dappert. Proposed data model changes for PREMIS 3.0. In *Presentation held at PREMIS Implementation Fair, September 5th 2013, Lisbon, Portugal*, 2013.

[8] E-ARK Project. D3.1 - report on available best practises. Technical report, E-ARK European Archival Records and Knowledge, 2014.

[9] M. Evans. XIP and PREMIS. In *Presentation held at the PREMIS Implementation Fair 2012, October 2nd 2012, Toronto, Canada*, 2012.

[10] FARO Technology Inc. FARO laser scanner focus 3D manual, October 2011.

[11] FARO Technology Inc. SCENE 5.1 - user manual, October 2012.

[12] K. Fernie, D. Gavrillis, and S. Angelis. The CARARE metadata schema, v.2.0, 2013.

[13] N. Friesen and C. Lange. Linked Data und Digitale Bibliotheken. In T. Pellegrini, H. Sack, and S. Auer, editors, *Linked Enterprise Data*, X.media.press, pages 221–243. Springer Berlin Heidelberg, 2014.

[14] T. Habing. METS 2.0. In *Presentation held at the"METS Now, and Then ... Discussions of Current and Future Data Model" Workshop at Digital Libraries 2014, September 11th 2014, London, UK*, 2014.

[15] K.-H. Häfele, A. Geiger, and T. Liebich. Implementation guide for IFC header section version 1.0.2. Technical report, buildingSmart, 2008.

[16] M. Hahn. Recommendations for preservation-aware digital object model. Technical report, SCAPE Project, 2014.

[17] ISO/TC 184/SC 4. ISO 16739:2013 - Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries, 2013.

[18] D. M. Jones, editor. *3D Laser Scanning for Heritage.* English Heritage, second edition edition, October 2011.

[19] M. Lindlar. DURAARK contribution to PREMIS Implementation Fair. Blog Entry on DURAARK website `http://duraark.eu/duraark-contribution-to-premis-implementation-fair/`, October 2014.

[20] PREMIS Editorial Committee. PREMIS Data Dictionary for Preservation Metadata, version 2.2, 2012.

[21] RLG. Automatic exposure: Capturing technical metadata for digital still images. White Paper, 2004.

[22] M. Smith. Final Report for the MIT FACADE Project: October 2006 - August 2009. Massachusets Institute of Technology, 2009.

[23] M. Tamke, M. Jensen Myrup, J. Beetz, T. Krijnen, and D. Edvardsen. Building information deduced - state and potetnials for information query in building information modelling. In *Proceedings of the 31st International Conference on Education and research in Computer Aided Architectural Design in Europe*, 2014.

[24] The Library of Congress. textMD: technical metadata for text. `http://www.loc.gov/standards/textMD/elementSet/index.html`.

[25] University Library at University of Illinois at Urbana-Champaign. *Best Practises for Technical Metadata.* `http://www.library.illinois.edu/dcc/bestpractices/chapter_10_technicalmetadata.html`.

8 Annex

8.1 *Buildm* Mapping to Existing Descriptive Metadata Schemas

buildm	**3.1 Draft**	**Historic Buildings**[55]	**MIT FACADE PIM**[56]	**PROBADO3D metadata core**	**Dublin Core**[57]	**CARARE**
DigitalObject:creator			dcterms:creator	CONTRIBUTOR	dc:creator (digital)	Digital Resource: Actors
DigitalObject:filename		file name of raw data	dcterms:title	TITLE / MODELFILE	dc:title (digital)	Digital Resource: Appellation
DigitalObject: dateCreated		date of capture	dcterms:created	DATES	dc:date (digital)	Digital Resource: Created
DigitalObject:isPartOf			dcterms:isPartOf	RELATION		Digital Resource: Is Part Of
DigitalObject:hasPart					RELATION	Digital Resource: Has Part
DigitalObject:Description				MODELDESCRIPTION	dc:description (digital)	Digital Resource: Description
DigitalObject:Identifier				MODELINFOID	dc:identifier (digital)	Digital Resource: Record information
DigitalObject:Format			dcterms:format	MODELFILE	dc:format (digital)	Digital Resource: Format
DigitalObject:hasType				MODELFILE	dc:type (digital)	Digital Resource: Type
DigitalObject: hasFormatDetails						Digital Resource: Format Details
DigitalObject:license			dcterms:rights	LICENSE	dc:rights (digital)	Rights
DigitalObject:provenance						
DigitalObject:unitCode						
DigitalObject:event				EVENT		
DigitalObject:levelOfDetail				MODELATTRIBUTE		
PhysicalAsset:name	Name	monument name	dcterms:title	TITLE	dc:title (physical)	Heritage Asset: Appellation
PhysicalAsset:location	Location		pim:location	PLACE	dc:coverage (physical)	Heritage Asset: Spatial
PhysicalAsset:latitude	Location		pim:location	PLACE	dc:coverage (physical)	Heritage Asset: Spatial
PhysicalAsset:longitude	Location		pim:location	PLACE	dc:coverage (physical)	Heritage Asset: Spatial
PhysicalAsset:address						Heritage Asset: Spatial
PhysicalAsset: postalCodeStart						Heritage Asset: Spatial
PhysicalAsset: postalCodeEnd						Heritage Asset: Spatial
PhysicalAsset: postOfficeBoxNumer						Heritage Asset: Spatial
PhysicalAsset: addressRegion						Heritage Asset: Spatial

[55]complete name:Historic Buildings and Monuments Commission for England (mandatory fields for 3D laser scans)

[56]MIT FACADE PIM combines derived values of Dublin Core with own created PIM values

[57]Dublin Core values may apply to digital object and physical asset alike. The mapping works under the assumption that physical and digital assets are described in separate DC records. We have indicated in paranthesis, which of the level - physical or digital - the mapping pertains to.

PhysicalAsset: addressLocality						Heritage Asset: Spatial
PhysicalAsset:Description			dcterms:description		dc:description (physical)	Heritage Asset: Description
PhysicalAsset:Identifier					dc:identifier (physical)	Heritage Asset: Record Information
PhysicalAsset:architect			dcterms:creator		dc:creator (physical)	Heritage Asset: Actors
PhysicalAsset:contributor			dcterms: contributor	CONTRIBUTOR	dc:contributor (physical)	Heritage Asset: Actors
PhysicalAsset:owner	Owner		dcterms:owner	CONTRIBUTOR		Heritage Asset: Actors
PhysicalAsset: completionDate	Construction Date				dc:date (physical)	
PhysicalAsset:startDate					dc:date (physical)	
PhysicalAsset: constructionTime	Construction Time				dc:date (physical)	
PhysicalAsset: rebuildingDate	Modification Date				dc:date (physical)	
PhysicalAsset: modificationDetails						
PhysicalAsset:buildingArea	Building Area			MODELATTRIBUTE		Heritage Asset: Characters
PhysicalAsset:function	Function		pim:buildingType	MODELATTRIBUTE		Heritage Asset: Characters
PhysicalAsset: architecturalStyle			pim: architecturalStyle			
PhysicalAsset:rightsDetails			dcterms:rights		dc:rights (physical)	Heritage Asset: Rights
PhysicalAsset:cost			pim:cost			
PhysicalAsset:floorCount	Number of Floors			MODELATTRIBUTE		
PhysicalAsset: numberOfRooms	Number of Rooms			MODELATTRIBUTE		

Table 128: *buildm* mapping

8.2 *Buildm* RDF Schema - Current DURAARK Draft Status

```
@prefix dc: <http://purl.org/dc/terms/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix schema: <http://schema.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix dc11: <http://purl.org/dc/elements/1.1/> .

<http://data-observatory.org/buildm/>
  dc:title ""@en ;
  rdfs:comment "The 'buildm' schema has been developed in context of
     the DURAARK project."@en ;
  owl:versionInfo "2.2, December 11, 2014" ;
  dc:creator [
    foaf:name "Ujwal Gadiraju" ;
    foaf:organization "L3S Research Center" ;
    foaf:mbox "gadiraju@l3s.de"
  ], [
    foaf:name "Besnik Fetahu" ;
    foaf:organization "L3S Research Center" ;
    foaf:mbox "fetahu@l3s.de"
  ], [
    foaf:name "Stefan Dietze" ;
    foaf:organization "L3S Research Center" ;
    foaf:mbox "dietze@l3s.de"
  ], [
    foaf:name "Michelle Lindlar" ;
    foaf:organization "TIB" ;
    foaf:mbox "Michelle.Lindlar@tib.uni-hannover.de"
  ], [
    foaf:name "Michael Panitz" ;
    foaf:organization "TIB" ;
    foaf:mbox "Michael.Panitz@tib.uni-hannover.de"
  ] .

<http://data-observatory.org/buildm/DigitalObject>
  a rdfs:Class ;
  rdfs:label "Digital Object" ;
  rdfs:subClassOf schema:CreativeWork .

<http://data-observatory.org/buildm/PhysicalAsset>
  a rdfs:Class ;
  rdfs:label "Physical Asset" ;
  rdfs:subClassOf <http://dbpedia.org/ontology/ArchitecturalStructure>
     .
```

```
<http://data-observatory.org/buildm/creator>
  a rdf:Property ;
  rdfs:label "creator" ;
  rdfs:comment "The creator/author of this DigitalObject. This is the
      same as the Author property for CreativeWork." ;
  rdfs:domain <http://data-observatory.org/buildm/#DigitalObject> ;
  rdfs:range [
    a owl:Class ;
    owl:unionOf (
     schema:Person
     schema:Organization
   )
  ] ;
  owl:equivalentProperty schema:creator .

schema:Person a owl:Class .
schema:Organization a owl:Class .
<http://data-observatory.org/buildm/name>
  a rdf:Property ;
  rdfs:label "Filename", "name" ;
  rdfs:comment "The filename of this DigitalObject.", "The title or
      name of this Physical Asset." ;
  rdfs:domain <http://data-observatory.org/buildm/#DigitalObject>,
      <http://data-observatory.org/buildm/#PhysicalAsset> ;
  rdfs:range schema:Text ;
  owl:equivalenProperty schema:name ;
  owl:equivalentProperty schema:name .

<http://data-observatory.org/buildm/dateCreated>
  a rdf:Property ;
  rdfs:label "dateCreated" ;
  rdfs:comment "The Date of Creation of this DigitalObject." ;
  rdfs:domain <http://data-observatory.org/buildm/#DigitalObject> ;
  rdfs:range schema:Date ;
  owl:equivalentProperty schema:dateCreated .

<http://data-observatory.org/buildm/location>
  a rdf:Property ;
  rdfs:label "location" ;
  rdfs:comment "The location of this Physical Asset." ;
  rdfs:domain <http://data-observatory.org/buildm/#PhysicalAsset> ;
  rdfs:range geo:SpatialThing ;
  owl:equivalentProperty geo:location .

<http://data-observatory.org/buildm/latitude>
  a rdf:Property ;
  rdfs:label "latitude" ;
  rdfs:comment "The WGS84 latitude of this Physical Asset (decimal
      degrees)." ;
  rdfs:domain <http://data-observatory.org/buildm/#PhysicalAsset> ;
  owl:equivalentProperty geo:lat .

<http://data-observatory.org/buildm/longitude>
```

```
  a rdf:Property ;
  rdfs:label "longitude" ;
  rdfs:comment "The WGS84 longitude of this Physical Asset (decimal
      degrees)." ;
  rdfs:domain <http://data-observatory.org/buildm/#PhysicalAsset> ;
  owl:equivalentProperty geo:long .

<http://data-observatory.org/buildm/streetAddress>
  a rdf:Property ;
  rdfs:label "streetAdress" ;
  rdfs:comment "The street address corresponding to this Physical
      Asset." ;
  rdfs:domain <http://data-observatory.org/buildm/#Physical Asset> ;
  rdfs:range schema:streetAddress ;
  owl:equivalentProperty schema:streetAddress .

<http://data-observatory.org/buildm/postalCodeStart>
  a rdf:Property ;
  rdfs:label "postalCode" ;
  rdfs:comment """The Postal Code corresponding to the location of
      this Physical Asset. If there is a range of postal codes, for
      instance within a gated community consisting of several
      buildings, then this represents the starting range of the postal
      codes.""" ;
  rdfs:domain <http://data-observatory.org/buildm/#PhysicalAsset> ;
  rdfs:range schema:Text ;
  rdfs:subPropertyOf schema:postalCode .

<http://data-observatory.org/buildm/postalCodeEnd>
  a rdf:Property ;
  rdfs:label "postalCode" ;
  rdfs:comment "The last postal code in the postal codes range
      corresponding to this physical asset." ;
  rdfs:domain <http://data-observatory.org/buildm/#PhysicalAsset> ;
  rdfs:range schema:Text ;
  rdfs:subPropertyOf schema:postalCode .

<http://data-observatory.org/buildm/postOfficeBoxNumber>
  a rdf:Property ;
  rdfs:label "postOfficeBoxNumber" ;
  rdfs:comment "The post office box number corresponding to the PO box
      address of this Physical Asset." ;
  rdfs:domain <http://data-observatory.org/buildm/#PhysicalAsset> ;
  rdfs:range schema:Text ;
  owl:equivalentProperty schema:postOfficeBoxNumber .

<http://data-observatory.org/buildm/addressRegion>
  a rdf:Property ;
  rdfs:label "addressRegion" ;
  rdfs:comment "The region corresponding to the location of this
      Physical Asset." ;
  rdfs:domain <http://data-observatory.org/buildm/#PhysicalAsset> ;
  rdfs:range schema:Text ;
  owl:equivalentProperty schema:addressRegion .

<http://data-observatory.org/buildm/addressLocality>
  a rdf:Property ;
  rdfs:label "addressLocality" ;
  rdfs:comment "The locality (town) corresponding to the location of
     this Physical Asset." ;
  rdfs:domain <http://data-observatory.org/buildm/#PhysicalAsset> ;
  rdfs:range schema:Text ;
  owl:equivalentProperty schema:addressLocality .

<http://data-observatory.org/buildm/addressCountry>
  a rdf:Property ;
  rdfs:label "addressCountry" ;
  rdfs:comment "The country corresponding to the location of this
     Physical Asset." ;
  rdfs:domain <http://data-observatory.org/buildm/#PhysicalAsset> ;
  rdfs:range schema:Country ;
  owl:equivalentProperty schema:addressCountry .

<http://data-observatory.org/buildm/isPartOf>
  a rdf:Property ;
  rdfs:label "isPartOf" ;
  rdfs:comment "This represents that Digital Object is a part of the
     other Digital Object. For instance, another scan or plan." ;
  rdfs:domain <http://data-observatory.org/buildm/#DigitalObject> ;
  rdfs:range <http://data-observatory.org/buildm/#DigitalObject> ;
  rdfs:subPropertyOf dc:isPartOf ;
  owl:inverseOf <http://data-observatory.org/buildm/hasPart> .

<http://data-observatory.org/buildm/hasPart>
  a rdf:Property ;
  rdfs:label "hasPart" ;
  rdfs:comment "This represents a part of a Digital Object. For
     instance, another scan or plan." ;
  rdfs:domain <http://data-observatory.org/buildm/#DigitalObject> ;
  rdfs:range <http://data-observatory.org/buildm/#DigitalObject> ;
  rdfs:subPropertyOf dc:hasPart ;
  owl:inverseOf <http://data-observatory.org/buildm/isPartOf> .

<http://data-observatory.org/buildm/description>
  a rdf:Property ;
  rdfs:label "description" ;
  rdfs:comment "A description of this Digital Object or Physical
     Asset." ;
  rdfs:domain [
    a owl:Class ;
    owl:unionOf (
     <http://data-observatory.org/buildm/#DigitalObject>
     <http://data-observatory.org/buildm/#PhysicalAsset>
   )
  ] ;
  rdfs:range schema:Text ;
  owl:equivalentProperty schema:description .

<http://data-observatory.org/buildm/#DigitalObject> a owl:Class .
<http://data-observatory.org/buildm/#PhysicalAsset> a owl:Class .
<http://data-observatory.org/buildm/identifier>
  a rdf:Property ;
  rdfs:label "identifier" ;
  rdfs:comment "A unique identifier (UUID) that represents either the
      Digital Object or Physical Asset." ;
  rdfs:domain [
    a owl:Class ;
    owl:unionOf (
     <http://data-observatory.org/buildm/#DigitalObject>
     <http://data-observatory.org/buildm/#PhysicalAsset>
   )
  ] ;
  rdfs:range xsd:string ;
  owl:equivalentProperty dc11:identifier .

<http://data-observatory.org/buildm/format>
  a rdf:Property ;
  rdfs:label "format" ;
  rdfs:comment "The media type format of this DigitalObject." ;
  rdfs:domain <http://data-observatory.org/buildm/#DigitalObject> ;
  rdfs:range dc:MediaTypeOrExtent ;
  owl:equivalentProperty dc:format .

<http://data-observatory.org/buildm/hasType>
  a rdf:Property ;
  rdfs:label "hasType" ;
  rdfs:comment "The type of this DigitalObject (e.g., scan, plan)." ;
  rdfs:domain <http://data-observatory.org/buildm/#DigitalObject> ;
  rdfs:range schema:Text .

<http://data-observatory.org/buildm/hasFormatDetails>
  a rdf:Property ;
  rdfs:label "hasFormatDetails" ;
  rdfs:comment "Additional information about the Digital Object, e.g.
      encoding or serialization." ;
  rdfs:domain <http://data-observatory.org/buildm/#DigitalObject> ;
  rdfs:range schema:Text .

<http://data-observatory.org/buildm/license>
  a rdf:Property ;
  rdfs:label "license" ;
  rdfs:comment "A license document that applies to this Digital
      Object, indicated by URL." ;
  rdfs:domain <http://data-observatory.org/buildm/#DigitalObject> ;
  rdfs:range schema:URL ;
  owl:equivalentProperty schema:license .

<http://data-observatory.org/buildm/provenance>
  a rdf:Property ;
  rdfs:label "provenance" ;
  rdfs:comment "A statement of any changes in ownership and custody of
      the Digital Object since its creation that are significant for
      its authenticity, integrity, and interpretation." ;
  rdfs:domain <http://data-observatory.org/buildm/#DigitalObject> ;
  rdfs:range dc:ProvenanceStatement ;
  owl:equivalentProperty dc:provenance .

<http://data-observatory.org/buildm/unitCode>
  a rdf:Property ;
  rdfs:label "unitCode" ;
  rdfs:comment "The unit of measurement given using the UN/CEFACT
      Common Code (3 characters). This determines in which unit
      properties corresponding to the Digital Object are entered." ;
  rdfs:domain <http://data-observatory.org/buildm/#DigitalObject> ;
  rdfs:range schema:Text ;
  owl:equivalentProperty schema:unitCode .

<http://data-observatory.org/buildm/Event>
  a rdf:Property ;
  rdfs:label "Event" ;
  rdfs:comment "Field for additional information about what the
      Digital Object was used for, such as presentations or
      competitions (e.g., Cadmesse 2014)." ;
  rdfs:domain <http://data-observatory.org/buildm/#DigitalObject> ;
  rdfs:range schema:Text ;
  owl:equivalentProperty schema:Event .

<http://data-observatory.org/buildm/architect>
  a rdf:Property ;
  rdfs:label "architect" ;
  rdfs:comment "Architect of this Physical Asset." ;
  rdfs:domain <http://data-observatory.org/buildm/#PhysicalAsset> ;
  rdfs:range <http://dbpedia.org/ontology/Architect> ;
  owl:equivalentProperty <http://dbpedia.org/ontology/architect> .

<http://data-observatory.org/buildm/contributor>
  a rdf:Property ;
  rdfs:label "contributor" ;
  rdfs:comment "A person who contributed in the construction of this
      Physical Asset. For example, the structural engineer, builder,
      and so on." ;
  rdfs:domain <http://data-observatory.org/buildm/#PhysicalAsset> ;
  rdfs:range schema:Person .

<http://data-observatory.org/buildm/owner>
  a rdf:Property ;
  rdfs:label "owner" ;
  rdfs:comment "Person or Organization that owns this Physical Asset."
      ;
  rdfs:domain <http://data-observatory.org/buildm/#PhysicalAsset> ;
  rdfs:range [
    a owl:Class ;
    owl:unionOf (
     schema:Person
     schema:Organization
   )
  ] .

<http://data-observatory.org/buildm/completionDate>
  a rdf:Property ;
  rdfs:label "completionDate" ;
  rdfs:comment "Year when the construction phase of this Physical
      Assest was finished (e.g., 2001)." ;
  rdfs:domain <http://data-observatory.org/buildm/#PhysicalAsset> ;
  rdfs:range xsd:integer ;
  owl:equivalentProperty <http://dbpedia.org/property/completionDate> .

<http://data-observatory.org/buildm/startDate>
  a rdf:Property ;
  rdfs:label "startDate" ;
  rdfs:comment "Year when the construction phase of this Physical
      Assest began (e.g., 1898)." ;
  rdfs:domain <http://data-observatory.org/buildm/#PhysicalAsset> ;
  rdfs:range xsd:integer ;
  owl:equivalentProperty <http://dbpedia.org/property/startDate> .

<http://data-observatory.org/buildm/constructionTime>
  a rdf:Property ;
  rdfs:label "startDate" ;
  rdfs:comment "The duration of construction of the building in days."
      ;
  rdfs:domain <http://data-observatory.org/buildm/#PhysicalAsset> ;
  rdfs:range xsd:integer .

<http://data-observatory.org/buildm/rebuildingDate>
  a rdf:Property ;
  rdfs:label "rebuildingDate" ;
  rdfs:comment "Year when the rebuilding phase of this Physical Assest
      began (e.g., 2010)." ;
  rdfs:domain <http://data-observatory.org/buildm/#PhysicalAsset> ;
  rdfs:range xsd:date ;
  owl:equivalentProperty <http://dbpedia.org/property/rebuildingDate> .

<http://data-observatory.org/buildm/modificationDetails>
  a rdf:Property ;
  rdfs:label "modificationDetails" ;
  rdfs:comment "Explanation of the modification of this Physical
      Asset." ;
  rdfs:domain <http://data-observatory.org/buildm/#PhysicalAsset> ;
  rdfs:range xsd:string .

<http://data-observatory.org/buildm/buildingArea>
  a rdf:Property ;
  rdfs:label "buildingArea" ;
  rdfs:comment "Gross floor area referring to the total floor area
```

```
        inside the building (Physical Asset) envelope, including the
        external walls, and excluding the roof." ;
    rdfs:domain <http://data-observatory.org/buildm/#PhysicalAsset> ;
    rdfs:range xsd:string .

  <http://data-observatory.org/buildm/function>
    a rdf:Property ;
    rdfs:label "function" ;
    rdfs:comment "Current or intended use of the Physical Asset." ;
    rdfs:domain <http://data-observatory.org/buildm/#PhysicalAsset> ;
    rdfs:range xsd:string .

  <http://data-observatory.org/buildm/architecturalStyle>
    a rdf:Property ;
    rdfs:label "architecturalStyle" ;
    rdfs:comment "Architectural Style of the Physical Asset." ;
    rdfs:domain <http://data-observatory.org/buildm/#PhysicalAsset> ;
    rdfs:range xsd:string ;
    owl:equivalentProperty
        <http://dbpedia.org/ontology/architecturalStyle> .

  <http://data-observatory.org/buildm/rightsDetails>
    a rdf:Property ;
    rdfs:label "rightsDetails" ;
    rdfs:comment "Information about rights (e.g. copyrights, license
        information) related to the Physical Asset (e.g.,K\"olner
        Dombauverwaltung is the responsible actor for the cathedral's
        preservation)." ;
    rdfs:domain <http://data-observatory.org/buildm/#PhysicalAsset> ;
    rdfs:range xsd:string .

  <http://data-observatory.org/buildm/cost>
    a rdf:Property ;
    rdfs:label "cost" ;
    rdfs:comment "Financial efforts, that were needed for realizing the
        construction of the Physical Asset in USD." ;
    rdfs:domain <http://data-observatory.org/buildm/#PhysicalAsset> ;
    rdfs:range xsd:double ;
    owl:equivalentProperty <http://dbpedia.org/ontology/cost> .

  <http://data-observatory.org/buildm/levelOfDetail>
    a rdf:Property ;
    rdfs:label "levelOfDetail" ;
    rdfs:comment "The level of detail of the Digital Object." ;
    rdfs:domain <http://data-observatory.org/buildm/#DigitalObject> ;
    rdfs:range xsd:String .

  <http://data-observatory.org/buildm/floorCount>
    a rdf:Property ;
    rdfs:label "floorCount" ;
    rdfs:comment "The number of floors that this Physical Asset
        contains." ;
    rdfs:domain <http://data-observatory.org/buildm/#PhysicalAsset> ;
    rdfs:range xsd:positiveInteger ;
    owl:equivalentProperty <http://dbpedia.org/ontology/floorCount> .

  <http://data-observatory.org/buildm/numberOfRooms>
    a rdf:Property ;
    rdfs:label "numberOfRooms" ;
    rdfs:comment "The number of rooms that this Physical Asset
        contains." ;
    rdfs:domain <http://data-observatory.org/buildm/#PhysicalAsset> ;
    rdfs:range xsd:positiveInteger .
```

Listing 1: RDF representation of the buildm data dictionary

8.3 *Buildm* XML Schema Definition - Current DURAARK Draft Status

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- Mit XMLSpy v2015 sp2 (x64) (http://www.altova.com) von Michael
    Panitz (TIB Hannover) bearbeitet -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:vc="http://www.w3.org/2007/XMLSchema-versioning"
    elementFormDefault="qualified" attributeFormDefault="unqualified"
    vc:minVersion="1.1">
 <xs:element name="buildm">
  <xs:annotation>
   <xs:documentation>DURAARK - descriptive metadata schema for
       building information </xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:sequence>
    <xs:element name="physicalAsset">
     <xs:annotation>
      <xs:documentation>Descriptive metadata about the tangible
          building.</xs:documentation>
     </xs:annotation>
     <xs:complexType>
      <xs:sequence>
       <xs:element name="Identifier" type="xs:string"
           maxOccurs="unbounded">
        <xs:annotation>
         <xs:documentation>A nonambiguous reference of the phyiscal
             asset within a given context.</xs:documentation>
        </xs:annotation>
       </xs:element>
       <xs:element name="name" type="xs:string" maxOccurs="unbounded">
        <xs:annotation>
         <xs:documentation>Title or name of the building, usually
             consisting of a combination of function and
             location.</xs:documentation>
        </xs:annotation>
       </xs:element>
       <xs:element name="latitude" type="xs:string">
        <xs:annotation>
         <xs:documentation>The latitude of the physical asset's
             location in decimal degrees.</xs:documentation>
        </xs:annotation>
       </xs:element>
       <xs:element name="longitude" type="xs:string">
        <xs:annotation>
         <xs:documentation>The longitutde of the physical asset's
             location in decimal degrees.</xs:documentation>
        </xs:annotation>
       </xs:element>
       <xs:element name="owner" type="xs:string" minOccurs="0"
           maxOccurs="unbounded">
        <xs:annotation>
         <xs:documentation>Person or organization who owns the
             physical asset. The element may be repeated to described
             different owners.
</xs:documentation>
        </xs:annotation>
       </xs:element>
       <xs:element name="buildingArea" type="xs:string" minOccurs="0">
        <xs:annotation>
         <xs:documentation>Total floor area inside the building,
             including the measuring unit.</xs:documentation>
        </xs:annotation>
       </xs:element>
       <xs:element name="floorCount" type="xs:integer" minOccurs="0">
        <xs:annotation>
         <xs:documentation>Total number of floors of the physical
             asset, including basement, sub-basement, ground and top
             levels.</xs:documentation>
        </xs:annotation>
       </xs:element>
       <xs:element name="numberOfRooms" type="xs:integer"
           minOccurs="0">
        <xs:annotation>
         <xs:documentation>Total number of rooms of the physical
             asset.</xs:documentation>
        </xs:annotation>
       </xs:element>
       <xs:element name="function" type="xs:string" minOccurs="0"
           maxOccurs="unbounded">
        <xs:annotation>
         <xs:documentation>Current or intended use of the Physical
             Asset.</xs:documentation>
        </xs:annotation>
       </xs:element>
       <xs:element name="architecturalStyle" type="xs:string"
           minOccurs="0" maxOccurs="unbounded">
        <xs:annotation>
         <xs:documentation>Architectural Style of the Physical
             Asset.</xs:documentation>
        </xs:annotation>
       </xs:element>
       <xs:element name="description" type="xs:string" minOccurs="0"
           maxOccurs="unbounded">
        <xs:annotation>
         <xs:documentation>A description of the physical asset, e.g.
             to give historical background or further describe the
             status.</xs:documentation>
        </xs:annotation>
       </xs:element>
       <xs:element name="location" type="xs:string" minOccurs="0">
```

```
 <xs:annotation>
  <xs:documentation>A general description of the Physical
      Asset's location.</xs:documentation>
 </xs:annotation>
</xs:element>
<xs:element name="streetAddress" type="xs:string" minOccurs="0">
 <xs:annotation>
  <xs:documentation>The street address corresponding to the
      Physical Asset.</xs:documentation>
 </xs:annotation>
</xs:element>
<xs:element name="postalCodeStart" type="xs:string"
   minOccurs="0">
 <xs:annotation>
  <xs:documentation>The postal code which corresponds to the
      location of the physical asset. If the physical asset is
      in a range of postal codes, for example if it describes a
      large appartment complex which spans over several postal
      codes, then this element marks the starting value of the
      postal code range. </xs:documentation>
 </xs:annotation>
</xs:element>
<xs:element name="postalCodeEnd" type="xs:string" minOccurs="0">
 <xs:annotation>
  <xs:documentation>If an address of a physical asset spans
      over several postal codes, e.g. for a large appartment
      complex, the starting postal code of the range is noted in
      postalCodeStart and the end of the range is noted in
      postalCodeEnd. In the case of addresses which just have
      one postal code, this value is empty.</xs:documentation>
 </xs:annotation>
</xs:element>
<xs:element name="postOfficeBoxNumber" type="xs:string"
   minOccurs="0">
 <xs:annotation>
  <xs:documentation>If the mailing address of the physical
      asset includes a post office box number, this number is
      described in this field.</xs:documentation>
 </xs:annotation>
</xs:element>
<xs:element name="addressRegion" type="xs:string" minOccurs="0">
 <xs:annotation>
  <xs:documentation>The region corresponding to the location of
      the physical asset, such as a state, province or area
      designation.</xs:documentation>
 </xs:annotation>
</xs:element>
<xs:element name="postalLocality" type="xs:string"
   minOccurs="0">
 <xs:annotation>
  <xs:documentation>The town / locality corresponding to the
      location of the physical asset.</xs:documentation>
 </xs:annotation>
```

```
</xs:element>
<xs:element name="architect" type="xs:string" minOccurs="0"
   maxOccurs="unbounded">
 <xs:annotation>
  <xs:documentation>The architect(s) of the physical
     asset.</xs:documentation>
 </xs:annotation>
</xs:element>
<xs:element name="contributor" type="xs:string" minOccurs="0"
   maxOccurs="unbounded">
 <xs:annotation>
  <xs:documentation>A person who contributed to the
     construction of the physical asset, e.g. the structural
     engineer or stone mason. </xs:documentation>
 </xs:annotation>
</xs:element>
<xs:element name="startDate" type="xs:integer" minOccurs="0">
 <xs:annotation>
  <xs:documentation>Year when the construction phase of the
     physical asset began.</xs:documentation>
 </xs:annotation>
</xs:element>
<xs:element name="completionDate" type="xs:integer"
   minOccurs="0">
 <xs:annotation>
  <xs:documentation>Year when the construction phase of the
     physical asset was completed.</xs:documentation>
 </xs:annotation>
</xs:element>
<xs:element name="constructionTime" type="xs:integer"
   minOccurs="0">
 <xs:annotation>
  <xs:documentation>Duration of the construction phase of the
     physical asset in days.</xs:documentation>
 </xs:annotation>
</xs:element>
<xs:element name="rebuildingDate" type="xs:date" minOccurs="0"
   maxOccurs="unbounded">
 <xs:annotation>
  <xs:documentation>Year when the rebuilding date of the
     physical asset began.</xs:documentation>
 </xs:annotation>
</xs:element>
<xs:element name="modificationDetails" type="xs:string"
   minOccurs="0" maxOccurs="unbounded">
 <xs:annotation>
  <xs:documentation>Explanation of the modification of the
     physical asset.</xs:documentation>
 </xs:annotation>
</xs:element>
<xs:element name="cost" type="xs:double" minOccurs="0"
   maxOccurs="unbounded">
 <xs:annotation>
```

```
      <xs:documentation>Financial efforts which were needed for
          realizing the constructino of the physical asset, in USD.
          </xs:documentation>
     </xs:annotation>
    </xs:element>
    <xs:element name="rightsDetails" type="xs:string" minOccurs="0"
        maxOccurs="unbounded">
     <xs:annotation>
      <xs:documentation>Information about rights, such as
          copyrights, license information or regulatory requirements
          related to the Physical Asset.</xs:documentation>
     </xs:annotation>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="digitalObject" maxOccurs="unbounded">
  <xs:annotation>
   <xs:documentation>Descriptive metadata about the digital object
       containing a representation of the physical asset, such as a
       plan or a scan.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Identifier" type="xs:string"
        maxOccurs="unbounded">
     <xs:annotation>
      <xs:documentation>A nonambigious reference of the digital
          object within a given context. Where possible, formal
          identification systems should be used.</xs:documentation>
     </xs:annotation>
    </xs:element>
    <xs:element name="creator" type="xs:string"
        maxOccurs="unbounded">
     <xs:annotation>
      <xs:documentation>Creator/author of the digital
          object.</xs:documentation>
     </xs:annotation>
    </xs:element>
    <xs:element name="name" type="xs:string" maxOccurs="unbounded">
     <xs:annotation>
      <xs:documentation>The name of the digital object. This may be
          the file name or reflect on the data which is inside the
          object.</xs:documentation>
     </xs:annotation>
    </xs:element>
    <xs:element name="dateCreated" type="xs:dateTime">
     <xs:annotation>
      <xs:documentation>The date the digital object was
          created.</xs:documentation>
     </xs:annotation>
    </xs:element>
```

```
<xs:element name="isPartOf" type="xs:string" minOccurs="0"
   maxOccurs="unbounded">
 <xs:annotation>
  <xs:documentation>Links the digital object to an overaching
     digital object it is a part of, e.g. in the case of plans
     for different fllors the object may link to an overall
     plan view of all the physical asset's rooms. The
     corresponding overaching object shall be identified
     through it's identifier.</xs:documentation>
 </xs:annotation>
</xs:element>
<xs:element name="hasPart" type="xs:string" minOccurs="0"
   maxOccurs="unbounded">
 <xs:annotation>
  <xs:documentation>Links the digital object to child objects
     it may be related to, e.g. in the case of scans of the
     different floors which the overaching building
     representation may link to. The children objects sjall be
     referenced through their identifiers.</xs:documentation>
 </xs:annotation>
</xs:element>
<xs:element name="format" type="xs:string" minOccurs="0">
 <xs:annotation>
  <xs:documentation>The media type format of the digital
     object. Recommendation is to use the mime type to fill
     this value. </xs:documentation>
 </xs:annotation>
</xs:element>
<xs:element name="hasType" type="xs:string" minOccurs="0">
 <xs:annotation>
  <xs:documentation>The type of this digital object, e.g. plan
     or scan.</xs:documentation>
 </xs:annotation>
</xs:element>
<xs:element name="hasFormatDetails" type="xs:string"
   minOccurs="0">
 <xs:annotation>
  <xs:documentation>Additional information out the digital
     object, e.g. it's encoding or compression
     information</xs:documentation>
 </xs:annotation>
</xs:element>
<xs:element name="description" type="xs:string" minOccurs="0"
   maxOccurs="unbounded">
 <xs:annotation>
  <xs:documentation>A description of the digital object, e.g.
     to give information of how and why the object was
     created.</xs:documentation>
 </xs:annotation>
</xs:element>
<xs:element name="provenance" type="xs:string" minOccurs="0"
   maxOccurs="unbounded">
 <xs:annotation>
```

```
          <xs:documentation>A statement of any changes in ownership and
              custody of the digital object since its creation that are
              significant for its authenticity, integrity, and
              interpretation.</xs:documentation>
         </xs:annotation>
        </xs:element>
        <xs:element name="license" type="xs:anyURI" minOccurs="0">
         <xs:annotation>
          <xs:documentation>A link to the license information to the
              digital object.</xs:documentation>
         </xs:annotation>
        </xs:element>
        <xs:element name="unitCode" type="xs:string" minOccurs="0">
         <xs:annotation>
          <xs:documentation>The unit of measurement given using the
              UN/CEFACT Common Code (3 characters). This determines in
              which unit properties corresponding to the Digital Object
              are entered.</xs:documentation>
         </xs:annotation>
        </xs:element>
        <xs:element name="levelOfDetail" type="xs:string" minOccurs="0">
         <xs:annotation>
          <xs:documentation>The level of detail / level of development
              (LOD) in which the physical asset is described / captured
              in the digital object. If a standard reference system is
              used to describe the LOD, the system shall be named with
              the value. </xs:documentation>
         </xs:annotation>
        </xs:element>
        <xs:element name="event" type="xs:string" minOccurs="0"
            maxOccurs="unbounded">
         <xs:annotation>
          <xs:documentation>Information for what the digital object was
              used for, such as in the case of digital objects created
              for presentations or competitions.</xs:documentation>
         </xs:annotation>
        </xs:element>
       </xs:sequence>
      </xs:complexType>
     </xs:element>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
 </xs:schema>
```

Listing 2: XSD representation of the buildm data dictionary

8.4 *E57m* XML Schema Definition - Current DURAARK Draft Status

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    vc:minVersion="1.1"
        xmlns:vc="http://www.w3.org/2007/XMLSchema-versioning">
    <xs:element name="e57m">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="E57root" maxOccurs="1" minOccurs="1">
                    <xs:annotation>
                        <xs:documentation>E57-root level metadata
                            describes basic attributes about the entire
                            file, such as the versionnumber of the file
                            format or the number of 3D scans or 2D
                            images contained in the
                            file.</xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="guid" type="xs:string">
                                <xs:annotation>
                                    <xs:documentation>A globally
                                        unique identification (GUID)
                                        String for the current version
                                        of the file. The standard does
                                        not prescribe the format of the
                                        GUID. Suggestions listed in the
                                        standard include IETF RFC4122
                                        UUIDs or combinations of
                                        make/model/serial number of the
                                        scanner plus creationdatetime
                                        for non-networked
                                        equipment.</xs:documentation>
                                </xs:annotation>
                            </xs:element>
                            <xs:element name="version"
                                type="xs:integer">
                                <xs:annotation>
                                    <xs:documentation>The version of
                                        the file format. The current
                                        version (as of November 2014)
                                        is 1.0.</xs:documentation>
                                </xs:annotation>
                            </xs:element>
                            <xs:element maxOccurs="1" minOccurs="0"
                                name="creation_datetime"
    type="xs:dateTime">
    <xs:annotation>
        <xs:documentation>Date and time
            the file was
            created.</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element minOccurs="0"
    name="coordinate_metadata"
    type="xs:string">
    <xs:annotation>
        <xs:documentation>Describes the
            Coordinate Reference System
            using the well-known text (WKT)
            specification as described in
            the Open Geospatial
            Consortiums's (OGC)
            specification for Coordinate
            Transformation Services.The WKT
            allows the 3D and 2D data
            stored in the file to be
            referenced in a standardized
            coordinate reference
            system.</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="scan_count"
    type="xs:integer">
    <xs:annotation>
        <xs:documentation>Number of single
            scans the file aggregates.
            Within the e57 file, a scan is
            described by a data3D
            object.</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element maxOccurs="1" minOccurs="0"
    name="image_count"
    type="xs:integer">
    <xs:annotation>
        <xs:documentation>Number of 2D
            images the file contains.
            Within the e57 file, each 2D
            image is described by a Image2D
            object.</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element maxOccurs="1" minOccurs="0"
    name="scan_size"
    type="xs:string">
    <xs:annotation>
        <xs:documentation>Describes each
            scan by the following
```

```
                              parameters: scanIndex number,
                              number of Rows, number of
                              Column, number of Points,
                              number of groups, number of
                              points per group, a boolean
                              flag indicating whether the
                              idElementName is
                              ‘‘columnIndex’’.</xs:documentation>
                      </xs:annotation>
                  </xs:element>
                  <xs:element maxOccurs="1" minOccurs="0"
                      name="image_size"
                      type="xs:string">
                      <xs:annotation>
                          <xs:documentation>Describes each
                              image by the following
                              parameters: imageIndex number,
                              image projection type (e.g.,
                              spherical, pinhole), image
                              encoding type, image width,
                              image height, number of bytes
                              in the image, image mask type ,
                              (e.g., for PNG masks), image
                              visual type.</xs:documentation>
                      </xs:annotation>
                  </xs:element>
              </xs:sequence>
          </xs:complexType>
      </xs:element>
      <xs:element maxOccurs="1" minOccurs="1" name="e57scan">
          <xs:annotation>
              <xs:documentation>E57-scan-level metadata
                  describes attributes about the
                  scan.</xs:documentation>
          </xs:annotation>
          <xs:complexType>
              <xs:sequence>
                  <xs:element maxOccurs="1" minOccurs="1"
                      name="guid" type="xs:string">
                      <xs:annotation>
                          <xs:documentation>Global unique
                              identifier for each scan
                              element.</xs:documentation>
                      </xs:annotation>
                  </xs:element>
                  <xs:element maxOccurs="1" minOccurs="0"
                      name="name" type="xs:string">
                      <xs:annotation>
                          <xs:documentation>A name for the
                              3D data as assigned by the
                              user.</xs:documentation>
                      </xs:annotation>
                  </xs:element>
```

```
<xs:element maxOccurs="unbounded"
    minOccurs="0" name="original_guids"
     type="xs:string">
     <xs:annotation>
         <xs:documentation>If the object
             has been modified, i.e., is a
             result of merging muliple scans
             during processing, this element
             contains the guid of one or
             more scansElement which the
             data originated
             from.</xs:documentation>
     </xs:annotation>
</xs:element>
<xs:element maxOccurs="1" minOccurs="0"
    name="description"
     type="xs:string">
     <xs:annotation>
         <xs:documentation>A description of
             the scans as supplied by the
             user.</xs:documentation>
     </xs:annotation>
</xs:element>
<xs:element maxOccurs="1" minOccurs="0"
    name="sensor_vendor"
     type="xs:string">
     <xs:annotation>
         <xs:documentation>Manufacturer of
             scanner/sensor used to capture
             the scan.</xs:documentation>
     </xs:annotation>
</xs:element>
<xs:element maxOccurs="1" minOccurs="0"
    name="sensor_model"
     type="xs:string">
     <xs:annotation>
         <xs:documentation>Model name of
             scanner/sensor used to capture
             the scan.</xs:documentation>
     </xs:annotation>
</xs:element>
<xs:element maxOccurs="1" minOccurs="0"
    name="sensor_serial_number"
     type="xs:string">
     <xs:annotation>
         <xs:documentation>Serial number of
             scanner/sensor used to capture
             the scan.</xs:documentation>
     </xs:annotation>
</xs:element>
<xs:element maxOccurs="1" minOccurs="0"
    name="sensor_hardware_version"
     type="xs:string">
```

```
        <xs:annotation>
            <xs:documentation>Hardware version
                of scanner/sensor used to
                capture the
                scan.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element maxOccurs="1" minOccurs="0"
       name="sensor_software_version"
        type="xs:string">
        <xs:annotation>
            <xs:documentation>Data collection
                version running on
                scanner/sensor used to capture
                the scan.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element maxOccurs="1" minOccurs="0"
       name="sensor_firmware_version"
        type="xs:string">
        <xs:annotation>
            <xs:documentation>Firmware version
                running on
                scanner/sensor.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element maxOccurs="1" minOccurs="0"
       name="temperature"
        type="xs:float">
        <xs:annotation>
            <xs:documentation>Weather data.
                Ambient temperature as measured
                by the sensor of the camera at
                the point of capture, in degree
                Celsius.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element minOccurs="0"
       name="relative_humidity"
       type="xs:float">
        <xs:annotation>
            <xs:documentation>Weather data.
                Relative humidity as measured
                by the sensor of the camera at
                point of capture, in
                percent.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element maxOccurs="1" minOccurs="0"
       name="atmospheric_pressure"
        type="xs:float">
        <xs:annotation>
            <xs:documentation>Weather data.
```

```
                Atmospheric pressure as
                measured by the sensor of the
                camera, at point of capture, in
                Pascal.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element default="xs:dateTime"
        maxOccurs="1" minOccurs="0"
        name="acquisition_start">
        <xs:annotation>
            <xs:documentation>Start date and
                time of scans acquisition, as
                an ISO8601
                date.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element default="xs:dateTime"
        maxOccurs="1" minOccurs="0"
        name="acquisition_end">
        <xs:annotation>
            <xs:documentation>End date and
                time of scans acquisition, as
                an ISO8601
                date.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element maxOccurs="1" minOccurs="0"
        name="pose">
        <xs:annotation>
            <xs:documentation>Pose information
                is necessary when the 3D scan
                data is stored in a local
                coordinate system, which is
                relative to the sensor, as
                opposed to a file-based
                coordinate system. The rotation
                and translation information
                captured in pose allows for the
                transformation of the 3D data
                to a file-based coordinate
                system.</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element maxOccurs="1"
                    minOccurs="0"
                    name="rotation">
                    <xs:annotation>
                        <xs:documentation>These
                            elements describe
                            the scalar part of
                            the quaternion (w),
                            which shall be
                            nonnegative, as
                            well as the i (x),
                            j (y) and k (z)
                            coefficients of the
                            quaternion.</xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                        <xs:sequence>
                          <xs:element
                              maxOccurs="1"
                              minOccurs="1"
                              name="w"
                          type="xs:float"/>
                          <xs:element
                              maxOccurs="1"
                              minOccurs="1"
                              name="x"
                          type="xs:float"/>
                          <xs:element
                              maxOccurs="1"
                              minOccurs="1"
                              name="y"
                          type="xs:float"/>
                          <xs:element
                              maxOccurs="1"
                              minOccurs="1"
                              name="z"
                          type="xs:float"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element maxOccurs="1"
                    minOccurs="0"
                    name="translation">
                    <xs:annotation>
                        <xs:documentation>These
                            elements describe
                            the x, y and z
                            coordinates of the
                            translation, in
                            meters.</xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                        <xs:sequence>
                          <xs:element
                              maxOccurs="1"
                              minOccurs="1"
                              name="x"
                          type="xs:float"/>
                          <xs:element
                              maxOccurs="1"
                              minOccurs="1"
                              name="y"
```

```
                        type="xs:float"/>
                        <xs:element
                            maxOccurs="1"
                            minOccurs="1"
                            name="z"
                        type="xs:float"/>
                      </xs:sequence>
                  </xs:complexType>
              </xs:element>
          </xs:sequence>
      </xs:complexType>
  </xs:element>
  <xs:element maxOccurs="1" minOccurs="0"
      name="index_bounds">
      <xs:annotation>
          <xs:documentation>The points in of
              the scans can be organized in
              rows, columns or by return
              numbers. For the respective
              organization form in use,
              minimum and maximum values need
              to be defined to describe the
              valid indices for the
              bounds.</xs:documentation>
      </xs:annotation>
      <xs:complexType>
          <xs:sequence minOccurs="0">
              <xs:element maxOccurs="1"
                  minOccurs="0"
                  name="row_minimum"
                   type="xs:integer"/>
              <xs:element maxOccurs="1"
                  minOccurs="0"
                  name="row_maximum"
                   type="xs:integer"/>
              <xs:element maxOccurs="1"
                  minOccurs="0"
                  name="col_minimum"
                   type="xs:integer"/>
              <xs:element maxOccurs="1"
                  minOccurs="0"
                  name="col_maximum"
                   type="xs:integer"/>
              <xs:element maxOccurs="1"
                  minOccurs="0"
                   name="return_minimum"
                       type="xs:integer"/>
              <xs:element maxOccurs="1"
                  minOccurs="0"
                   name="return_maximum"
                       type="xs:integer"/>
          </xs:sequence>
      </xs:complexType>
```

```
</xs:element>
<xs:element minOccurs="0"
    name="cartesian_bounds">
    <xs:annotation>
        <xs:documentation>E57 may use two
           different coordinate systems:
           cartesian or spherical. If the
           cartesian coordinate system is
           used, the minimum and maximum
           values for the bounding region
           needs to be defined: the min.
           and max. extend in the x
           direction, min and max extend
           in the y direction and min. and
           max. extend in the z direction,
           in meter.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:sequence minOccurs="1">
            <xs:element maxOccurs="1"
                minOccurs="1"
                name="x_minimum"
                 type="xs:float"/>
            <xs:element maxOccurs="1"
                minOccurs="1"
                name="x_maximum"
                 type="xs:float"/>
            <xs:element maxOccurs="1"
                minOccurs="1"
                name="y_minimum"
                 type="xs:float"/>
            <xs:element maxOccurs="1"
                minOccurs="1"
                name="y_maximum"
                 type="xs:float"/>
            <xs:element maxOccurs="1"
                minOccurs="1"
                name="z_minimum"
                 type="xs:float"/>
            <xs:element maxOccurs="1"
                minOccurs="1"
                name="z_maximum"
                 type="xs:float"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element minOccurs="0"
    name="sphericalbounds">
    <xs:annotation>
        <xs:documentation>E57 may use two
           different coordinate systems:
           cartesian or spherical. If the
           points are represented in
```

```
                spherical coordinates, the
                bounding region needs to be
                defined: min. and max. (range)
                extend in the r direction in
                meters t, min. and max.
                (elevation) extend in the phi
                direction in radians and min.
                and max. (azimuth) extend in
                the theta direction in
                radians.</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence minOccurs="1">
                <xs:element maxOccurs="1"
                    minOccurs="1"
                    name="range_minimum"
                     type="xs:float"/>
                <xs:element maxOccurs="1"
                    minOccurs="1"
                    name="range_maximum"
                     type="xs:float"/>
                <xs:element maxOccurs="1"
                    minOccurs="1"
                     name="elevation_minimum"
                         type="xs:float"/>
                <xs:element maxOccurs="1"
                    minOccurs="1"
                     name="elevation_maximum"
                         type="xs:float"/>
                <xs:element maxOccurs="1"
                    minOccurs="0"
                     name="azimuth_minimum"
                         type="xs:float"/>
                <xs:element maxOccurs="1"
                    minOccurs="0"
                     name="azimuth_maximum"
                         type="xs:float"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element maxOccurs="1" minOccurs="0"
        name="intensity_limits">
        <xs:annotation>
            <xs:documentation>The intensity
                describes the strength of the
                signal for a point. The
                intensity limits need to be
                defined if the points in the
                pointRecord of the  scans
                contain intensity information
                and if the sensor/scanner is
                has defined minimum and maximum
                values for the intensity which
                can be produced by the
                scanner/sensor. The standard
                does not define a unit for this
                element.</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element maxOccurs="1"
                    minOccurs="1"
                     name="intensity_minimum"
                         type="xs:integer"/>
                <xs:element maxOccurs="1"
                    minOccurs="1"
                     name="intensity_maximum"
                         type="xs:integer"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element maxOccurs="1" minOccurs="0"
        name="color_limits">
        <xs:annotation>
            <xs:documentation>PointRecords can
                be optionally described by
                color elements. These elements
                describe the limit of values
                for RGB, which the scanner is
                capable of producing. The
                standard does not prescribe a
                unit.</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence minOccurs="1">
                <xs:element maxOccurs="1"
                    minOccurs="1"
                     name="color_red_minimum"
                         type="xs:integer"/>
                <xs:element maxOccurs="1"
                    minOccurs="1"
                     name="color_red_maximum"
                         type="xs:integer"/>
                <xs:element maxOccurs="1"
                    minOccurs="1"
                     name="color_green_minimum"
                         type="xs:integer"/>
                <xs:element maxOccurs="1"
                    minOccurs="1"
                     name="color_green_maximum"
                         type="xs:float"/>
                <xs:element maxOccurs="1"
                    minOccurs="1"
                     name="color_blue_minimum"
                         type="xs:integer"/>
                <xs:element maxOccurs="1"
                    minOccurs="1"
                    name="color_blue_maximum"
                        type="xs:integer"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element maxOccurs="1" minOccurs="1"
        name="pointSize"
        type="xs:integer">
        <xs:annotation>
            <xs:documentation>Returns the
                number of records found in the
                respective
                scan.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element maxOccurs="1" minOccurs="1"
        name="point_fields">
        <xs:annotation>
            <xs:documentation>The point fields
                check the structure of the
                point records for the
                availability of standardized
                fields as well as the
                meaningfulness for some of the
                content. </xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element maxOccurs="1"
                    minOccurs="0"
                    name="cartesian_fields">
                    <xs:annotation>
                        <xs:documentation>Checks
                            for the existence
                            of a pointer to a
                            buffer with the x,
                            y and z
                            coordinates,
                            returning true for
                            the respective
                            element. The
                            cartesian_invalid_state_field
                            indicates whether
                            the content of the
                            x, y and z fields
                            are not
                            meaningful.</xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                        <xs:sequence
                            maxOccurs="1"
                            minOccurs="1">
                          <xs:element
                              maxOccurs="1"
                              minOccurs="1"
                          name="cartesian_x_field"
                              type="xs:boolean"/>
                          <xs:element
                              maxOccurs="1"
                              minOccurs="1"
                          name="cartesian_y_field"
                              type="xs:boolean"/>
                          <xs:element
                              maxOccurs="1"
                              minOccurs="1"
                          name="cartesian_z_field"
                              type="xs:boolean"/>
                          <xs:element
                              maxOccurs="1"
                              minOccurs="1"
                          name="cartesian_invalid_state_field"
                          type="xs:boolean"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element maxOccurs="1"
                    minOccurs="0"
                    name="spherical_fields">
                    <xs:annotation>
                        <xs:documentation>Checks
                            for the existence
                            of a pointer to a
                            buffer with the x,
                            y and z
                            coordinates,
                            returning true for
                            the respective
                            element. The
                            spherical_invalid_state_field
                            indicates whether
                            the content of the
                            x, y and z fields
                            are not
                            meaningful.</xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                        <xs:sequence
                            maxOccurs="1"
                            minOccurs="1">
                          <xs:element
                              maxOccurs="1"
                              minOccurs="1"
```

```
                                            name="spherical_range_field"
                                               type="xs:boolean"/>
                                            <xs:element
                                               maxOccurs="1"
                                               minOccurs="1"
                                            name="spherical_elevation_field"
                                               type="xs:boolean"/>
                                            <xs:element
                                               maxOccurs="1"
                                               minOccurs="1"
                                            name="spherical_azimuth_field"
                                               type="xs:boolean"/>
                                            <xs:element
                                               maxOccurs="1"
                                               minOccurs="1"
                                            name="spherical_invalid_state_field"
                                            type="xs:boolean"/>
                                          </xs:sequence>
                                      </xs:complexType>
                                  </xs:element>
                                  <xs:element maxOccurs="1"
                                     minOccurs="0"
                                     name="point_range">
                                      <xs:annotation>
                                          <xs:documentation>Returns
                                             the values of
                                             point_range min.
                                             and max. values.
                                             The
                                             point_range_scaled_integer
                                             field either
                                             indicates that the
                                             point records
                                             should be
                                             configured as a
                                             scaled integer
                                             node, returning 1,
                                             or returns the
                                             float
                                             node.</xs:documentation>
                                      </xs:annotation>
                                      <xs:complexType>
                                          <xs:sequence>
                                            <xs:element
                                               maxOccurs="1"
                                               minOccurs="1"
                                            name="point_range_minimum"
                                               type="xs:float"/>
                                            <xs:element
                                               maxOccurs="1"
                                               minOccurs="1"
                                            name="point_range_maximum"
                                               type="xs:float"/>
```

```
                    <xs:element
                        maxOccurs="1"
                        minOccurs="1"
                    name="point_range_scaled_integer"
                        type="xs:float"
                    />
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element maxOccurs="1"
            minOccurs="0" name="angles">
            <xs:annotation>
                <xs:documentation>For
                    scans in a
                    spherical
                    coordinate system,
                    these values return
                    the min. and max.
                    angle which may be
                    used. The
                    angle_scaled_integer
                    either returns a 1,
                    indicating that the
                    point record should
                    be configured as a
                    scaled integer
                    node, or returns
                    the float
                    node.</xs:documentation>
            </xs:annotation>
            <xs:complexType>
                <xs:sequence>
                    <xs:element
                        maxOccurs="1"
                        minOccurs="1"
                    name="angle_minimum"
                        type="xs:float"/>
                    <xs:element
                        maxOccurs="1"
                        minOccurs="1"
                    name="angle_maximum"
                        type="xs:float"/>
                    <xs:element
                        maxOccurs="1"
                        minOccurs="1"
                    name="angle_scaled_integer"
                        type="xs:float"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element maxOccurs="1"
            minOccurs="0"
            name="index_fields">
```

```
<xs:annotation>
    <xs:documentation>The
        index fields
        indicate if the
        respective index
        fields for the type
        of index (row,
        column, return) are
        active. For the
        return index, the
        count field is
        checked against
        being active as
        well. The maximum
        fields return the
        maximum values for
        the respective
        index, the minimum
        is assumed to be
        set to
        0.</xs:documentation>
</xs:annotation>
<xs:complexType>
    <xs:sequence>
      <xs:element
         maxOccurs="1"
         minOccurs="1"
      name="row_index_field"
         type="xs:boolean"/>
      <xs:element
         maxOccurs="1"
         minOccurs="1"
      name="row_index_maximum"
         type="xs:integer"/>
      <xs:element
         maxOccurs="1"
         minOccurs="1"
      name="column_index_field"
         type="xs:boolean"/>
      <xs:element
         maxOccurs="1"
         minOccurs="1"
      name="column_index_maximum"
         type="xs:integer"/>
      <xs:element
         maxOccurs="1"
         minOccurs="1"
      name="return_index_field"
         type="xs:boolean"/>
      <xs:element
         maxOccurs="1"
         minOccurs="1"
```

```
                                                name="return_count_field"
                                                    type="xs:boolean"/>
                                                <xs:element
                                                    maxOccurs="1"
                                                    minOccurs="1"
                                                name="return_maximum"
                                                    type="xs:integer"/>
                                            </xs:sequence>
                                        </xs:complexType>
                                    </xs:element>
                                    <xs:element maxOccurs="1"
                                        minOccurs="0"
                                        name="time_fields">
                                        <xs:annotation>
                                            <xs:documentation>Checks
                                                for the presence
                                                and validity of a
                                                time_stamp in the
                                                file, returning
                                                true when
                                                time_stamp
                                                exists.Time_Maximum
                                                indicates a maximum
                                                values which point
                                                record time_stamp
                                                fields should be
                                                configured
                                                with.</xs:documentation>
                                        </xs:annotation>
                                        <xs:complexType>
                                            <xs:sequence>
                                                <xs:element
                                                    maxOccurs="1"
                                                    minOccurs="1"
                                                name="time_stamp_field"
                                                    type="xs:boolean"/>
                                                <xs:element
                                                    maxOccurs="1"
                                                    minOccurs="1"
                                                name="is_Time_Stamp_invalid"
                                                    type="xs:boolean"/>
                                                <xs:element
                                                    maxOccurs="1"
                                                    minOccurs="1"
                                                name="time_Maximum"
                                                    type="xs:float"/>
                                            </xs:sequence>
                                        </xs:complexType>
                                    </xs:element>
                                    <xs:element maxOccurs="1"
                                        minOccurs="0"
                                        name="intensity_color_fields">
                                        <xs:annotation>
```

```
                    <xs:documentation>Checks
                       for the presence
                       and use of
                       intensity and color
                       schemes in the
                       respective fields,
                       returning true when
                       found. The
                       intensity_scaled_integer
                       field indicates
                       either that the
                       point record
                       intensity fields
                       should be
                       configured as a
                       scaled integer node
                       (returning 1) or
                       returns the float
                       node or the integer
                       node. The
                       is_color_invalid_field
                       indicates whether
                       the content of the
                       color fields for
                       red, green and blue
                       are not
                       meaningful.</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                    <xs:sequence>
                      <xs:element
                         maxOccurs="1"
                         minOccurs="1"
                      name="intensity_field"
                         type="xs:boolean"/>
                      <xs:element
                         maxOccurs="1"
                         minOccurs="1"
                      name="is_intensity_invalid_field"
                      type="xs:boolean"/>
                      <xs:element
                         maxOccurs="1"
                         minOccurs="1"
                      name="intensity_scaled_integer"
                         type="xs:float"/>
                      <xs:element
                         maxOccurs="1"
                         minOccurs="1"
                      name="color_red_field"
                         type="xs:boolean"/>
                      <xs:element
                         maxOccurs="1"
                         minOccurs="1"
```

```
                                    name="color_green_field"
                                        type="xs:boolean"/>
                                    <xs:element
                                        maxOccurs="1"
                                        minOccurs="1"
                                    name="color_blue_field"
                                        type="xs:boolean"/>
                                    <xs:element
                                        maxOccurs="1"
                                        minOccurs="1"
                                    name="is_color_invalid_field"
                                        type="xs:boolean"/>
                                  </xs:sequence>
                              </xs:complexType>
                          </xs:element>
                      </xs:sequence>
                  </xs:complexType>
              </xs:element>
          </xs:sequence>
      </xs:complexType>
  </xs:element>
  <xs:element maxOccurs="1" minOccurs="0"
      name="e57image">
      <xs:annotation>
          <xs:documentation>Information at the images
              level describes each 2D image contained
              within the E57 file.</xs:documentation>
      </xs:annotation>
      <xs:complexType>
          <xs:sequence>
              <xs:element maxOccurs="1" minOccurs="0"
                  name="name" type="xs:string">
                  <xs:annotation>
                      <xs:documentation>A name for the
                          image as assigned by the
                          user.</xs:documentation>
                  </xs:annotation>
              </xs:element>
              <xs:element maxOccurs="1" minOccurs="1"
                  name="guid" type="xs:string">
                  <xs:annotation>
                      <xs:documentation>Global unique
                          identifier for each image
                          element.</xs:documentation>
                  </xs:annotation>
              </xs:element>
              <xs:element maxOccurs="1" minOccurs="0"
                  name="representation"
                  type="xs:string">
                  <xs:annotation>
                      <xs:documentation>Describes the
                          representation type of the
                          image. The representation type
```

```
                            can either be defined by the
                            spherical, cylindrical or
                            pinhole camera projection model
                            - or follow no camera
                            projection model, in which case
                            the image shall serve as a
                            visual reference
                            only.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element maxOccurs="1" minOccurs="0"
                    name="description"
                    type="xs:string">
                    <xs:annotation>
                        <xs:documentation>A description of
                            the image as supplied by the
                            user.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element maxOccurs="1" minOccurs="0"
                    name="acquisition_datetime"
                    type="xs:dateTime">
                    <xs:annotation>
                        <xs:documentation>Timestamp of the
                            acquisition time of the image,
                            as an ISO8601
                            date.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element maxOccurs="1" minOccurs="0"
                    name="associated_data3D_guid"
                    type="xs:string">
                    <xs:annotation>
                        <xs:documentation>Lists the guid
                            of the 3D scan image, which
                            this image corresponds
                            with.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element maxOccurs="1" minOccurs="0"
                    name="sensor_vendor"
                    type="xs:string">
                    <xs:annotation>
                        <xs:documentation>Manufactures of
                            the sensor which was used to
                            capture this
                            image.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element maxOccurs="1" minOccurs="0"
                    name="sensor_model"
                    type="xs:string">
                    <xs:annotation>
                        <xs:documentation>The model name
                            of the sensor which was used to
                            capture this
                            image.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element maxOccurs="1" minOccurs="0"
                    name="sensor_serial_number"
                    type="xs:string">
                    <xs:annotation>
                        <xs:documentation>The serial
                            number of the sensor which was
                            used to capture this
                            image.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element maxOccurs="1" minOccurs="0"
                    name="pose">
                    <xs:annotation>
                        <xs:documentation>Pose information
                            is necessary when the image
                            data is stored in a local
                            coordinate system, which is
                            relative to the sensor, as
                            opposed to a file-based
                            coordinate system. The rotation
                            and translation information
                            captured in pose allows for the
                            transformation of the Image
                            data to a file-based coordinate
                            system.</xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                        <xs:sequence minOccurs="0">
                            <xs:element maxOccurs="1"
                                minOccurs="0"
                                name="rotation">
                                <xs:annotation>
                                    <xs:documentation>These
                                        elements describe
                                        the scalar part of
                                        the quaternion (w),
                                        which shall be
                                        nonnegative, as
                                        well as the i (x),
                                        j (y) and k (z)
                                        coefficients of the
                                        quaternion.</xs:documentation>
                                </xs:annotation>
                                <xs:complexType>
                                    <xs:sequence>
                                      <xs:element
                                          maxOccurs="1"
```

```
                                    minOccurs="1"
                                    name="w"
                                type="xs:float"/>
                                <xs:element
                                    maxOccurs="1"
                                    minOccurs="1"
                                    name="x"
                                type="xs:float"/>
                                <xs:element
                                    maxOccurs="1"
                                    minOccurs="1"
                                    name="y"
                                type="xs:float"/>
                                <xs:element
                                    maxOccurs="1"
                                    minOccurs="1"
                                    name="z"
                                type="xs:float"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element maxOccurs="1"
                        minOccurs="0"
                        name="translation">
                        <xs:annotation>
                            <xs:documentation>These
                                elements describe
                                the x, y and z
                                coordinates of the
                                translation, in
                                meters.</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element
                                    maxOccurs="1"
                                    minOccurs="1"
                                    name="x"
                                type="xs:float"/>
                                <xs:element
                                    maxOccurs="1"
                                    minOccurs="1"
                                    name="y"
                                type="xs:float"/>
                                <xs:element
                                    maxOccurs="1"
                                    minOccurs="1"
                                    name="z"
                                type="xs:float"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
```

```
        </xs:complexType>
    </xs:element>
    <xs:element maxOccurs="unbounded"
        minOccurs="0"
        name="visual_ref_representation">
        <xs:annotation>
            <xs:documentation>These elements
                are used, if an image is to be
                used as a visual reference. The
                image is then described with
                the size of the image data
                (either jpeg or png) found in
                the blob node,  if used - the
                size of the image mask in the
                blob node as well as the image
                width and height in
                pixels.</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element maxOccurs="1"
                    minOccurs="0"
                    name="jpeg_image_size"
                        type="xs:integer"/>
                <xs:element maxOccurs="1"
                    minOccurs="0"
                    name="png_image_size"
                        type="xs:integer"/>
                <xs:element maxOccurs="1"
                    minOccurs="0"
                    name="image_mask_size"
                        type="xs:integer"/>
                <xs:element maxOccurs="1"
                    minOccurs="1"
                    name="image_width"
                    type="xs:integer"/>
                <xs:element maxOccurs="1"
                    minOccurs="1"
                    name="image_height"
                    type="xs:integer"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element maxOccurs="unbounded"
        minOccurs="0"
        name="spherical_representation">
        <xs:annotation>
            <xs:documentation>These elements
                are used for images which are
                mapped to 3D using the
                spherical camera projection
                model. The image is then
                described through the size of
```

```
                            the image data (either jpeg or
                            png) found in the blob node, if
                            used - the size of the image
                            mask in the blob node, the
                            image width and heights in
                            pixel as well as by the pixel's
                            width and height in
                            radians.</xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element maxOccurs="1"
                                minOccurs="0"
                                 name="jpeg_image_size"
                                     type="xs:integer"/>
                            <xs:element maxOccurs="1"
                                minOccurs="0"
                                 name="png_image_size"
                                     type="xs:integer"/>
                            <xs:element maxOccurs="1"
                                minOccurs="0"
                                 name="image_mask_size"
                                     type="xs:integer"/>
                            <xs:element maxOccurs="1"
                                minOccurs="1"
                                name="image_width"
                                 type="xs:integer"/>
                            <xs:element maxOccurs="1"
                                minOccurs="1"
                                name="image_height"
                                 type="xs:integer"/>
                            <xs:element maxOccurs="1"
                                minOccurs="1"
                                name="pixel_width"
                                 type="xs:float"/>
                            <xs:element maxOccurs="1"
                                minOccurs="1"
                                name="pixel_height"
                                 type="xs:float"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element maxOccurs="unbounded"
                    minOccurs="0"
                     name="pinhole_representation">
                     <xs:annotation>
                         <xs:documentation>These elements
                             are used for images which are
                             mapped to 3D using the pinhole
                             camera projection model. The
                             image is then described with
                             the size of the image data
                             (either jpeg or png) found in
                             the blob node, if used - the
                             size of the image mask in the
                             blob node, the image width and
                             height in pixel, the camera's
                             focal length in meters, the
                             pixels' width and height, both
                             in meters, as well as the x and
                             y coordinates of the principal
                             point found in the image, in
                             pixels.</xs:documentation>
                     </xs:annotation>
                     <xs:complexType>
                         <xs:sequence>
                            <xs:element maxOccurs="1"
                                minOccurs="0"
                                 name="jpeg_image_size"
                                     type="xs:integer"/>
                            <xs:element maxOccurs="1"
                                minOccurs="0"
                                 name="png_image_size"
                                     type="xs:integer"/>
                            <xs:element maxOccurs="1"
                                minOccurs="0"
                                 name="image_mask_size"
                                     type="xs:integer"/>
                            <xs:element maxOccurs="1"
                                minOccurs="1"
                                name="image_width"
                                 type="xs:integer"/>
                            <xs:element maxOccurs="1"
                                minOccurs="1"
                                name="image_height"
                                 type="xs:integer"/>
                            <xs:element maxOccurs="1"
                                minOccurs="1"
                                name="focal_length"
                                 type="xs:float"/>
                            <xs:element maxOccurs="1"
                                minOccurs="1"
                                name="pixel_width"
                                 type="xs:float"/>
                            <xs:element maxOccurs="1"
                                minOccurs="1"
                                name="pixel_height"
                                 type="xs:float"/>
                            <xs:element maxOccurs="1"
                                minOccurs="1"
                                 name="principal_point_x"
                                     type="xs:float"/>
                            <xs:element maxOccurs="1"
                                minOccurs="1"
                                 name="principal_point_y"
                                     type="xs:float"/>
                         </xs:sequence>
                     </xs:complexType>
                 </xs:element>
                 <xs:element maxOccurs="unbounded"
                    minOccurs="0"
                     name="cylindrical_representation">
                     <xs:annotation>
                         <xs:documentation>These elements
                             are used for images which are
                             mapped to 3D using the
                             cylindrical projection model.
                             The image is then described
                             with the size of the image data
                             (either jpeg or png) found in
                             the blob node, if used - the
                             size of the image mask in the
                             blob node, the image width and
                             height in pixel, the camera's
                             focal length in meters, the
                             pixel's width in radians and
                             height in meters as well as the
                             radius of the cylinder in
                             meters and the y coordinates of
                             the principal point found in
                             the image, in
                             pixels.</xs:documentation>
                     </xs:annotation>
                     <xs:complexType>
                         <xs:sequence>
                            <xs:element maxOccurs="1"
                                minOccurs="0"
                                 name="jpeg_image_size"
                                     type="xs:integer"/>
                            <xs:element maxOccurs="1"
                                minOccurs="0"
                                 name="png_image_size"
                                     type="xs:integer"/>
                            <xs:element maxOccurs="1"
                                minOccurs="0"
                                 name="image_mask_size"
                                     type="xs:integer"/>
                            <xs:element maxOccurs="1"
                                minOccurs="1"
                                name="image_width"
                                 type="xs:integer"/>
                            <xs:element maxOccurs="1"
                                minOccurs="1"
                                name="image_height"
                                 type="xs:integer"/>
                            <xs:element maxOccurs="1"
                                minOccurs="1"
                                name="pixel_width"
                                 type="xs:float"/>
```

```
                                                        <xs:element maxOccurs="1"
                                                            minOccurs="1"
                                                            name="pixel_height"
                                                             type="xs:float"/>
                                                        <xs:element maxOccurs="1"
                                                            minOccurs="1" name="radius"
                                                             type="xs:float"/>
                                                        <xs:element maxOccurs="1"
                                                            minOccurs="1"
                                                             name="principal_point_y"
                                                                 type="xs:float"/>
                                                    </xs:sequence>
                                                </xs:complexType>
                                            </xs:element>
                                        </xs:sequence>
                                    </xs:complexType>
                                </xs:element>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:schema>
```

Listing 3: XSD representation of the e57m data dictionary

8.5 *Ifcm* XML Schema Definition - Current DURAARK Draft Status

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified"
    vc:minVersion="1.1"
        xmlns:vc="http://www.w3.org/2007/XMLSchema-versioning">
    <xs:element name="ifcm">
        <xs:complexType>
            <xs:sequence>
                <xs:element maxOccurs="1" minOccurs="1" name="header">
                    <xs:annotation>
                        <xs:documentation>The following elements are
                            extracted from the STEP physical file
                            format header portion of the IFC file. They
                            contain information about the creation
                            process of the file as well as about the
                            schema and view make-up of the digital
                            object.</xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                        <xs:sequence maxOccurs="1" minOccurs="1">
                            <xs:element maxOccurs="1" minOccurs="1"
                                name="name" type="xs:string">
                                <xs:annotation>
                                    <xs:documentation>Name of digital
                                        object as used for data
                                        exchange, as taken from the
                                        name attribute in IFC header's
                                        FILE_NAME
                                        entity.</xs:documentation>
                                </xs:annotation>
                            </xs:element>
                            <xs:element maxOccurs="1" minOccurs="1"
                                name="creationDate"
                                type="xs:dateTime">
                                <xs:annotation>
                                    <xs:documentation>Creation date of
                                        IFC file in ISO 8601 format, as
                                        taken from the time_stamp
                                        attribute in IFC header's
                                        FILE_NAME
                                        entity.</xs:documentation>
                                </xs:annotation>
                            </xs:element>
                            <xs:element maxOccurs="1" minOccurs="0"
                                name="author" type="xs:string">
                                <xs:annotation>
                                    <xs:documentation>Creator of IFC
                                        file, as taken from the author
                                        attribute in IFC header's
                                        FILE_NAME
                                        entity.</xs:documentation>
                                </xs:annotation>
                            </xs:element>
                            <xs:element maxOccurs="1" minOccurs="0"
                                name="organization"
                                type="xs:string">
                                <xs:annotation>
                                    <xs:documentation>Organization,
                                        which the creator of the IFC
                                        file is associated with, as
                                        taken from the organization
                                        attribute in IFC header's
                                        FILE_NAME
                                        entity.</xs:documentation>
                                </xs:annotation>
                            </xs:element>
                            <xs:element maxOccurs="1" minOccurs="0"
                                name="preprocessor"
                                type="xs:string">
                                <xs:annotation>
                                    <xs:documentation>Name and version
                                        of the toolbox used to create
                                        the IFC file, as taken from the
                                        preprocessor_version attribute
                                        in IFC header's FILE_NAME
                                        entity.</xs:documentation>
                                </xs:annotation>
                            </xs:element>
                            <xs:element maxOccurs="1" minOccurs="0"
                                name="originatingSystem"
                                type="xs:string">
                                <xs:annotation>
                                    <xs:documentation>Name and version
                                        of the CAD system or other
                                        application used to generate
                                        the IFC file, including the
                                        build information, where
                                        possible. As taken from the
                                        originating_system attribute in
                                        IFC header's FILE_NAME
                                        entity.</xs:documentation>
                                </xs:annotation>
                            </xs:element>
                            <xs:element maxOccurs="1" minOccurs="0"
                                name="authorization"
                                type="xs:string">
                                <xs:annotation>
                                    <xs:documentation>Name and address
                                        of the person / organization
```

```
                        who authorized the transfer of
                        the IFC file, as taken from the
                        authorization attribute in IFC
                        header's FILE_NAME
                        entity.</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element maxOccurs="1" minOccurs="1"
               name="fileSchema"
                type="xs:string">
                <xs:annotation>
                    <xs:documentation>Schema version
                        of the IFC file, as taken from
                        the FILE_SCHEMA entity in the
                        IFC header
                        file.</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element maxOccurs="1" minOccurs="1"
               name="viewDefinition"
                type="xs:string">
                <xs:annotation>
                    <xs:documentation>Describes one or
                        more model view definitions of
                        the IFC File, as taken from the
                        view_defintion attribute of the
                        FILE_DESCRIPTION entity in the
                        IFC header file. Values for
                        view definitions have to be
                        formally agreed upon by
                        buildingSMART.</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element maxOccurs="1" minOccurs="0"
               name="exportOptions"
                type="xs:string">
                <xs:annotation>
                    <xs:documentation>Describes export
                        options or additional filters
                        used for the IFC generation, as
                        taken from the FILE_DESCRIPTION
                        entity in the IFC header
                        file.</xs:documentation>
                </xs:annotation>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element maxOccurs="1" minOccurs="1"
   name="ifcparameters">
    <xs:annotation>
        <xs:documentation>The ifcParameters are values
            from corresponding IFC entities. For
```

```
                transparency reasons, the name of the ifcm
                elements are congruent with the name of the
                corresponding IFC entity.</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence maxOccurs="1" minOccurs="1">
                <xs:element maxOccurs="unbounded"
                    minOccurs="0" name="ifcApplication"
                    type="xs:string">
                    <xs:annotation>
                        <xs:documentation>Describes
                            software involved in the
                            IFC/BIM process of the object,
                            as taken from the
                            ifcApplication enitity. The
                            notation includes application
                            developer, version, application
                            full name and
                            identifier.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element maxOccurs="unbounded"
                    minOccurs="0"
                    name="IfcGeometricRepresentationContext"
                        type="xs:string">
                    <xs:annotation>
                        <xs:documentation>Describes the
                            context in which applies to a
                            group of geometric shapes in
                            the file including the
                            representation context group
                            (model or plan with respective
                            subgroups), the dimension count
                            of the coordinate space the
                            representation is modelled in,
                            the precision or tolerance with
                            which two points are assumed to
                            be identical to each other and
                            the TrueNorth or offset from
                            the project's coordinate system
                            (optional).</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element maxOccurs="unbounded"
                    minOccurs="0" name="ifcSiUnit"
                    type="xs:string">
                    <xs:annotation>
                        <xs:documentation>Describes the
                            units which apply to different
                            measure in the IFC file,
                            stating the dimension to be
                            measured and the unit applied.
                            This does not only apply to
                            length, area and mass units but
                            to all subjects included in the
                            BIM process such as
                            electricity, light, thermo
                            dynamics or
                            time.</xs:documentation>
                    </xs:annotation>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element maxOccurs="1" minOccurs="1"
        name="countObjects">
        <xs:annotation>
            <xs:documentation>This group contains counts
                for various basic entities within the ifc
                object.</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence maxOccurs="1" minOccurs="1">
                <xs:element maxOccurs="1" minOccurs="0"
                    name="floorCount"
                    type="xs:integer">
                    <xs:annotation>
                        <xs:documentation>Describes the
                            number of floors / storeys
                            which are modelled in the IFC
                            file.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element maxOccurs="1" minOccurs="0"
                    name="roomCount"
                    type="xs:integer">
                    <xs:annotation>
                        <xs:documentation>Describes the
                            number of rooms which are
                            modelled in the IFC
                            file.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element maxOccurs="1" minOccurs="0"
                    name="wallCount"
                    type="xs:integer">
                    <xs:annotation>
                        <xs:documentation>Describes the
                            number of walls which are
                            modelled in the IFC
                            file.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element maxOccurs="1" minOccurs="0"
                    name="windowsCount"
                    type="xs:integer">
                    <xs:annotation>
                        <xs:documentation>Describes the
                            number of windows which are
                            modelled in the IFC
                            file.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element maxOccurs="1" minOccurs="0"
                    name="doorCount"
                    type="xs:integer">
                    <xs:annotation>
                        <xs:documentation>Describes the
                            number of doors which are
                            modelled in the IFC
                            file.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element maxOccurs="1" minOccurs="0"
                    name="pipeCount"
                    type="xs:integer">
                    <xs:annotation>
                        <xs:documentation>Describes the
                            number of pipes which are
                            modelled in the IFC
                            file.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element maxOccurs="1" minOccurs="0"
                    name="columnCount"
                    type="xs:integer">
                    <xs:annotation>
                        <xs:documentation>Describes the
                            number of columns which are
                            modelled in the IFC
                            file.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element maxOccurs="1" minOccurs="0"
                    name="numberOfComponents"
                    type="xs:integer">
                    <xs:annotation>
                        <xs:documentation>Describes the
                            total number IfcProduct
                            subtypes which have been
                            instantiated in the object
                            (counts al doors, windows,
                            roofs, walls,
                            etc.).</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element maxOccurs="1" minOccurs="0"
                    name="numberOfRelations"
                    type="xs:integer">
```

```
                <xs:annotation>
                    <xs:documentation>Describes the
                        total number relations -
                        internal as well as external -
                        which are used in the IFC
                        file.</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element maxOccurs="1" minOccurs="0"
                name="numberOfActors"
                type="xs:integer">
                <xs:annotation>
                    <xs:documentation>Describes the
                        number of actors which are
                        listed in the IFC
                        file.</xs:documentation>
                </xs:annotation>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element maxOccurs="1" minOccurs="1"
    name="informationMetric">
    <xs:annotation>
        <xs:documentation>This group contains elements
            which are based on counted and/or
            calculated values of
            entities.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element maxOccurs="1" minOccurs="0"
                name="numberOfEntityTypesUsed"
                type="xs:integer">
                <xs:annotation>
                    <xs:documentation>Describes the
                        number of different entities
                        which are used in the IFC file.
                        Each different entity is only
                        counted once, regardless of how
                        many times it is instantiated
                        in the file.</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element maxOccurs="1" minOccurs="0"
                name="numberOfTotalEntitiesUsed"
                type="xs:integer">
                <xs:annotation>
                    <xs:documentation>Gives the total
                        count of entities which are
                        used in the IFC
                        file.</xs:documentation>
                </xs:annotation>
```

```
                        </xs:element>
                        <xs:element maxOccurs="1" minOccurs="0"
                            name="optionalAttributes"
                            type="xs:integer">
                            <xs:annotation>
                                <xs:documentation>Describes the
                                    percentage of OPTIONAL
                                    schema-level attributed which
                                    are provided with values in
                                    this file.</xs:documentation>
                            </xs:annotation>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element maxOccurs="1" minOccurs="1"
                name="Dependencies">
                <xs:annotation>
                    <xs:documentation>This group contains
                        information about internal or external
                        dependencies of the IFC object. The element
                        contained within is an aggregate of various
                        entities.</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                    <xs:sequence maxOccurs="1" minOccurs="1">
                        <xs:element maxOccurs="unbounded"
                            minOccurs="0" name="webResourceLink"
                            type="xs:string">
                            <xs:annotation>
                                <xs:documentation>Describes every
                                    web resource (URL) which the
                                    IFC links to, including from
                                    which entity type in the IFC
                                    file it was referenced from and
                                    how many times it was linked
                                    from that
                                    type.</xs:documentation>
                            </xs:annotation>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

Listing 4: XSD representation of the ifcm data dictionary