



DURAARK
DURABLE
ARCHITECTURAL
KNOWLEDGE

D2.5 Software prototype v2

DURAARK

FP7 – ICT – Digital Preservation

Grant agreement No.: 600908

Date: 2015-07-31

Version 2.0

Document id. : duraark/2015/D.2.5/v2.0



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 – Kunstakademiet 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.2.5
Title of document	:	Software prototype v2
Deliverable type	:	Software prototype
Contractual date of delivery	:	2015-07-31
Actual date of delivery	:	2015-07-31
Lead beneficiary	:	Fraunhofer Austria (FhA)
Author(s)	:	Martin Hecher < martin.hecher@fraunhofer.at > (FhA) Dag Field Edvardsen < dag.fjeld.edvardsen@catenda.no > (Catenda) Sebastian Ochmann < ochmann@cs.uni-bonn.de > (UBO) Michelle Lindlar < michelle.lindlar@tib.uni-hannover.de > (LUH) Michael Panitz < michael.panitz@tib.uni-hannover.de > (LUH) Hamid Rofoogaran < hamid.rofoogaran@ltu.se > (LTU) Ujwal Gadiraju < gadiraju@l3s.de > (L3S) Besnik Fetahu < fetahu@l3s.de > (L3S)
Responsible editor(s)	:	Martin Hecher < martin.hecher@fraunhofer.at > (FhA)
Quality assessor(s)	:	Michelle Lindlar < michelle.lindlar@tib.uni-hannover.de > (LUH) Raoul Wessel < wesselr@cs.uni-bonn.de > (UBO)
Approval of this deliverable	:	Jakob Beetz < j.beetz@tue.nl > (TUE) Stefan Dietze < dietze@l3s.de > (LUH)
Distribution	:	Public
Keywords list	:	prototype, workbench, use cases

Executive Summary

In this report we describe the final version of the integrated software prototype in DURAARK. Focus of work since the last prototype in M18 was the enhancement of workflows within the graphical user interfaces developed in the project to carry out use cases in the architecture, engineering and construction (AEC) domain. The developed workflows combine the individual components developed by project partners and allows the integrated software prototype to be used as a “**semantic building information modeling (BIM) archival and retrieval system**“. Since M18 the software design was improved and now provides the **Service Platform** which encapsulates the core functionality developed in DURAARK. This platform allows the reuse of the developed functionality in 3rd party software, as well as in existing stakeholder workflows via a platform independent application programming interface (API). We provide two graphical user interfaces which utilize the Service Platform. The **WorkbenchUI** is a web-application covering all defined use cases, as well as an extension to the **Grasshopper®** software. Grasshopper® is a domain-specific software in the AEC world and is utilizing the Service Platform to allow stakeholders to work with pointcloud and BIM files in their common work environment. This extension is also demonstrating the integration capabilities of the Service Platform into 3rd party software.

Table of Contents

1	Introduction	6
1.1	Moving further from D2.4	6
1.2	Overview of this Deliverable	10
1.3	Implementation Status	10
1.4	Accessing the Software Prototypes	12
1.5	Deviations from D2.4	17
2	WorkbenchUI Workflows	18
2.1	Pre-ingest Workflow	18
2.1.1	Addressed Use Cases	18
2.1.2	User Manual	20
2.2	Retrieval Workflow	29
2.2.1	Addressed Use Cases	32

2.2.2	User Manual	34
2.3	Maintenance Workflow	35
3	Grasshopper® Workflows	40
3.1	Addressed Use Cases	40
3.2	Processing Building Data in Stakeholder Workflows	41
3.3	Workflow Example	43
4	Service Platform	47
4.1	Session Service	48
4.2	Metadata Service	49
4.3	Semantic Digital Archive Service	49
4.3.1	SDAS	50
4.3.2	Focused Crawling	54
4.4	Geometric Enrichment Service	55
4.4.1	IFC Reconstruction	55
4.4.2	Reveal Invisible StructurEs (RISE)	56
4.5	Digital Preservation Service	57
4.5.1	SIP Generator	58

4.5.2	ROSETTA DPS	62
5	Decisions & Risks	66
5.1	Technical decisions and impacts	66
5.2	Risk assessment	67
6	Software Licenses	69
7	Conclusions & Impact	70
8	Future Work	72
Appendices		76
1	Software Design Finalization	76
2	Development Quality Assurance	78
2.1	Source Code and Documentation	78
2.2	Continuous Integration	79
2.3	Deployment	79
2.4	Release Planning	80

1 Introduction

This document describes the second version of the integrated software prototype with an extended feature-set compared to the previous version (D2.4) and the final iteration of the prototype's software architecture. We present the *Service Platform*, which integrates the main software prototype components of partner work packages into a service-oriented architecture. The Service Platform provides an application programming interface (API) to make the component functionality accessible by the DURAARK graphical user interfaces and also for 3rd party software. Two graphical user interfaces are developed within the project: the *WorkbenchUI* and the DURAARK *Grasshopper®* components. The WorkbenchUI is a web-application allowing stakeholders to carry out all use cases defined in the project and is developed in WP2. The Grasshopper® components are a proof-of-concept integration of DURAARK's functionality into a 3rd party software in the architecture, engineering and construction (AEC) domain, carried out in WP7. They answer the demand and conclusion from D7.2, where stakeholders required that data ingestion and retrieval should take place as close as possible to their common work environment. Both graphical user interfaces demonstrate how to integrate the components developed in DURAARK into (3rd party) software products via the Service Platform's API.

1.1 Moving further from D2.4

The focus of development since the last deliverable was twofold:

1. The enhancement and update of the workflows that cover the use cases defined in D2.2.1 to provide stakeholders with an integrated and coherent user experience when using the DURAARK system¹.
2. The enhancement and update of the web services developed in D2.4 together with their APIs.

¹The “DURAARK system” here describes the whole software ecosystem developed in the project, containing the Service Platform and the graphical user interfaces WorkbenchUI and Grasshopper® components.

As a preparation for workflows enhancement the monolithic **web services** code base from D2.4 was refactored into a microservice-based² platform, the *Service Platform*. It categorizes the software prototype components described in D2.4, Section 3.2 into five *web services*. Each service groups one or multiple software prototype components into a single unit which is accessible via a coherent API. Each unit can be used independently of the others which allows to integrate either all functionality or only parts of it into existing software systems and workflows. The following list shows an overview of the five services and also shows which work package is responsible for component(s) which are hosted by the service (more details on each service are available in Section 4, together with a graphical overview in Figure 1):

- The *Session Service* handles a user session and manages a list of available input files (Responsibility: WP2)
- The *Metadata Service* hosts the metadata extraction components for E57 and IFC files (Responsibility: WP3)
- The *Semantic Digital Archive Service* provides access to the *Semantic Digital Archive Storage* component (SDAS) and the *Focused Crawler* component for the semantic enrichment of input files (Responsibility: WP3)
- The *Geometric Enrichment Service* groups the components for geometric enrichment, e.g., *IFC Reconstruction* and *RISE* (component for detecting in-wall electrical applicances). In M36 the service will be extended with the *Difference Detection* and the *Pointcloud Compression* components (Responsibility: WP4, WP5, WP7)
- The *Archival Transfer Service* provides functionality to create submission information packages (SIPs) and to upload data to the Rosetta digital preservation system (DPS) via the *Rosetta SIP Generator* and the *BagIt Generator* components (Responsibility: WP6)

²Microservices: <http://martinfowler.com/articles/microservices.html>

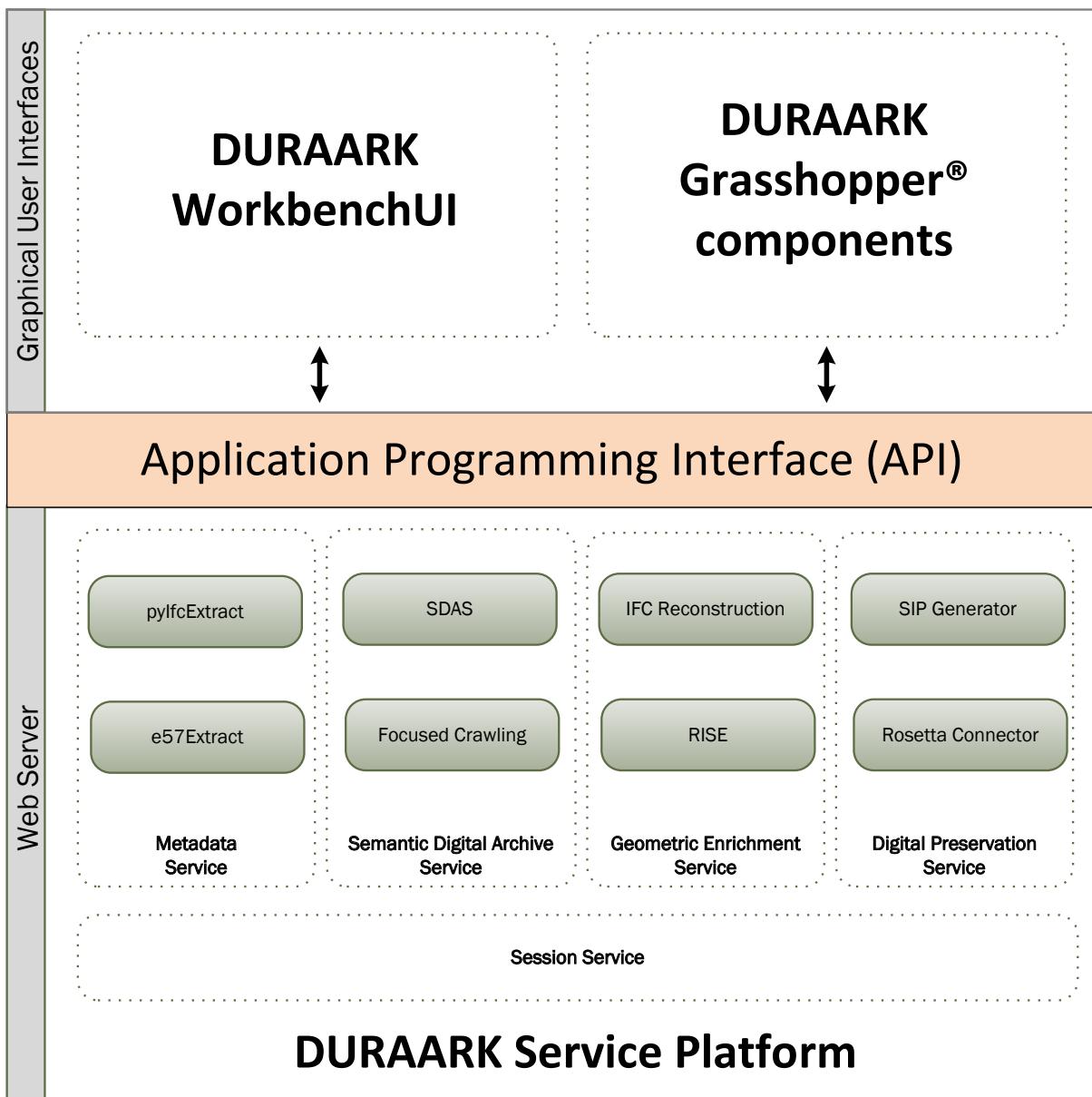


Figure 1: **DURAARK design:** Each *Service* is grouping together one or more *components* developed by project partners (the services and the respective components are described in detail in Section 4). The communication to graphical user interfaces is done via an application programming interface layer.

The WorkbenchUI serves as a graphical user interface which is described in D2.4, Section 3.1.2 was enhanced in functionality and robustness. The WorkbenchUI is the reference implementation of a graphical user interface for the Service Platform. The user interaction

within the WorkbenchUI is centered around *workflows*. The original five workflows in D2.4 were streamlined into a set of three workflows to better reflect the necessities derived from the use case description in D2.2.1. Each workflow contains steps a stakeholder carries out from within the WorkbenchUI. The following list gives an overview of the workflow set:

Pre-ingest Workflow This workflow consists of a sequence of steps a stakeholder has to perform to prepare 3D input data files representing a building for digital preservation. During those steps metadata for the input files is collected and the semantic and geometric enrichment is done. The output of the preingest workflow is a *Submission Information Package (SIP)* containing the 3D data files, the (enriched) metadata and the geometric enrichments. The Workbench provides the possibility to upload the SIP to the Rosetta digital preservation system or to download the SIP for local processing. Additionally the enriched metadata is stored into the semantic knowledge database (SDAS, see Section 4.3.1).

Retrieval Workflow The metadata stored in the SDAS during the archival workflow allows stakeholders to query the system in order to find stored buildings. The result of a query consists of metadata with links to the actual data files (if they were uploaded to Rosetta). After the retrieval of the files for a specific building the files and the metadata can be processed, i.e. to update metadata entries.

Maintenance Workflow The maintenance workflow takes care of the updating process of existing metadata and 3D files in the SDAS and the Rosetta DPS and ensures consistency between the two systems. The maintenance workflow is a “behind the scenes“ workflow, as its steps are not directly carried out by a stakeholder, but get triggered in the background when a stakeholder is updating data via the Workbench.

Additionally this document introduces the DURAARK Grasshopper® components. Here we show how to use the Service Platform to integrate DURAARK’s functionality into an existing, 3rd party software which is used by stakeholders in the AEC domain. Section 3 describes the workflow within Grasshopper to utilize DURAARK’s functionality. The workflow is derived from the demands of stakeholders summarized in D7.2.

1.2 Overview of this Deliverable

In Section 2 a description of the streamlined and updated set of three workflows is given. The section is written as a user manual which explains how to carry out the workflows in the WorkbenchUI. Section 3 then describes the workflow when using DURAARK's functionality within the Grasshopper® software. Section 4 describes the services which provide DURAARK's functionality to the WorkbenchUI, the Grasshopper® components and future 3rd party software. Section 5 gives an update on Decisions & Risks that were identified additionally to the ones described in D2.4. Section 6 updates the licensing information on the graphical user interface, services and used components. In Section 7 a conclusion is given which includes a description of next steps for the integrated software prototype towards M36.

1.3 Implementation Status

D2.5 is the last official deliverable for the integrated software prototype. As it is scheduled for M30, and not at the end of the project in M36, not all components are yet in a finished state. For this reason we give an overall status of the implementation here. This is necessary because for coherence reasons we describe workflow functionality in a few parts of the deliverable which is not fully finished yet or has to be implemented still. Section ?? provides a list of items that will be worked on towards M36, including the points shortly described here.

The following items in this section are outlining the (technical) functionalities necessary to perform the workflows described in Sections 2 and 3. For each functionality a description of the maturity level is given. The parts in the deliverable, where functionality is described which is not yet fully implemented, contain hints on that fact, too.

We decided to put this section at the beginning of the document, although the workflows are not yet formally introduced, to clearly lay out the implementation status of the integrated software prototype. The reader is advised to come back to this section after reading the document to assess the maturity level of the software at its current state.

The implementation plan for the missing functionality is detailed in our release planning. Appendix 2.4 contains links and information to the plan towards M36.

Pre-ingest Workflow Status The pre-ingest workflow is in a state to ingest data into the SDAS knowledge graph and the Rosetta DPS, which provides the base for the remaining workflows. Following list details the implementation status for the different steps in the workflow:

- The Metadata Extraction Service for *buildM* properties is fully functional, as well as the editing functionality of the metadata by the user.
- The Semantic Digital Archive Service is fully functional regarding the enrichment of a 3D input file with information topics and the storage of the information within the SDAS knowledge graph. The selection of enrichment candidates by a stakeholder is not yet implemented and will be available in M36.
- The *IFC Reconstruction* component is fully functional.
- The component for detecting electrical appliances within walls – *RISE* – provides all necessary functionality for processing the showcase pointcloud data set “Nygade“, which is available in the online demo of the WorkbenchUI (see 1.4). The processing of arbitrary pointcloud data sets will be available in M36.
- The generation of a Submission Information Package (SIP) from data produced in the pre-ingest workflow, except for the inclusion of the semantic enrichment data (the *buildM+* data). This will be available for M36.
- The transfer of the SIP to the Rosetta DPS is fully functional.
- The storage of the metadata produced in the pre-ingest workflow within the SDAS knowledge graph is fully working. We do not yet store the location of the archived 3D data files back to the SDAS knowledge graph. This functionality will be available in M36.
- The extraction of an extensive set of “building element level“ information from pointclouds is currently under development. This information will be gathered in

the pre-ingest workflow and will then be stored into the SDAS knowledge graph to enable stakeholders to create sophisticated queries on building element levels. Examples for such properties are ,e.g., floor count, wall count, area, etc. The full set of properties will be available in M36.

Retrieval Workflow Status The search functionalities for the DURAARK system are working. SPARQL queries can be sent via the Semantic Digital Archive Service to the SDAS knowledge graph and return the corresponding results, e.g., when querying for a building added to the arvhive via the pre-ingest workflow. For M36 the graphical user interface for executing queries will be improved, together with the possibility to upload custom queries to the system.

The retrieval of the 3D data files which are archived into Rosetta via the pre-ingest workflow is in its finishing state.

Maintenance Workflow Status The completed pre-ingest workflow and the perservation policies (available in M36) are a pre-requisite to complete and implement the maintenance workflow. Since the pre-ingest workflow was finished only recently only some parts of the maintenance workflow have been implemented in prototypical ways. Their completion is scheduled for M36.

Grasshopper® Workflows Status The Grasshopper® components described in D7.2 are fully working regarding their capabilities to carry out the design related use cases. For M36 the components will be extended to also support the pre-ingest and the retrieval workflow from within the Grasshopper® software.

1.4 Accessing the Software Prototypes

The integrated software prototype is a combination of a graphical user interface (WorkbenchUI) and a set of services (Service Platform) which provide the functionality developed in DURAARK. The WorkbenchUI and the services are hosted publicly. The source code is available on the Github code hosting platform, a discussion on the licensing is given in

Section 6. The following listing on how to access the software prototypes and on where to find more detailed information. For services the source code link also contains the API documentation:

WorkbenchUI

Type:	Javascript web service
License:	MIT
Description:	D2.5, Section 2
Demo application:	http://workbench.duraark.eu
Source code:	https://github.com/DURAARK/workbench-ui

Session Service

Type:	Javascript web service
License:	MIT
Description:	D2.5, Section 4.1
API documentation:	http://data.duraark.eu/services/api/sessions
API endpoint:	http://data.duraark.eu/services/api/sessions
Source code:	http://github.com/DURAARK/duraark-sessions

Metadata Service

Type:	Javascript web service
License:	MIT
Description:	D2.5, Section 4.2
API documentation:	http://data.duraark.eu/services/api/metadata
API endpoint:	http://data.duraark.eu/services/api/metadata
Source code:	http://github.com/DURAARK/duraark-metadata

Semantic Digital Archive Service

Type: Javascript web service
License: MIT
Description: D2.5, Section 4.3
API documentation: <http://data.duraark.eu/services/api/sda>
API endpoint: <http://data.duraark.eu/services/api/sda>
Source code: <http://github.com/DURAARK/duraark-sda>

Geometric Enrichment Service

Type: Javascript web service
License: MIT
Description: D2.5, Section 4.4
API documentation: <http://data.duraark.eu/services/api/geometricenrichment>
API endpoint: <http://data.duraark.eu/services/api/geometricenrichment>
Source code: <http://github.com/DURAARK/duraark-geometricenrichment>

Digital Preservation Service

Type: Javascript web service
License: MIT
Description: D2.5, Section 4.5
API documentation: <http://data.duraark.eu/services/api/digitalpreservation>
API endpoint: <http://data.duraark.eu/services/api/digitalpreservation>
Source code: <http://github.com/DURAARK/duraark-digitalpreservation>

pyIfcExtract Component

Type: Python Component
License: LGPL
Description: D3.3, Sections 4.3, 4.4
Source code: <http://github.com/DURAARK/pyIfcExtract>

e57Extract Component

Type: C++ Component
License: GPL
Description: D2.4, Section 3.2.2
Source code: <http://github.com/DURAARK/e57Extract>

SDAS component

Type: REST API Component
License: Creative Commons CC0 1.0 Universal Public Domain Dedication
Description: D3.3, Section 2
API endpoint: <http://data.duraark.eu/services/CrawlAPI>

Focused Crawling component

Type: Java component
License: LGPL
Description: D3.6, Section 2
Source code: https://github.com/bfetahu/focused_crawler

IFC Reconstruction Component

Type: C++ component
License: GPL
Description: D4.2, Section 4
Source code: Available on request from University of Bonn (UBO)

RISE component

Type: Javascript/C++ component
License: BSD
Description: D5.4
Source code: Available as part of the Geometric Enrichment Service repository

SIP Generator Component (LTU)

Type: Java component
License: BSD
Description: D2.5, Section 4.5.1
Source code: <http://github.com/DURAARK/sip-generator-ltu>

SIP Generator Component (TIB)

Type: Java component
License: BSD
Description: D2.5, Section 4.5.1
Source code: <http://github.com/DURAARK/sip-generator-tib>

Rosetta Connector Component

Type: Java component
License: BSD
Description: D2.5, Section 4.5.2
Source code: <http://github.com/DURAARK/rosetta-connector>

Grasshopper® components

Type: Components for the Grasshopper® software
Description: D7.2, Section 4.3
Source code: The components will be available for public download in M36.

1.5 Deviations from D2.4

The software architecture in D2.4 included the *PROBADO3D* component to store metadata generated within the pre-ingest workflow and the *Rosetta-PROBADO3D Connector* to handle the communication between the WorkbenchUI and the Rosetta DPS. It turned out that the SDAS (see Section 4.3.1) was fully capable of providing the functionality to store the metadata, and that PROBADO3D was unnecessary as additional component. Therefore we decided to drop the PROBADO3D component and use the SDAS component instead. The *Rosetta-PROBADO3D Connector* component was renamed to *Rosetta Connector* and handles the archival transfer between the WorkbenchUI and the Rosetta DPS.

2 WorkbenchUI Workflows

DURAARK defines a set of three workflows which are inspired by the *Preservation – Curation Lifecycle Model* described in report D2.2.3. Each workflow is addressing a subset of the use cases defined in D2.2.1. The WorkbenchUI allows to go through a workflow in a step-by-step manner. This section explains the steps for each workflow and lists the connected use cases. The data produced or worked on in a step is also described.

2.1 Pre-ingest Workflow

The pre-ingest workflow prepares the 3D data files so that they can be transferred to a digital preservation system in form of a Submission Information Package (SIP, see D6.6.1). The different steps in the workflow handle the selection of input 3D data files, the extraction, manipulation and enrichment of metadata and the creation of files containing additional geometric information. After the pre-ingest workflow is finished all necessary data files, enriched metadata and geometric enrichment files are available to create a SIP. The SIP can be transferred to the Rosetta digital preservation system or it can be downloaded locally. More information on the SIP generation can be found in Section 4.5.

2.1.1 Addressed Use Cases

The use cases related to this workflow in the context of *core long-term preservation tasks* are:

- UC1: Deposit 3D architectural objects
- UC9: Enrich BIM/IFC model with metadata from a repository

The use cases related in the context of *production and consumption-oriented* tasks are:

- UC4: Detect differences between planning state and as-built state³
- UC7: Plan, document and verify retrofitting/energy

Table 1 provides an overview of the software prototype components which contribute to use cases ordered by the hosting service. The table contains a reference to the deliverable which explains the component in more detail and also links the overview description in this deliverable, if available. Starting with M30 the deliverables contains a section in the introduction which explains the relation of the component to the use cases. For M30 D3.6 and D5.4 contain such a section explaining, how exactly the component adds to a use case. As mentioned in Section 1 the component *Difference Detection* is not yet available in the M30 version of the integrated software prototype. The component will be available in M36 and will cover UC4 in more detail. Currently the *RISE* component is contributing to UC4, described in D5.4.

The Sessions Service is left out of the table below as it serves as an internal management component for handling user sessions and files and does not contribute to a use case directly.

Use Case	Service	Component	Overview	Deliverable
UC1	Metadata	pyIfcExtract		D3.3, Sec. 4.3, 4.4
UC1	Metadata	e57Extract		D2.4, Sec. 3.2.2
UC1, UC9	Semantic Digital Archive	SDAS	4.3.1	D3.3, Sec. 2
UC1, UC9	Semantic Digital Archive	Focused Crawling	4.3	D3.6, Sec. 1.3, 2
UC7	Geometric Enrichment	RISE	4.4.2	D5.4
UC1	Geometric Enrichment	IFC Reconstruction	4.4.1	D4.2, Sec. 4
UC1	Digital Preservation	SIP Generator	4.5.1	D2.5

Table 1: Use cases related to the pre-ingest workflow addressed by different components ordered by the hosting service.

A listing of the licenses for each component and the services is given in Section 6, a link to the source code and to the service APIs can be found in Section 1.4.

³UC4 is mainly covered by the *Difference Detection* component, which is not yet integrated and will be available in M36, accompanied by D4.3.

2.1.2 User Manual

Figure 4 depicts the sequence of steps carried out by a stakeholder in the pre-ingest workflow. It shows actions a stakeholder can trigger in the WorkbenchUI and which events result from that action in the Service Platform layer. Additionally, the figure shows which data is worked on or produced in each step. Each step is described in more detail in the following paragraphs.

Selection of a session As a first step the user selects a session from the two provided showcases, i.e. “Haus 30” (a building in Berlin) and “Nygade” (a building in Denmark). Figure 2 shows the selection page, which is shown when opening <http://workbench.duraark.eu>. For the following steps (except for the “Electrical Appliances Detection” step) we assume that “Haus 30” is selected. After opening the session a list of IFC and E57 files is presented⁴. The user selects the files which should go into the session (see Figure 3). For “Haus 30” he selects the BIM model Plan3D_Haus30_PREVIEW.ifc and a point cloud scan of a floor of the building, Plan3D_Haus30_130-v2-0G_PREVIEW.e57. On Service Platform level a new, unique folder is created in the internal storage, which is a container for the data in the sessions and which will be added to a SIP at the end of the workflow. All services share the same internal storage to allow file-based information exchange between the components in a straightforward manner. The selected input files are stored in the newly created session folder. The creation of the session folder and the storage of the metadata and additional 3D data files which will be created in the next steps is done by the Session Service. An additional role of the Session Service is the management of the input 3D files which are selectable by the user in this step.

⁴The files are located on the server the Service Platform is running on. The root folder with files is configurable.

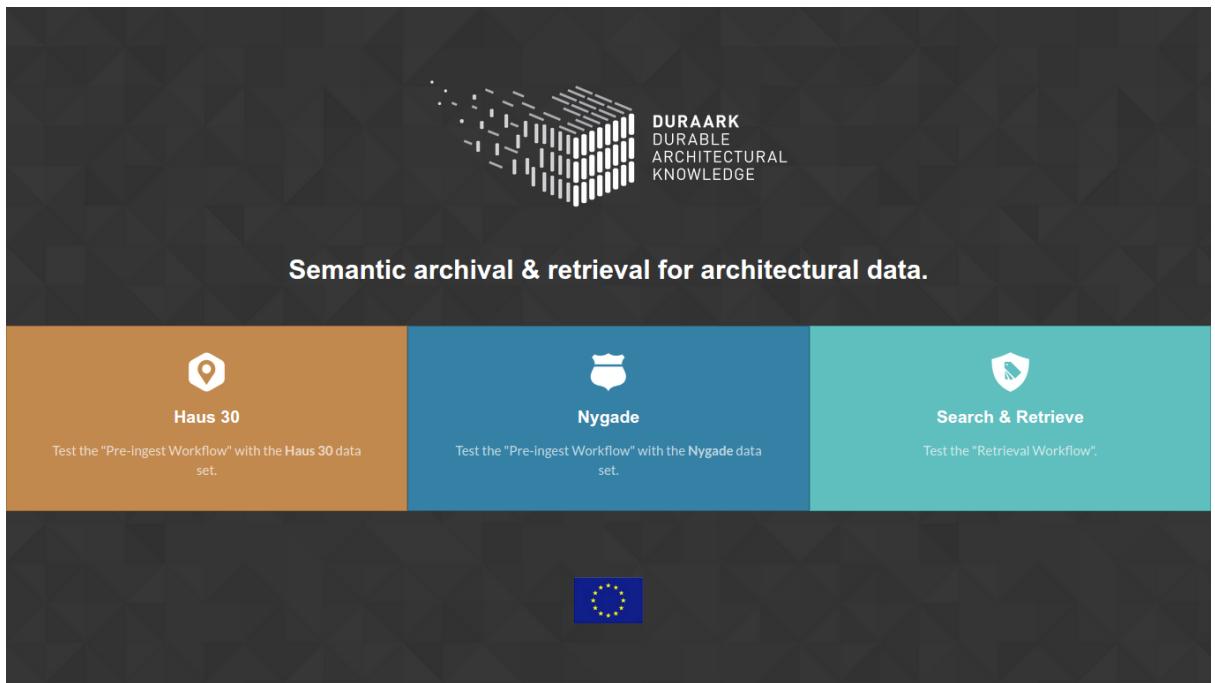


Figure 2: Pre-ingest Workflow: Selection of a session

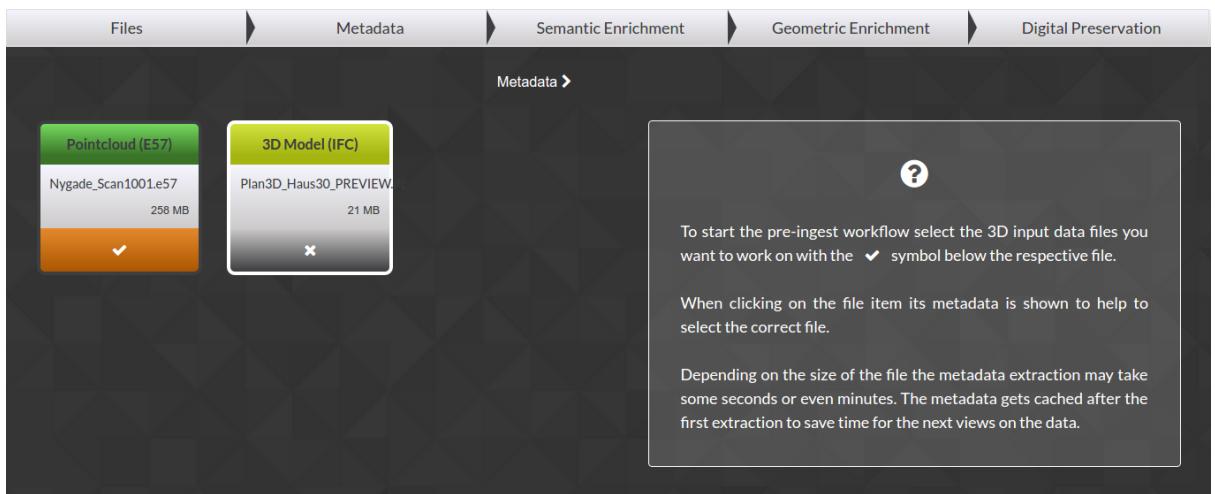


Figure 3: Pre-ingest Workflow: Selection of input files

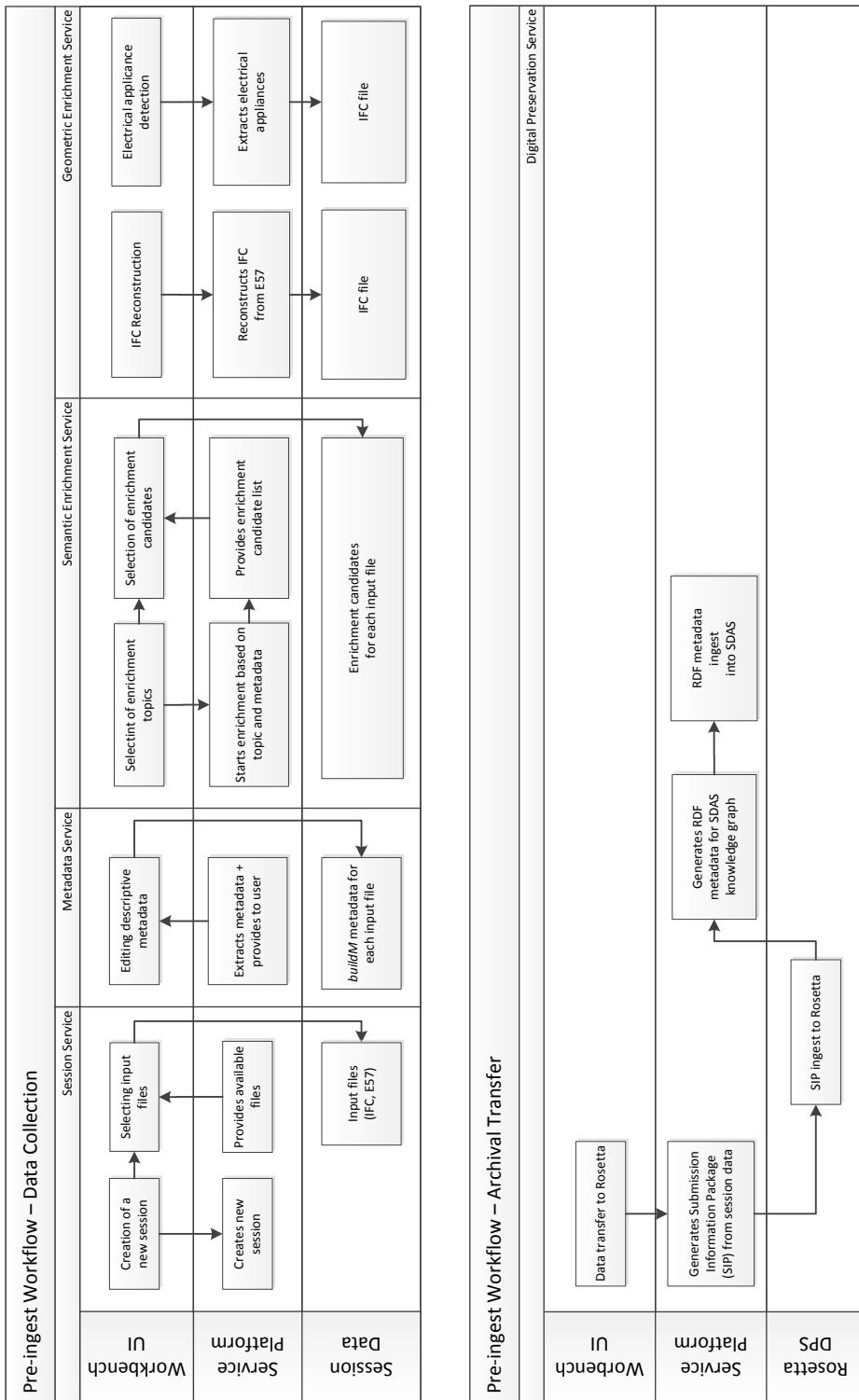


Figure 4: Pre-ingest Workflow

Editing descriptive metadata In a next step the Metadata Service extracts technical and descriptive metadata from the two selected files. The metadata contains information on the building itself, as well as information directly related to the file. For instance, metadata from the `Plan3D_Haus30_PREVIEW.ifc` file contains the *latitude* and *longitude* of the building, which is directly related to the “Haus 30” building. The metadata in the file also contains the *file schema*, a property describing internals of the file itself, but which not related to the building. To manage the different metadata types DURAARK developed the `buildM` schema (see Section 4.3.1). It contains properties which are related to a building, e.g., latitude and longitude. Building related properties are summarized as **buildM:PhysicalAsset**. For metadata describing file related properties like the file schema (e.g., “E57”), buildM defines **buildM:DigitalObjects**. In the WorkbenchUI properties related to **buildm:PhysicalAsset** as “Building Metadata”, properties related to **buildM:DigitalObjects** as “Digital Representation Metadata”.

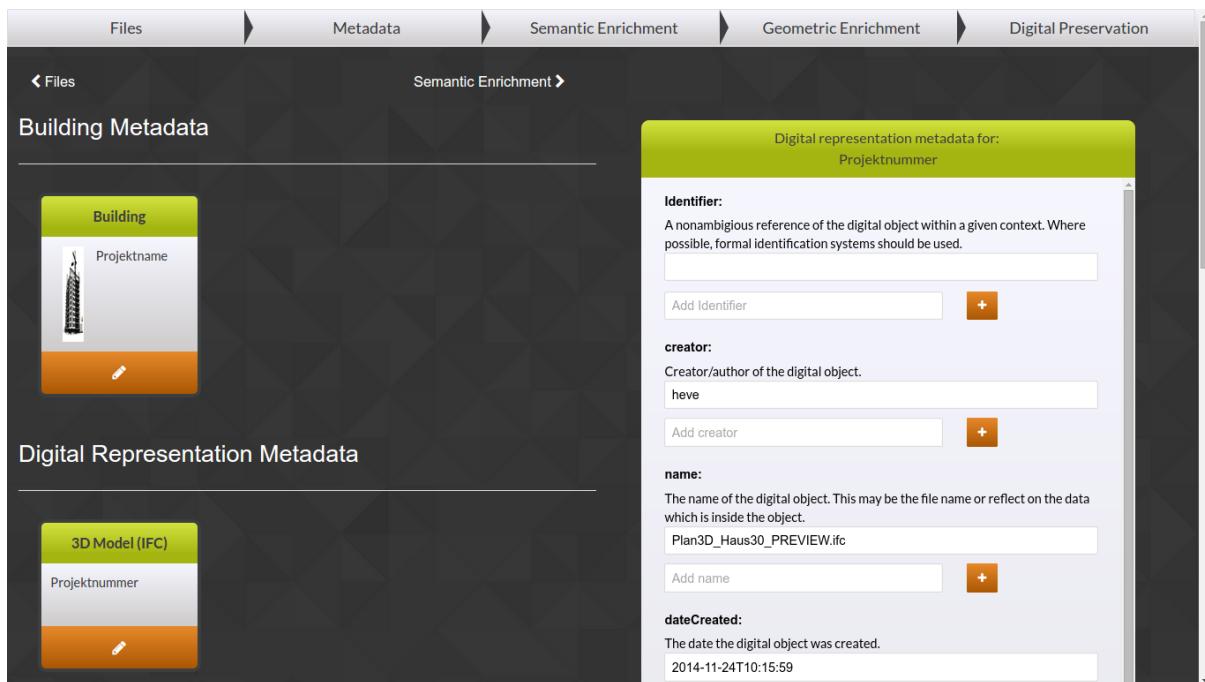


Figure 5: Pre-ingest Workflow: Editing descriptive metadata

Selection of enrichment topics After the previous steps the session contains all metadata information on the input files. Now the stakeholder can enrich the existing metadata of each input file with additional “smart” information, so called *information topics*. When selecting an enrichment topic, information related to the building is searched in public data sources like DBpedia⁵. The WorkbenchUI provides a set of enrichment topics tailored for the use with our example data sets, in this case for “Haus 30”. The first topic is enriching the file with information on the building, architecture and region around the building (which is located in Berlin). The second topic focuses on information on the political context in the region the building is located. When the stakeholder selects one or multiple topics the Semantic Digital Archive Service starts the enrichment process in utilizing the *Focused Crawling*⁶ component (see Section 4.3.2). The duration of the enrichment process depends on the information topics, therefore the user is provided with a visual notification when the process is finished. He can continue with the workflow while the enrichment process is running in the background.

In M36 the WorkbenchUI will provide a possibility to provide stakeholders to store custom information topics in the system. With this feature it will be possible to create topics which are useful in the daily work of a stakeholder who is using the pre-ingest workflow. For instance, for a renovation company it makes sense to add a topic which enriches the building with information on the used materials.

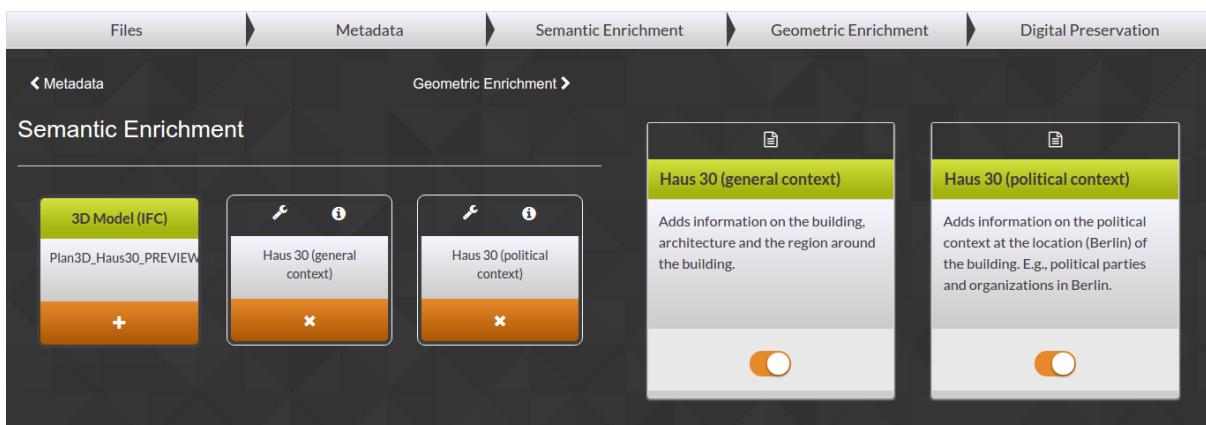


Figure 6: Pre-ingest Workflow: Selection of enrichment topics

⁵DBpedia: <http://wiki.dbpedia.org/>

⁶The *Focused Crawling* component calls an enrichment topic a “seed list” in D3.6. For the user interface the term “information topic” is used for easier transportation of its behaviour.

Selection of enrichment candidates The result of the previous enrichment is a list of unified resource identifiers (URIs) which contain information related to the building, so called “enrichment candidates”. The user is presented with the URIs (see Figure 7) and can inspect the URI in clicking on the link. A new page is opened which contains related information. For M30 all found URIs will be added to the archive. In M36 a sorting and selection mechanism will be implemented to allow for a selective enrichment of useful information, depending on the stakeholder’s need.

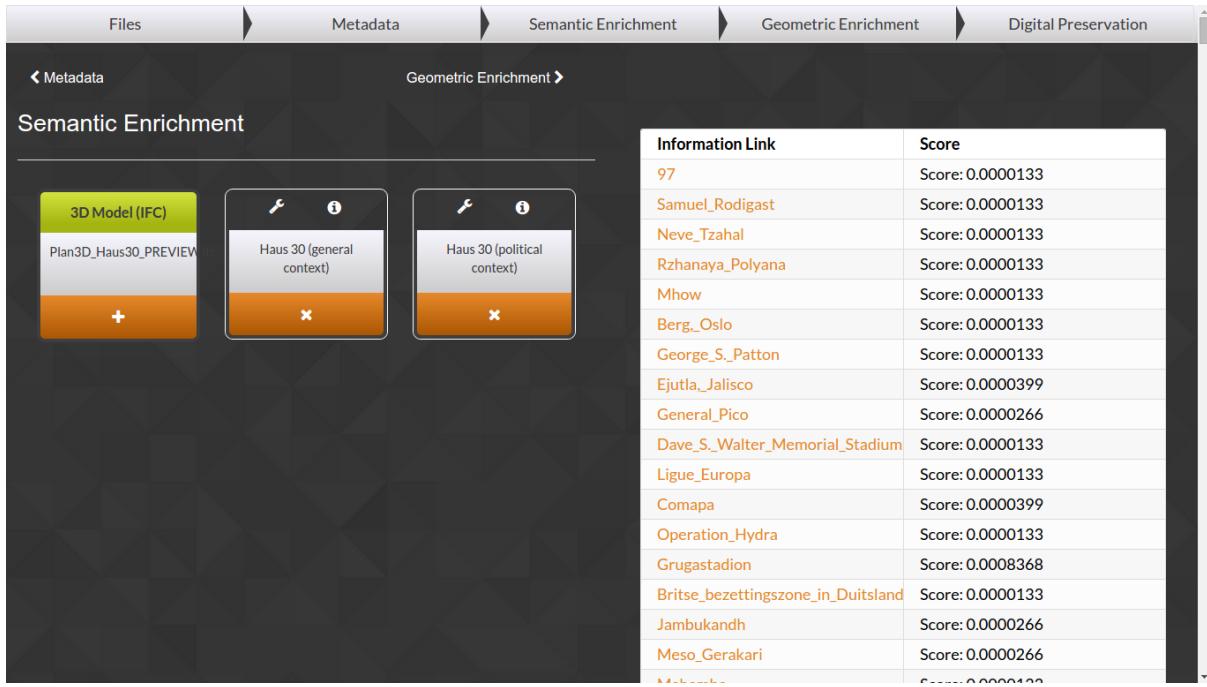


Figure 7: Pre-ingest Workflow: Presentation of the enrichment candidates

IFC Reconstruction After the semantic enrichment the geometric enrichment is performed. Currently two components are available from the Geometric Enrichment Service, the *IFC Reconstruction* (see Section 4.4.1) and the *RISE* component for the detection of electrical appliances (see Section 4.4.2), which are depicted in Figure 8. In M36 two additional components – *Difference Detection* and *Pointcloud Compression* – will also be provided by the service. The Difference Detection component is already usable in the Grasshopper® software (see Section 3). For the integration into the WorkbenchUI a fully automated registration between IFC and pointcloud files is a pre-requisite, which is under development. See Section 5.2 for a discussion on the risk of this development.

For each E57 input file the IFC Reconstruction component can be selected to reconstruct a BIM model as an IFC file from it. After the reconstruction the resulting IFC file gets stored to the session data.

In M36 the reconstruction process will additionally extract building level information like the number of floors and rooms, the amount of space, etc. from the pointcloud data and store it in the IFC file. Making the data available in the IFC file has two reasons:

- An arbitrary IFC viewer can use or show this additional information.
- At the end of the pre-ingest workflow the Metadata Service will extract the building element level information from the IFC and store it into the Semantic Digital Archive Service, which will allow stakeholders to use this information also to search for buildings (see [2.2](#)).

The IFC file is marked as *Derivative copy* of the original E57 point cloud file. Marking files as *Derivative copy* is necessary for the creation of the SIP at the end of the pre-ingest workflow.

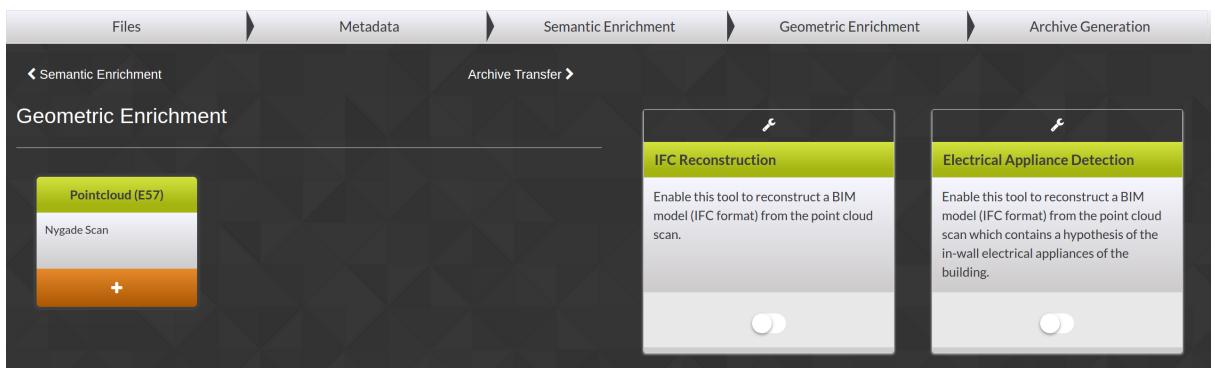


Figure 8: Pre-ingest Workflow: Geometric Enrichment Tools

Electrical appliance detection For each E57 point cloud file an IFC file can be reconstructed which contains a hypothesis of the in-wall electrical appliances. The responsible component for detecting those “nearly invisible structures” is the *Reveal Invisible StructurEs (RISE)* component (see Section [4.4.2](#)). The result of this step is again an IFC file, enriched with information on the in-wall electrical appliances. The produced

IFC file is marked as Derivative copy of the originating E57 point cloud file. For this step we assume that the stakeholder has selected the “Nygade” session, as the component only supports this data set at the moment.

As a stakeholder you have to enable the tool by clicking on the respective item in Figure 8 first. An additional item is appearing next to the E57 point cloud file, indicating that the detection of the in-wall appliance started in the background. A loading spinner appears during the processing. When it finishes, an information icon shows that the process has successfully finished. Clicking on the icon presents the stakeholder with an overview of the results (see Figure 9), where the extracted hypothesis, the used rule set for the installation zones and the detection results of the visible sockets are shown.

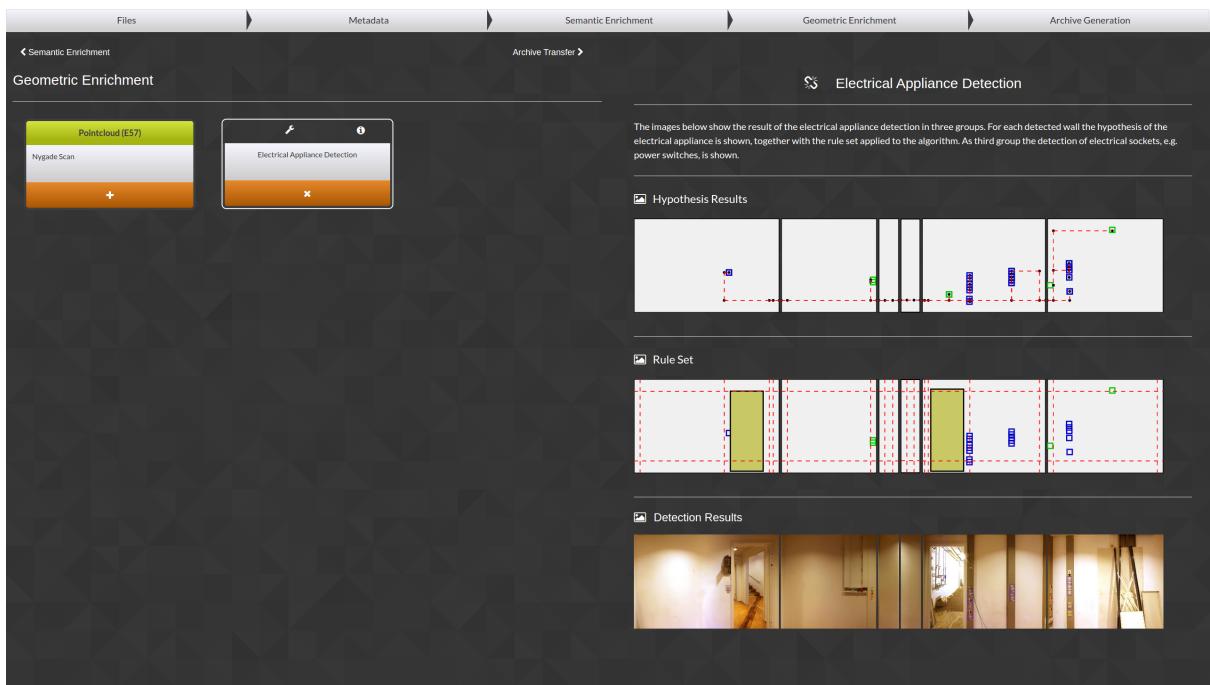


Figure 9: Pre-ingest Workflow: Electrical appliance detection results

Data transfer to Rosetta At this point the session contains all necessary data to generate a SIP. The session data is provided to the Digital Preservation Service which internally uses the *SIP Generator* component (see Section 4.5.1) to create a SIP from the data. The SIP contains all necessary information for the Rosetta DPS to process the data and to ingest it. If the ingest is successful the data is accessible via a set of web services

provided by Rosetta.

After a successful archival in Rosetta the Digital Preservation Service ingests the descriptive metadata and the semantic enrichment data into the SDAS knowledge graph, making the data searchable in the retrieval workflow. This step is essential as it is the connection between the archival and the retrieval of building information.

In M36 the building element level information extracted from a point cloud will also be added to the SDAS knowledge graph. The connection of the metadata in the SDAS to the actual 3D files stored in Rosetta will also be available in M36. Currently it is possible to search for the archived buildings, but it is not yet possible to retrieve the data files from that information⁷.

Figure 10 shows a screenshot of the GUI to transfer the data to Rosetta.

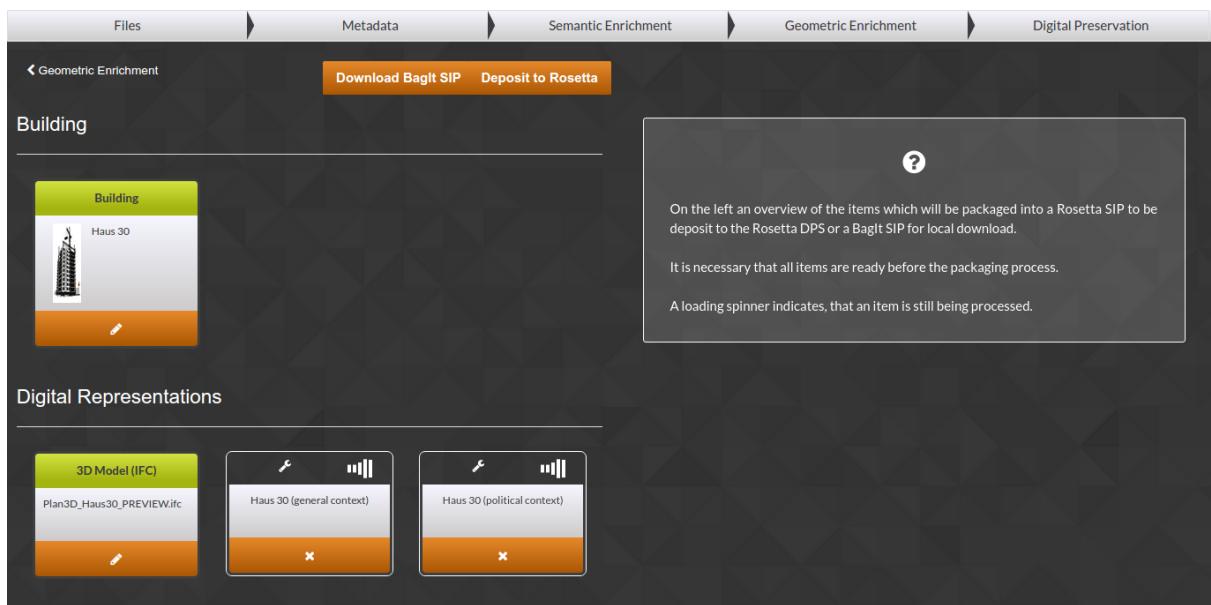


Figure 10: Pre-ingest Workflow: Data transfer to Rosetta

⁷The implementation of this feature will be available in a few weeks, though, to provide all necessary parts to start the implementation of the maintenance workflows (see Section 2.3)

Local storage as BagIt file The session data can also be stored on the local harddisk. In this case the SIP is created as “BagIt”⁸. In this scenario the user is responsible for the actual preservation and maintenance of the data. The procedure to ingest the descriptive metadata into the SDAS is the same as described in the “Data transfer to Rosett” step. The only difference is that the link to the actual data file will be a local one, meaning that after a search for a building in the retrieval workflow the data file for a found building can not automatically be retrieved from the Rosetta system. With the information in the metadata it is possible to contact the creator of the BagIt file, and to negotiate further steps for the access of the data file.

2.2 Retrieval Workflow

The retrieval workflow allows stakeholders to use the WorkbenchUI as a flexible search system for architectural data. The Semantic Digital Archive Service provides access to the knowledge graph developed in DURAARK, the SDAS (see 4.3.1). It contains a continuously growing information pool of buildings and their surrounding context, e.g., geographic, historic or legal context. The pre-ingest workflow directly adds or updates data in the SDAS knowledge graph, e.g., in using the *Focused Crawling* component to add information topics related to a building. Additionally, for each building instance ingested into the SDAS knowledge graph a background task is automatically enriching the metadata with additional information. See D3.6 for more information on that.

There are three categories of properties which are stored in the SDAS knowledge graph and which serve as search criteria for user defined queries:

Metadata properties During the pre-ingest workflow descriptive metadata for a building (Physical Asset) and for its digital representations (Digital Objects, i.e. IFC and E57 files) are extracted, edited and stored as buildM⁹ instances in the SDAS knowledge graph.

⁸BagIt: <https://tools.ietf.org/html/draft-kunze-bagit-11>

⁹buildM schema: https://github.com/DURAARK/Schemas/blob/master/xml/2015_03_30_duraark_buildm_2_2.xsd

Properties from the semantic enrichment During the semantic enrichment in the pre-ingest workflow the user can select information topics he finds important for the building, e.g., information on the geographic or historic context. The Semantic Digital Archive Service utilizes the *Focused Crawling* component to search for information on this topic which are related to the building. This information is then stored in the SDAS knowledge graph.

Properties from the geometric enrichment During the geometric enrichment in the pre-ingest workflow “building element properties” will be derived from the point cloud files and stored in the SDAS knowledge graph. Those properties (e.g., number of floors, number of storeys, etc.) are then available for stakeholders as search criteria in the search interface of the WorkbenchUI. This functionality is currently under development, and will be available through a combination of the *IFC Reconstruction* and the *pyIfcExtract* metadata extraction component in M36.

The WorkbenchUI leverages the information in the SDAS knowledge graph to provide stakeholders with the possibility to combine the properties of the three types in their search queries. This gives stakeholders fine-grained control what to search for. The power of the system lies in the possibility to dynamically add new information to a building, e.g., if a building owner is going to renovate multiple buildings, he might be interested in legal restrictions on what changes to the building are allowed. In this case he can add information on the legal context of the buildings via the semantic enrichment step in the pre-ingest workflow and then use the retrieval workflow to explore this information. The pre-ingest workflow step “Select enrichment topics” describes how to add such information to new or existing buildings.

Queries are very diverse and depend on the stakeholder’s need. They reach from simple queries like “list all buildings which are located in Berlin” to more sophisticated ones where multiple properties are used, e.g., list all buildings which have the architectural style of “Art Deco”, are designed by the architect “Ludwig Hoffmann” and are located in “Berlin”. The search system uses SPARQL¹⁰ as query language. SPARQL queries are very flexible and allow fine-grained control how to combine properties into a search query. The WorkbenchUI in M30 provides a set of example queries to showcase the functionality. In M36 a graphical interface for visually creating queries will be added.

¹⁰SPARQL: <http://www.w3.org/TR/sparql11-overview/>

The result of a search query is a list of unified resource identifiers (URIs) which contain information which fits the search criteria. The stakeholder can open the URI link to inspect the information behind it. If the result is a building that was ingested via the pre-ingest workflow the according 3D files which are stored in Rosetta will be accessible. This feature is not yet available in M30, but will be in M36.

The stakeholder has the following possibilities to further process the result:

- Load a building into a new pre-ingest workflow session to change, add or update data (see Section 2.3 on how the system handles the update of data)
- Download the E57 pointcloud or IFC files of a building to process them further in the Grasshopper®¹¹ software. DURAARK provides a set of plugins for Grasshopper® which allow sophisticated processing of pointcloud and IFC files with the functionality to perform IFC reconstructions, difference detection between files, subsampling, etc. Those functionalities are also provided within the WorkbenchUI, but having them exposed as Grasshopper plugins allows stakeholders to work with the files in a software they already know and also provides more fine-grained control with the direct manipulation of the 3D data. The plugins and the corresponding workflows are described in Section 3.
- Process the files outside of the WorkbenchUI and Grasshopper with a tool of choice. The changed files can be stored back into the DURAARK system via the pre-ingest workflow. The internal mechanisms to update the 3D data files in the system are described in Section 2.3.

The execution of search queries (and the saving of query templates in M36) is provided by the Semantic Digital Archive Service. Via the service's API stakeholders can also directly integrate the search mechanism into their own software or workflows, e.g., to build a company-internal visual query editor focused on a sub-set of the information stored in the SDAS knowledge graph.

Figure 11 shows a graphical representation of the workflow.

¹¹<http://www.grasshopper3d.com/>

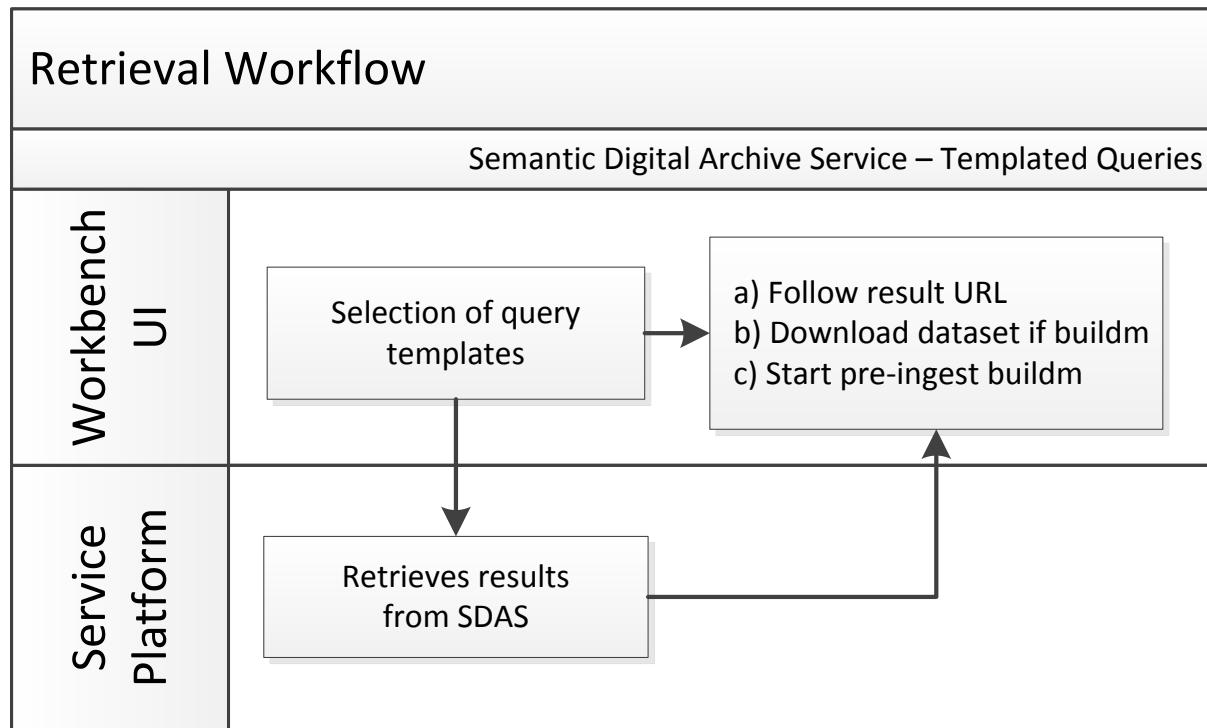


Figure 11: **Retrieval Workflow**

2.2.1 Addressed Use Cases

The retrieval workflow is essential to nearly all of the defined use cases. The use cases related to this workflow in the context of *core long-term preservation* tasks are:

- UC2 - Search and retrieve archived objects

The use cases related in the context of *production and consumption-oriented* tasks are:

- UC4: Detect differences between planning state and as-built state¹²
- UC5: Monitor the evolution of a structure over time
- UC6: Identify similar objects within a point cloud scan

¹²UC4 is covered by the *Difference Detection* component, which is not yet integrated and will be available in M36 accompanied by D4.3.

- UC7: Plan, document and verify retrofitting/energy
- UC8: Exploit contextual information for urban planning

Table 2 provides an overview of the software prototype components which contribute to use cases ordered by their hosting services, together with additional references to the respective component. For components taking part in the preparation of the queryable metadata see Table 1. As already stated in 2.1.1 the *Difference Detection* component will be available in M36. UC4 is covered by this component, so a detailed description on how the component adds to the use case will be provided in D4.3. The *SDAS* component is implicitly adding to all use cases in the retrieval workflow, as it provides the metadata for the cases.

Use Case	Service	Component	Overview	Deliverable
all	Semantic Digital Archive	SDAS	4.3.1	D3.3, Sec. 2
UC2, UC5, UC6, UC7, UC8	Semantic Digital Archive	Focused Crawling	4.3	D3.6, Sec. 1.3, 2
UC4, UC5, UC7	Geometric Enrichment	RISE	4.4.2	D5.4
UC4, UC5, UC6	Geometric Enrichment	IFC Reconstruction	4.4.1	D4.2, Sec. 4

Table 2: Use cases related to the retrieval workflow addressed by different components ordered by the responsible service.

A listing of the licenses for each component and the services is given in Section 6, a link to the source code and the service APIs can be found in Section 1.4.

2.2.2 User Manual

Selection of query templates The WorkbenchUI provides templates for a set of queries to showcase the search functionality. See Figure 12 for a screenshot of the search GUI. When a stakeholder selects a query template the Semantic Digital Archive Service generates a SPARQL query which is sent to the SDAS component. A list of matching results is returned and displayed in the GUI. In M30 the results can be inspected in clicking on link. In M36 stakeholders can either download the files from the list or mark them as input files to create a new pre-ingest session to update the data. The mechanisms for updating existing data in the SDAS and the Rosetta DPS and the different update scenarios are described in 2.3.

The query templates provided in the WorkbenchUI are a subset of the templates described in D3.6, Section 5. A detailed description of the queries can be found there, together with the corresponding SPARQL query.

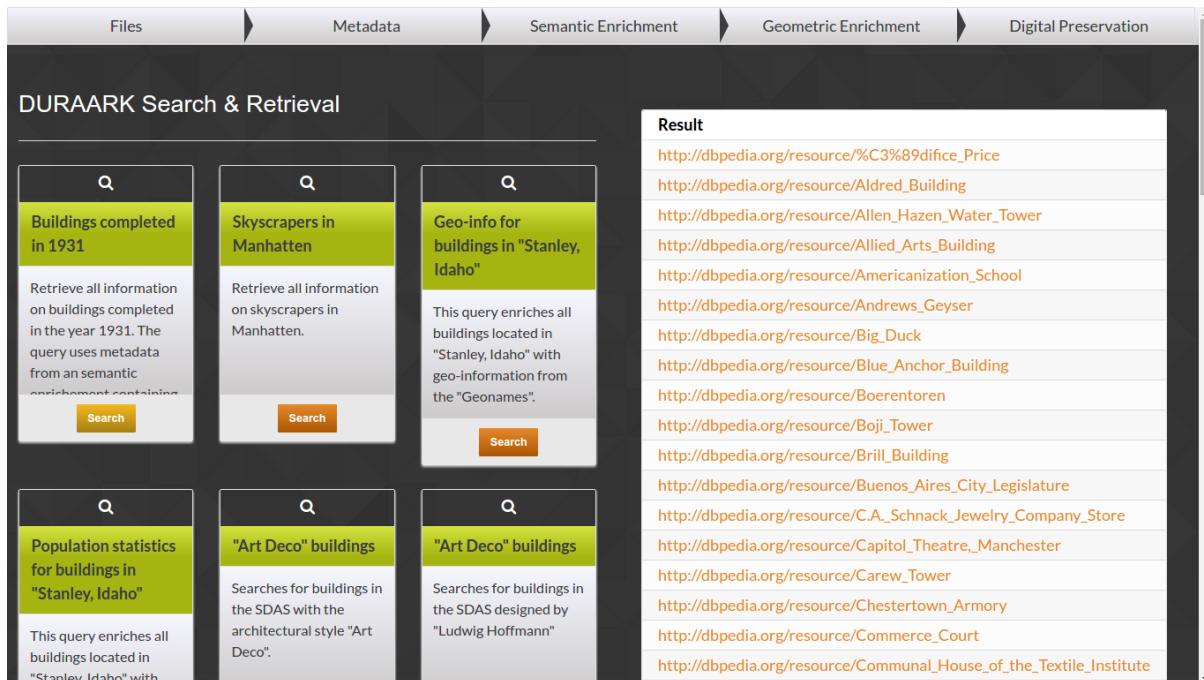


Figure 12: Retrieval Workflow: Selection of query templates

2.3 Maintenance Workflow

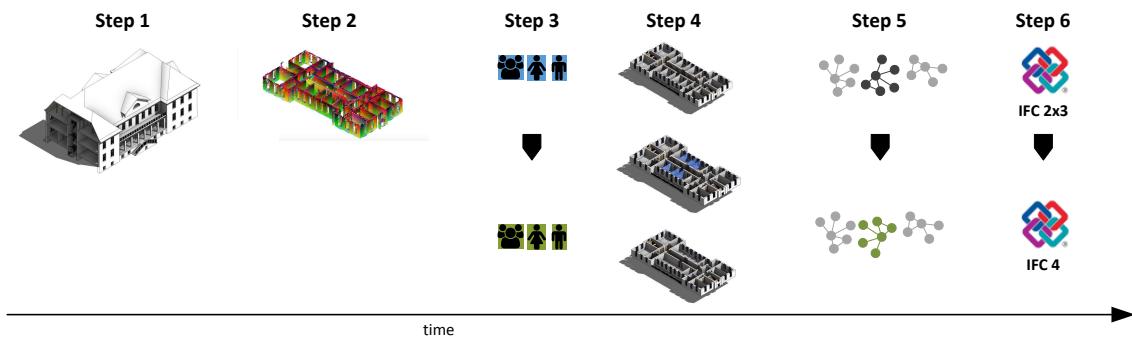


Figure 13: Overview of different lifecycle stages of the Haus 30 dataset illustrating necessary maintenance steps.

The steps associated with maintenance can be roughly categorized into two parts:

- maintenance of individual Archival Information Packages (AIP, see D6.6.1, Section 2.1) (Steps 1 – 4 below)
- maintenance of commonly shared resources such as vocabularies and datasets used for the semantic enrichment and referred to from the AIPs. (Steps 1, 6)

The maintenance workflow plays an important role in the following use cases:

- UC2 - Search and retrieve archived objects
- UC3 - Maintain semantic digital archive
- UC4 - Detect differences between planning and as-build-state
- UC5 - Monitor the evolution of a structure over time
- UC7 - Plan, document and verify retrofitting/energy

Those use cases all depend on the data stored in the SDAS knowledge graph and in the Rosetta DPS. Therefore the maintenance workflow – which takes care of consistent data updates in the SDAS and Rosetta – is implicitly adding to them.

Maintenance steps are mainly carried out by DURAARK system administrators and curators specialized in the general infrastructure (as opposed to the preservation of concrete artifacts). For this reason the description of the workflow is not written in the “user manual” style like the two workflows before.

The particular preservation challenges and necessary steps associated with dynamic, “living” artifacts such as buildings will be illustrated with an example scenario illustrating the (fictive) example lifecycle of the “Haus 30“ dataset which is depicted in Figure 13. The following paragraphs explain the scenario over the buildings lifetime and describe the necessary maintenance steps.

Step 1: Initial archival of an IFC file

A user uploads an initial IFC file from a newly constructed building via the pre-ingest workflow described in Section 2.1. A unique identifier for the building is generated and stored in the respective **buildM:PhysicalAsset**¹³ node that is created within the SDAS at the end of the pre-ingest workflow.

Step 2: Update with a new pointcloud file

Some years later, a water damage has caused structural problems to some walls in the building. A laser scan is performed and the respective pointcloud files are registered with the IFC files and stored as newly ingested E57 files into the archive. The new AIP is associated to the same **buildM:PhysicalAsset** in the SDAS. Queries of the SDAS asking for all AIPs associated with a particular Physical Asset (e.g., “Haus 30”) can then be filtered to include only the latest datasets or dataset including a particular file

The challenge in this step (adding a pointcloud file to an existing IFC model) is the correct identification of the respective **buildM:PhysicalAsset** to augment it with another pointcloud file, which is a **buildM:DigitalObject** in terms of the buildM schema.

¹³The previous text in this deliverable denoted a building as “Physical Asset“. In this workflow section we write the term as **buildM:PhysicalAsset** to make clear that we mean the buildM schema property “PhysicalAsset“, which is important in this context. The same goes for the term “Digital Object“)

At first the stakeholder searches the archive for the building with help of the retrieval workflow, e.g., by entering its street address into the generic search form (see Section 2.2). The search result is the URL to a buildM instance which corresponds to the building. The stakeholder opens a new session in the WorkbenchUI with this buildM instance. Re-using steps from the pre-ingest workflow in Section 2.1, the stakeholder adds a new file to the building, which creates a new **buildM:DigitalObject** entry in the buildM instance. Metadata for the **buildM:PhysicalAsset** might also be modified by the stakeholder. The (modified) **copy of the buildM** instance is stored into the SIP created at the end of the pre-ingest workflow and the **differences** of the modified instance are stored in the SDAS as RDF triples referring to the existing buildM instance that was initially created in Step 1. This is realized through a separate *Named Graph* that carries metadata such as timestamps and users. By searching through the SDAS across all available named graphs, all additions can be found by the user. By using complete copies of the shared information as it is stored in the (BagIt or Rosetta) AIPs, the archived versions however are stored as independent complete packages that could be accessed on their own. Links to "earlier" versions of the building (from Step 1) are stored in two different ways:

1. links to the respective named graphs in the SDAS. Metadata and RDF content of each version of the building is stored in a separate named graph in the SDA. Each new version then contains an RDF statement pointing to this named graph resource containing the earlier version.
2. IDs of the AIPs in the DPS. In the metadata of the newer version the ID / URL of an earlier archival package that is contained in the DURAARK archive is added to the metadata of the new version.

The principal technical approaches of these delta operations have been illustrated in D3.3, Section 4.4 "Versioning of evolving data sets" on page 29.

Step 3: Change of ownership or other metadata

Some time later, the owner of the building changes and an according new entry is made into the buildM instance.

While Step 2 included the addition of **buildM:DigitalObjects** and a modification of the metadata of the respective **buildM:PhysicalAsset**, in this step the user only introduces changes to the metadata without ingesting new digital objects. The user searches and retrieves the respective “Haus 30” instance in its *most recent state* (by calculating the deltas of potential earlier versions) and is presented with the metadata record using the WorkbenchUI to allow the edition of descriptive metadata as shown in 5.

The changes are stored only in the SDAS, not as a new AIP. More information on policies when triggering AIP creation will be presented in D6.3, due in M36.

Step 4: Building modification

The new owner decides to use the building in a slightly different function. For this purpose, a few of the inner wall are removed and the surplus doors are closed with interior gypsum walls.

Similar to Step 2, in this step an additional **buildM:DigitalObject** is stored alongside the existing representations from Step 1 and Step 2. However, in this case the changing state of the building is not represented by an augmented geometric enrichment (the point-cloud) but with an additional IFC file with removed walls and covered doors. This not only changes the descriptive metadata of the **buildM:PhysicalObject** (number of rooms) but also introduces metadata “contradicting” predecessor versions.¹⁴ Technically, the new version is ingested as if it were a regular new item in the archive as described in Step 1. In the ingest however, earlier versions of the building already residing in the archive are selected in the WorkbenchUI and indicated as predecessors. A new buildM record is created that

¹⁴Ideally, this workflow would include the retrieval of the existing IFC file from the archive, modification in an IFC-capable end-user CAD/BIM tool (ArchiCad, Revit etc.) and an export that is structurally identical apart from the introduced changes. In practice however, such scenarios are currently rather unlikely, since even the same tools are not implemented rigidly enough to allow IFC based round-tripping modifications. This issue becomes even harder when considering different tools with different versions that might even operate on different underlying IFC schema versions.

uses the same **buildM:PhysicalObject** for this **buildM:DigitalObject**. However, for the specific attribute (e.g., number of rooms) the new values are assigned to the respective buildM attributes. Using difference detection (e.g., via SPARQL queries) on the different metadata records stored in the SDA and assigned to the same **buildM:DigitalObject**, the user can search for the evolution of the building.

Step 5: Modification of referenced datasets

Along the way, the vocabulary used for the classification of building elements has changed, and its predecessor version is not provided under the same unified resource identifier (URI) by the owner. This is relevant as during the semantic enrichment step in the pre-ingest workflow the stakeholder selected enrichment topics which were stored into the SDAS knowledge graph and which contain external linked datasets referring to the now unavailable vocabulary. In this step the administrator of the Rosetta DPS system is notified by the Semantic Digital Archive Service that the externally linked datasets will be updated. A new snapshot of the changed external vocabularies is stored into the Rosetta DPS. The DPS now contains two different versions of the external datasets used for the enrichment. Due to the complexity of the task, a complete migration of all changed resources however is not archiveable in a fully automated way and is considered out of scope.

Step 6: Evolution of file formats (migration)

In this step, a new IFC schema version or E57 file format specification has reached a maturity state and acceptance in the market so that the usage of the predecessor versions becomes unlikely. The decision to migrate the file format is left to the maintainer of the archive. In case the decision is made to migrate the system maintainer triggers a search for all candidate AIPs that contain legacy/outdated versions of the respective file formats and schedules them for migration. When a stakeholder then searches for a building after the migration is done he can set a filter in the query to be presented with representations in a certain format only. Notice however, that such migration steps will be lossy in some cases where backwards-compatibility was not of central interest to the respective standardization organization.

3 Grasshopper® Workflows

The WorkbenchUI is a system for ingesting architectural data into the DURAARK system and to retrieve data from it via the two workflows “Pre-ingest” and “Retrieve”. This allows the WorkbenchUI to cover the use cases defined in DURAARK. It is also possible to use geometric enrichment components in the WorkbenchUI to create new datasets bases on input files, e.g., in using the *IFC Reconstruction* and the *RISE* component. However, “creators of data” such as architects and engineers in offices or the facility management of institutional building owners, indicated in D7.2 that they want to be able to directly manipulate the IFC and E57 files and use them together as part of their daily work and common working environment. This is a requirement the WorkbenchUI cannot directly satisfy. This stakeholder demand motivated the development of the Grasshopper® components, which provide specialized tools to directly work with the 3D data files in the digital work environment of stakeholders. Therefore, the Grasshopper® components were developed as a proof-of-concept implementation to

- provide tools within the DURAARK system to manipulate 3D data files in a working environment certain stakeholders are familiar with and
- provide the same functionality regarding pre-ingest and retrieval workflow like in the WorkbenchUI (which is a target for M36).

3.1 Addressed Use Cases

The development of the Grasshopper® components is contributing to the following use cases in the context of *core long-term preservation tasks*¹⁵:

- UC1: Deposit 3D architectural objects
- UC2: Search and retrieve archived objects

¹⁵Those will be implemented in M36.

The use cases related to these workflows in the context of *production and consumption-oriented* tasks are:

- UC4: Detect differences between planning state and as-built state
- UC5: Monitor the evolution of a structure over time
- UC7: Plan, document and verify retrofitting/energy

D7.2 describes in detail which workflows are supported and relates them to the respective use cases. Section 3.3 below gives one example for UC7.

3.2 Processing Building Data in Stakeholder Workflows

WP7 gave insights how stakeholder select, prepare and synthesize data for the use in building projects and how the DURAARK system could integrate and support these. The general practices of preparing building data for use in design or retrofitting projects takes place in the following steps (see Figure 14):

- **Retrieve Data**

Information formats ranging from photos, drawings to BIM and database objects on the building object is collected from different sources, such as public archives from municipalities or state, building owner, literature and other sources such as the web.

- **Select and Analyse**

The different information is mapped to each other, analysed for relevance and eventually scaled, filtered or enhanced to focus on the certain aspect or location under consideration of the task, as for instance for the upgrade of the electrical appliances on a certain section of a building.

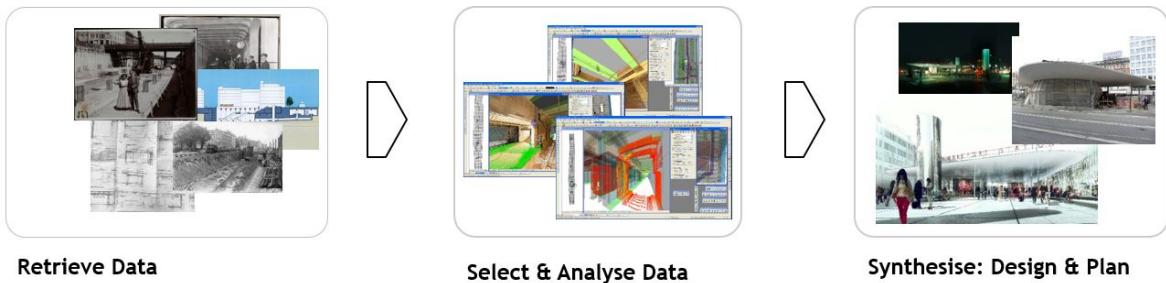


Figure 14: General steps to select and refine building data - Dataset: Nørreport Station Copenhagen / Grontmij

- **Synthesis: Design and Plan**

A coherent state of the selected building information is generated, which typically describes the state of the building at a certain moment in time. This is usually the present state. The data represents the ground truth for all further planning and often as well construction activities. The creator of this data hence bears the responsibility for the quality and completeness of the information.

D7.2 furthermore showed that the feature-set in currently available software in the AEC domain for working with IFC and E57 is limited when using those two formats combined within a software. Stakeholders want to perform data refinement steps, as well as to select and merge data from both file formats. Current software does not provide easy-to-use interfaces for those tasks yet. This provided further motivation for the development of a Grasshopper® based integration of the DURAARK services to evaluate their modularity, their function and the ability to deploy them in the software stakeholders are already using for processing of these data types. The implementation is a proof-of-concept on how to carry out workflows defined in DURAARK within Grasshopper®. The feature-set of the Grasshopper® components not only provides the functionality to refine, compare and select within and between pointcloud and IFC data, but will provide means to retrieve and ingest data from and to the DURAARK system directly in utilizing the Service Platform. This activity is planned until M36.

The Grasshopper® components currently fill the functionality gap between the retrieval of datasets with the WorkbenchUI and the design and planning activities with this data (see Figure 15). It therefore integrates the DURAARK system in the daily workflow and

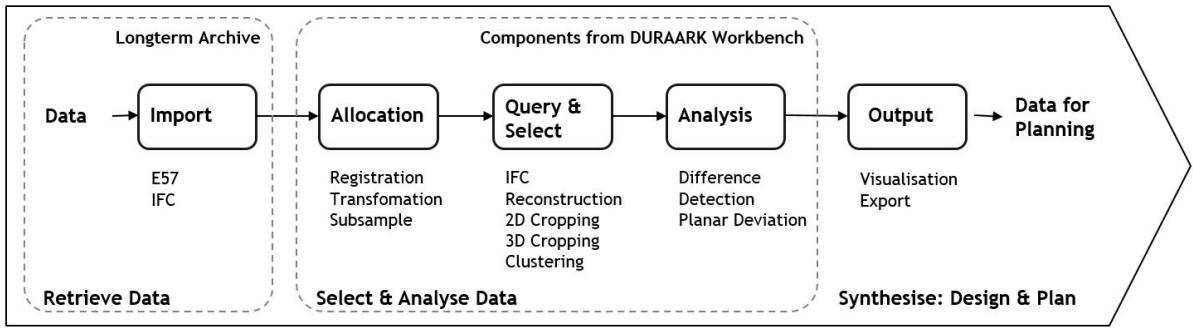


Figure 15: Overview of the developed DURAARK Grasshopper® components in relation to the overall flow of building data for design and reconstruction tasks of buildings.

in the work environment of stakeholders.

In general the Grasshopper® environment consists of all sort of components which can be arranged and coupled together in an input/output manner to fit the needs of a stakeholders project and process. Some of the DURAARK Grasshopper® components described in this section are directly integrating DURAARK component functionality via the Service Platform, e.g., the IFC Reconstruction component. Others pre-process (e.g., refine) data in order to make it usable with the DURAARK components and the stakeholders workflow and do not use the Service Platform but are necessary for the preparation of data (a detailed descriptions of the available DURAARK Grasshopper® components and their functionality can be found in D7.2, Section 4.3).

3.3 Workflow Example

D7.2 describes all the workflows that can be carried out within the Grasshopper® software in the DURAARK context. In this section we exemplify the use of the Grasshopper® components within a renovation project by an architect (UC7). Following the above introduced schema (see Section 3.2) the simplified steps of such a workflow are:

- **Retrieve Data**

The stakeholder has an up-to-date E57 pointcloud file of a building as a starting point and wants to use the DURAARK system to enrich his/her planning process for the renovation. A click on the DURAARK Workbench UI component in the



Figure 16: An E57 pointcloud as seen in the Grasshopper® software.

Grasshopper® software takes the stakeholder to the retrieval workflow of the WorkbenchUI, where files related to the project can be searched and retrieved (E57 and/or IFC files). The architect finds an older BIM model and an up-to-date newly archived E57 pointcloud file. These files are downloaded and imported to Grasshopper® in Figure 16 (note that this connection to the WorkbenchUI is planned for M36, but it is described here to have a full workflow from beginning to end).

- **Select and Analyse** The stakeholder wants to check for differences between the old BIM model and the current state of the building represented by the pointcloud in order to have a solid base for further planning.

- **Allocation**

The stakeholder uses the DURAARK Grasshopper® components for *Registration*, *Transformation* and *Subsampling* in order to merge and register the retrieved E57 pointcloud with the BIM model (see D7.2, Section 4.3.2 for details).

- **Query and Select** Using the *Cropping* component the stakeholder crops out



Figure 17: The difference between a BIM model and an E57 pointcloud. The architect realizes a large difference (red indicates large differences, green no or only small differences) and reconstructs a new BIM model using the *IFC Reconstruction* component.

the relevant part of the E57 pointcloud. (see D7.2, Section 4.3.3 for details)

- **Analysis** Using the *Difference Detection* component the stakeholder investigates the difference between the planning model and the E57 pointcloud. The stakeholder realizes a large difference between the planning model and the E57 pointcloud (see Figure 17 and D7.2, Section 4.3.4)
- **Query and Select** Using the *IFC Reconstruction* component the stakeholder reconstructs an IFC model from the E57 pointcloud.

• Synthesise: Design and Plan

The final step of the consolidation of the models and the start of the actual architectural planning process takes place when the stakeholder merges the models by attaching the desired metadata from the planning model to the reconstructed model. The enriched, reconstructed model becomes the base for the further design.

DURAARK Grasshopper® components also support the planning stage in utilizing

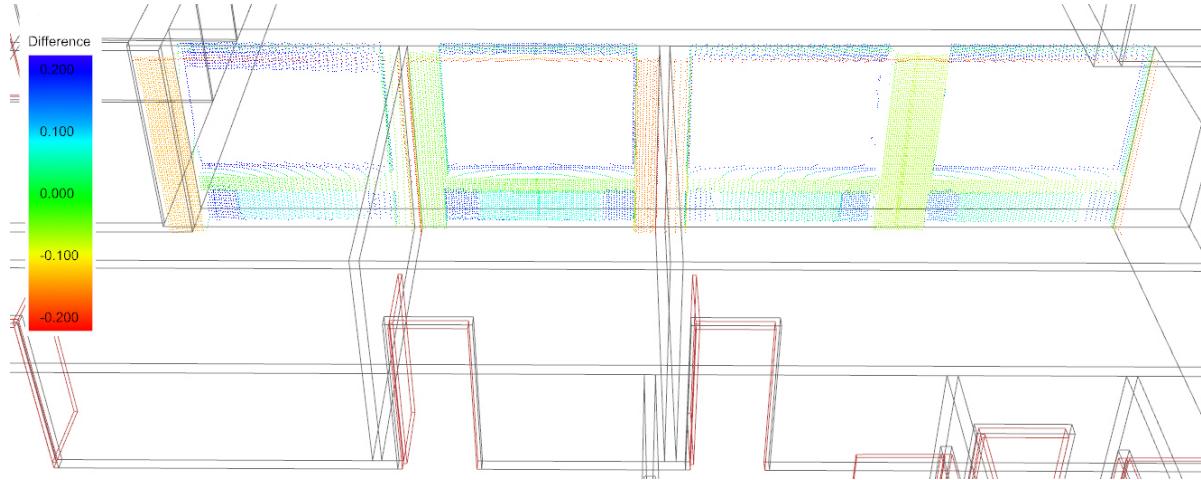


Figure 18: The deviation between a BIM wall element and the related point cloud. This information can be passed on from the architect to the, e.g., engineer through the DURAARK system.

the *Planar Deviation* component to get detailed measurements and visualisations of the deviation between the “real world” and the planning model (see Figure 18). This can for example be used to inform the responsible construction company that a deviation has to be fixed.

- **(Re-)Ingest Data**

In order to document the state of the building the architect ingests the pointcloud and the reconstructed model with information on the found deviation into the DURAARK system (see Figure 18). A click on the Grasshopper® *Ingest* component takes the stakeholder to the pre-ingest workflow of the WorkbenchUI where new files can be ingested to the DURAARK system (note that this component is planned for M36).

The DURAARK system here serves here as a long-time archive, but can as well be used to create a document which presents an immutable state of a design or planning process.

4 Service Platform

When working with 3D data in the Architecture, Engineering, and Construction (AEC) domain the produced data files tend to become very large. Depending on the overall size of a building, the size of the scans ranges from a few gigabytes up to a few terabytes. The ongoing trend towards improved acquisition hardware with increasing resolution is likely to further increase the data volume. E.g. the pointcloud scan of our show case dataset “Haus 30” is roughly 60GB in its original size. Depending on the architectural site and project, multiple of those files need to be handled. Regarding storage space this is a solvable problem, however, when it comes to transmitting those large amounts of data over a network connection outside of the company’s network the transmission times are a problematic factor to be taken into account.

In DURAARK the developed functionality is encapsulated into the *Service Platform* which is hosted on a web server. Before a stakeholder can use, e.g., the WorkbenchUI to work with 3D files these files have to be uploaded to the server. Depending on the file sizes and network bandwidth the amount of time necessary to transmit 3D data files (especially pointcloud scans) is not acceptable from a user experience point of view.

For this reason DURAARK follows the approach to bring the services to the data, not the data to the services. That means that the DURAARK Service Platform is designed to be easily deployable near (in a network topology sense) the file servers hosting the 3D data a user wants to work with. Due to the inherent nature of a distributed web application it is then possible to connect from the WorkbenchUI to one or multiple Service Platforms. This also means that the WorkbenchUI as graphical user interface is independent from the Service Platform, and can be hosted on a different server. This will allow multiple stakeholders like architects, constructors or facility managers to collaborately work together on a session where the data files are distributed over multiple servers in (future) versions of the WorkbenchUI.

A second benefit of the chosen approach is the possibility to adapt and integrate the services and components almost seamlessly into the environments of stakeholder. The implementation efforts are expected to be low for web-based platforms of stakeholders. However especially tools for 3D modeling and pointcloud processing on the stakeholder side are not web-based, but use powerful standalone software on desktop machines. In

Section 3 we exemplified one approach on how to integrate the DURAARK platform into existing software environments of stakeholders and shows how a DURAARK archive could be a natural part of stakeholder workflows.

For the DURAARK Grasshopper® components – which are running on the local computer within the Grasshopper® software – we recommend to install the Service Platform locally on the same computer. The reason being is that the processed 3D data files will be displayed in the Grasshopper® software and have therefore to reside locally. If Grasshopper® is connecting to a remote instance of the Service Platform the files have to be downloaded before display, resulting in long download times for large files and in a bad user experience (see discussion at the beginning of this page). Although the Service Platform would run on the local computer then, it will still provide the same pre-ingest and retrieval functionality as provided in the WorkbenchUI, as it will simply connect to the remote components SDAS (see Section 4.3.1) and Rosetta DPS (see Section 4.5.2).

Figure 1 shows the Service Platform and the graphical user interfaces WorkbenchUI and Grasshopper®. Each service is hosting components behind an application programming interface (API). A more detailed description of the design can be found in Appendix 1. The following sections describe the services and components available within the DURAARK Service Platform and is an update to D2.4, Section 3. The components for the Metadata Service are not included anymore, as there have been no substantial updates since D2.4. The API documentation for the different services can be found on the respective Github page listed in Section 1.4.

4.1 Session Service

The Session Service is a small, though very central service that has two purposes:

Session Management A *session* is created by a user to work on a building (i.e. Physical Asset) and its related IFC or E57 files (i.e. Digital Objects). During the pre-ingest workflow a set of components (accessed via their hosting service) creates additional data and metadata that is stored within the session. All services of the Service Platform have access to the stored data and can also add and change it. At the end of the pre-ingest workflow the session data is used to create a SIP package.

File Management The files which are available within the WorkbenchUI are managed by this service. For the M30 prototype we support files stored locally on the server running the Service Platform. Until M36 the service will be extended to be able to connect to the **bimsync**¹⁶ software, which is serving IFC files via an API. bimsync is developed by the consortium partner CATENDA, which is also responsible for this extension work.

4.2 Metadata Service

During the pre-ingest workflow the extraction of metadata from IFC and E57 files is handled by the Metadata Service. Depending on the file type the service delegates the work to the *pyIfcExtract* component for IFC files and the *E57Extract* component for E57 files. The *pyIfcExtract* component is described in D3.3, Section 4.3 and Section 4.4, the *E57Extract* component is described in D2.4, Section 3.2.2. The result of the respective tool is stored within the metadata service, which allows to export the metadata from a building as *buildM* XML during the SIP creation process in the pre-ingest workflow, or as RDF serialization to store the metadata into the SDAS knowledge graph after a successfull archival of the session data into the Rosetta DPS (see Section 4.5 for more detailed information on the archival process.).

4.3 Semantic Digital Archive Service

This section describes the core components of the SDA as part of the semantic enrichment, namely the SDA Storage (SDAS) and the Focused Crawler, which is introduced and described in detail in report D3.6.

¹⁶<https://bimsync.com/>

4.3.1 SDAS

In this section, the main functionalities of the Semantic Digital Archive Storage (SDAS) within the DURAARK system are described. The SDAS serves as central storage for all semantic and geometric metadata within DURAARK, and as such, contains a continuously growing *knowledge graph* of buildings, their digital models and their context, where the latter covers geometric information as well as semantic information about, for instance, the geographic, historic or legal context. Data within the SDAS can be roughly categorised into the following three categories:

1. **Primary metadata of digital objects and physical assets:** the SDAS serves as a repository of metadata describing the physical assets (buildings) and their context themselves as well as the data object representing them. As such, it is an index and catalogue providing information about the buildings preserved in the DURAARK system as a Linked Data set.
2. **Geometric metadata:** the SDAS captures some baseline geometric information about the shapes and structures captured by the described digital assets, as provided by WP3, WP4 and WP5.
3. **Semantic enrichments:** targeted crawls retrieve related background knowledge from the Linked Data graph about data captured in the SDAS. This includes specifically data further describing the geographic, environmental or structural context of the captured physical assets. Cross-domain reference graphs such as DBpedia and Freebase are used together with more focused datasets with clear temporal or regional focus. While external datasets evolve, crawling has been chosen as method of choice to capture related information within the SDAS.

Schema

Each of the categories described above adheres to a different schema. While the first category (*base metadata for digital and physical assets*) is expressed using a well-defined vocabulary, namely the *buildM* schema introduced in D6.2, the *semantic enrichments*, i.e. crawled context graphs are following arbitrary vocabularies used in their source datasets,

for instance, the DBpedia ontology¹⁷ or the Geonames ontology¹⁸.

The *buildM* schema, a central vocabulary for annotating digital models and physical structures, primarily describes the concepts *Digital Object*¹⁹ and *Physical Asset*²⁰, where suitable vocabulary terms are derived from a number of existing vocabularies (see D6.2). As such, the population of *buildM* instances defines the core of the SDAS and serves as central registry of buildings and their digital models, which are further enriched with contextual background knowledge from our focused crawling/enrichment functionalities. The overall combination of a set of *buildM* instances describing a particular asset and their corresponding context graphs (crawls) is referred to as *buildM+* (D6.2).

Implementation

The SDAS prototype implementation is based on a *triple store*, i.e. a NoSQL graph database for RDF data²¹. Specifically, due to its performance and scalability, the SDAS uses the OpenLink Virtuoso²² triple store as the database backbone for the SDAS. It provides a mechanism for persistent storage and access of RDF graphs. The triple store supports RDF data serializations such as RDF/XML as well as N3/N-triples.

Populating the SDAS

The SDAS is being populated primarily with data produced by the metadata extraction and geometric/semantic enrichment components described in this deliverable. Additionally, the DURAARK WorkbenchUI provides a GUI to manually configure crawls for enriching data in the SDAS.

Figure ?? provides a general overview of the focused crawling module for populating the SDAS. The crawling process is initiated by first providing a seed list that encodes the *information need*. The seed list is in the form of entity URIs, usually coming from publicly available knowledge bases (i.e. DBpedia). The crawler is triggered through the APIs

¹⁷<http://dbpedia.org/ontology/>

¹⁸<http://www.geonames.org/ontology>

¹⁹<http://data.duraark.eu/vocab/buildm/DigitalObject>

²⁰<http://data.duraark.eu/vocab/buildm/PhysicalAsset>

²¹The term 'quad store' on the other hand includes an additional context column per RDF triple that allows the fine grained capturing of context, provenance and named graphs. These however can also be employed using the more common triple stores

²²<http://virtuoso.openlinksw.com>

described in the following section.

In addition, new data can be added to the SDAS by using the built-in interface to the triple store implementation used in the prototype²³.

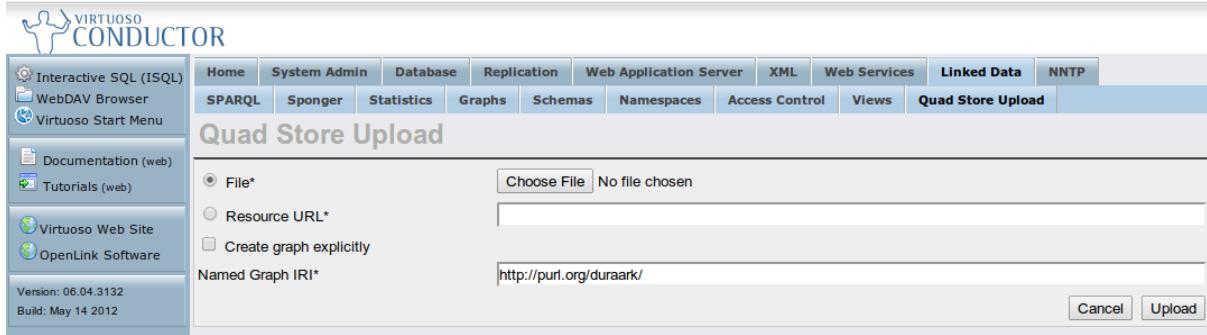


Figure 19: The Virtuoso Conductor UI for inserting data into the SDAS.

At present, the following steps need to be followed in order to insert triples into the store.

- Log into the Virtuoso UI²⁴ as an administrator.
- Set destination of the RDF store to insert the triples.
- Set the RDF IRI (Internationalized Resource Identifier)²⁵.

Querying the SDAS

The data stored in the SDAS can be queried using the following SPARQL endpoint: <http://data.duraark.eu/sparql>. The SPARQL endpoint is exposed both as visual web interface and as a REST API, where queries are directly transmitted as part of the HTTP GET request. The Semantic Digital Archive Service is proxying requests from the WorkbenchUI (or 3rd party software) to this endpoint. Having the SDAS as a remotely accessible component allows the Service Platform to be installed locally (e.g., when using the Grasshopper® components) and still have access to the SDAS knowledge graph²⁶.

²³Virtuoso Conductor

²⁴<http://meco.l3s.uni-hannover.de:8829/conductor/>

²⁵<http://www.w3.org/TR/rdf11-concepts/#section-IRIs>

²⁶The same principle holds true for the remote Rosetta DPS component. In this case the Digital Preservation Service is communicating with the remote Rosetta DPS.

Figure 20 shows the corresponding SPARQL query editor GUI, where the graph IRI can be specified alongside the query itself. The query in the figure retrieves all IFC files that describe buildings in *Chicago*.

More example queries showcasing the potential of the SDAS prototype, can be found at <http://data-observatory.org/sdas/>.

Virtuoso SPARQL Query Editor

Default Data Set Name (Graph IRI)
http://purl.org/duraark/

Query Text

```
SELECT ?ifc
WHERE { ?s dura:object_identifier ?ifc.
         ?ifc dura:hasLoc "http://dbpedia.org/resource/Chicago"}
```

(Security restrictions of this server do not allow you to retrieve remote RDF data, see [details](#).)

Results Format: (The CXML output is disabled, see [details](#))

Execution timeout: milliseconds (values less than 1000 are ignored)

Options: Strict checking of void variables

(The result can only be sent back to browser, not saved on the server, see [details](#))

Figure 20: The Virtuoso SPARQL query editor.

4.3.2 Focused Crawling

This section describes the second version of the *Focused Crawling* component which is implementing the *semantic enrichment* functionalities of the SDA. With respect to the initial prototype, significant improvements have been designed and implemented. Specifically, we introduce a focused crawler for linked data (detailed description in D3.6), which replaces the previously developed crawling environment with a more targeted and hence scalable approach. Crawls are either based on (a) manually defined seed lists, for instance, to retrieve relevant linked data subgraphs about the geographic, historical or infrastructural context of buildings and their model or (b) automatically extracted seeds, directly derived from existing *buildM* instances. Based on experimentally defined crawl configurations, we introduce an efficient means to crawl linked data of relevance to the specific instances in the SDAS. The focused crawler is exposed via the Semantic Digital Archive Service to the WorkbenchUI and other 3rd party software. The Semantic Digital Archive Service is internally using the API of the *Focused Crawling* component.

4.4 Geometric Enrichment Service

The general idea of the Geometric Enrichment Service is to derive – in an automatic or semi-automatic manner – high-level, semantically meaningful information from raw, low-level data. For example, pointcloud scans essentially are a set of unstructured points as observed from a scanning device. This data does not directly yield a deeper insight into the building's structure such as walls, number of rooms, locations of doors, etc. We aim to make this information available in a form that can be

1. automatically understood and analyzed by, e.g., architectural software, and
2. further modified by the user on the level of building elements for, e.g., planning of retrofitting tasks.

The IFC file format is able to store the geometric information of a building and also can encode higher-level information such as relations between building elements, measurements, as well as more abstract metadata such as time schedules for construction or renovation tasks. The Geometric Enrichment Service helps the user by providing components for the highly automatic generation and derivation of high-level representations from low-level data. The gathered information is stored in the generated IFC files, e.g., when using the IFC Reconstruction component to reconstruct geometry out of a pointcloud file, as well as into the SDAS knowledge graph. There the extracted quantitative metadata such as the number of rooms, room areas, volumes, etc. enables searching and filtering of the datasets using these criteria.

4.4.1 IFC Reconstruction

The IFC Reconstruction component derives IFC files from indoor point cloud scans in a highly automatic manner. The goal is to provide the user with easy to use means for the reconstruction of building models which provide a deeper insight into the building's structure, enable automatic searching and analysis (e.g., taking of measurements), and allow further processing in industry-standard architectural software for, e.g., planning of retrofitting. To achieve this goal, we developed and implemented an approach which

reconstructs the building using volumetric entities such as walls and their relations (e.g., how walls are connected). This directly follows the paradigms implemented by the IFC format and thus enables storing the results in native IFC format. An example for a reconstruction performed using our software is shown in Figure 21. The current prototype of this component is described in D5.3; we will further develop our method and software towards the third version of the prototype which will be part of D5.5 in M36.

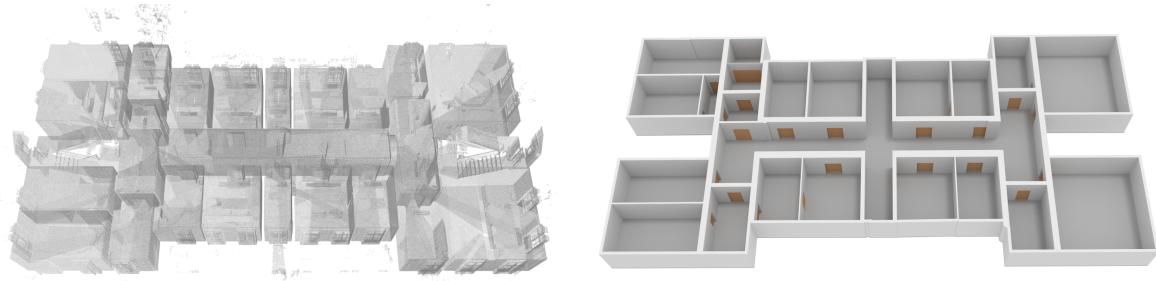


Figure 21: Automatic reconstruction obtained using the IFC reconstruction component. Left: Input point cloud of a building storey consisting of multiple scans. Right: Reconstruction of rooms, walls and doors.

4.4.2 Reveal Invisible StructurEs (RISE)

The *Reveal Invisible Structures (RISE)* component is responsible for detecting nearly invisible structures within point cloud files. Its target is to detect in-wall electrical appliances. The input to the component is a point cloud file and a panoramic image for each scanner position in the building. From the imagery it is possible to detect light switches, power sockets and other objects from pre-trained classes. Based on these visible elements of the electrical appliance a shape-grammar based algorithm creates a hypothesis of the in-wall electrical appliance. The outcome of the algorithm is dependent on the context the building was built in. For instance, in Germany there is a standardized rule-set on where to position electrical appliances in walls. The standard was released at a certain date, so one can assume that for a building in Germany, if built after the standard release date this rule-set applies. The *SDAS* component contains information on the location and the time the building was constructed and could be used to determine the correct rule-set for in-wall appliances, if existing. For the M30 integration of the RISE component into the WorkbenchUI no automatic search for an eventual rule-set is implemented, but

the information on the building context is available in the system and accessible via the WorkbenchUI.

D5.4 gives an extensive description of the RISE component.

4.5 Digital Preservation Service

As described in previous deliverables, the WorkbenchUI allows Submission Information Package generation in two formats: BagIt and Rosetta SIP. Rosetta SIP was chosen as an output format to allow for the end-to-end workflow testing using the Rosetta system in place at TIB. As a second information package structure BagIt was chosen to allow for a wider adoption of the WorkbenchUI processes. BagIt has a particularly large userbase in its originating country, the United States, where it was developed by the Library of Congress, California Digital Library and Stanford. As such, it remains the main SIP format in many large data repositories such as Stanford Data Repository (SDR)²⁷ or is supported as a common data exchange package, e.g., by Dryad²⁸. Archivematica is an open-source digital preservation system with a growing customer base which supports BagIt for ingest²⁹ as well as for its AIP (Archival Information Package) section³⁰. For details about the history and make-up of the Rosetta SIP as well as BagIt we refer the reader to D2.2.3 section 4.3.2. As described earlier in this deliverable, the WorkbenchUI passes the SIP to a digital preservation system for long-term archival. The following sections describe the processes of the WorkbenchUI SIP generator component as well as the processes covered in the digital preservation system itself.

²⁷For a discussion of BagIt in the context of the Standford Digital Repository implementation, see Tom Cramer's and Katherine Kott's D-Lib article under <http://www.dlib.org/dlib/september10/cramer/09cramer.html>

²⁸See http://wiki.datadryad.org/BagIt_Handshaking for Bagit Handshaking Module in Dryad

²⁹https://wiki.archivematica.org/Bag_ingest

³⁰https://wiki.archivematica.org/AIP_structure

4.5.1 SIP Generator

The SIP Generator supports producers of digital material with its organization and packaging as Submission Information Packages (SIP). The provided *SIP Generator* is a Java tool which is capable of generating both, Rosetta SIP and BagIt files.

Currently two Java implementations of the *SIP Generator* are available. The first one is developed by LTU, the second one is developed by TIB (see Section 1.4 for links to the respective source code). The described input and output characteristics of the data described below is followed by both implementations. Providing multiple implementations of the *SIP Generator* component provides stakeholders with two examples on how to integrate existing SIP generator functionality into the Digital Preservation Service.

Input

During the processes which are covered by the WorkbenchUI metadata is extracted and modified, as well as newly generated, e.g. when creating an IFC reconstruction from an E57 pointcloud file. To work with this data the *SIP Generator* needs it to be provided by the WorkbenchUI in a defined layout, which serves as input for the SIP generation processes. The produced data is categorized into three types:

- master files
- derivative files/copies
- metadata

Master files are the master representations of each *Intellectual Entity*. They are the objectives of digital preservation itself. A master file is the original file which will be kept during every preservation action in the future. From this representation any modified master representations and/or derivative representations are created. Any master file has to be available in an archival format.

Derivative files/copies are the access representations of each *Intellectual Entity*. They are generated based on the corresponding master file for access and usability purposes or

just give new opportunities for further processing. A derivative copy is not the goal of digital preservation, hence, it does not need to be available in an archival format.

Metadata is divided in descriptive and technical metadata. The descriptive *buildM* metadata describes both the Physical Asset – the building – and the Digital Object – the master representation available either as IFC or E57 file – and is partly extracted, partly edited by the stakeholder. Beside, the technical metadata – *ifcm* for IFC and *e57m* for e57 master representations – consists of information needed for a successful and sustainable digital preservation of the digital object itself.

Additionally, the semantically enriched metadata captured as *buildm+* data will be part of the Submission Information Package in M36.

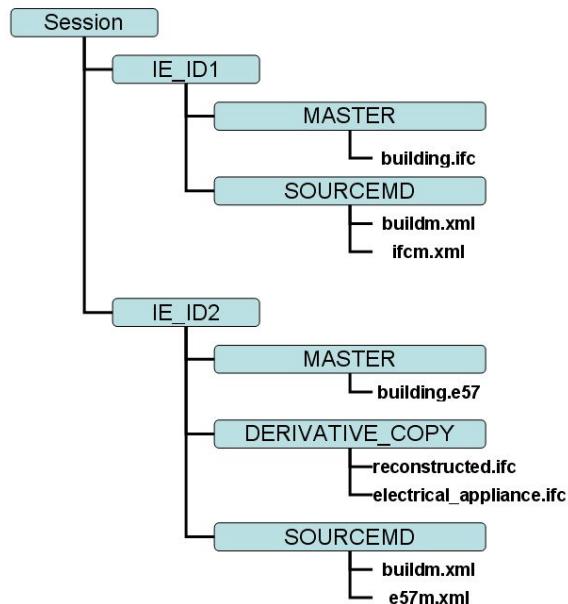


Figure 22: Exemplary overview of the input folder and its components

Figure 22 shows the structure of the input folder as it is provided by the WorkbenchUI for each *session*. A session consists of at least one *Intellectual Entity*, but can also have more than one. Every file corresponding to an *Intellectual Entity* is provided in a distinct folder, named as *IE* for *Intellectual Entity* followed by a consecutive number as (temporary) *ID*.

Every *Intellectual Entity* folder again consists of the folders *MASTER* and *SOURCEMD*. The *MASTER* folder holds the master representation file for the *Intellectual Entity*, an

IFC or an E57 file. Second, the *SOURCEMD* folder holds the descriptive metadata as a *buildM* instance and the technical metadata as *ifcm*, respective *e57m* instance. In the future support for the mentioned *buildM+* will be captured in *SOURCEMD* as well.

If the session contains one or more derivative copies the WorkbenchUI creates the additional folder *DERIVATIVE_COPY* which holds every generated derivative file of the master representation. Derivative files are created by the geometric enrichment components *IFC Reconstruction*, *Rise* and *Point Cloud Compression* (available in M36). See Section 4.4 for a description of those components.

Independent of the output format – either Rosetta SIP or BagIt – this structure is the input format for the *SIP Generator* component.

Rosetta SIP

If the user wants to ingest the provided session files into the Rosetta DLP system a Rosetta SIP file needs to be created. Figure 23 shows the structure of the *SIP Generator*'s output for a Rosetta SIP according to the input structure shown in Figure 22. Every *Intellectual Entity* is treated as a separate SIP which will be ingested into Rosetta. Even though the folder structure on the first level looks the same, the descriptive XML content has been altered by the SIP generation process: Based on the descriptive *buildM* metadata a file **dc.xml** is created for Rosetta related processing prerequisites. The newly created folder **content** holds the file **ie.xml**, together with the folder **streams**. **ie.xml** is a Rosetta compliant METS file containing administrative and structural information, as well as the provided *buildM* and *ifcm*, respectively *e57m* as source metadata. **streams** contains the data files of the *Intellectual Entity*, at least the master representation which is located in the folder **MASTER**. If provided, the derivative representations are located in the folder **DERIVATIVE_COPY**.

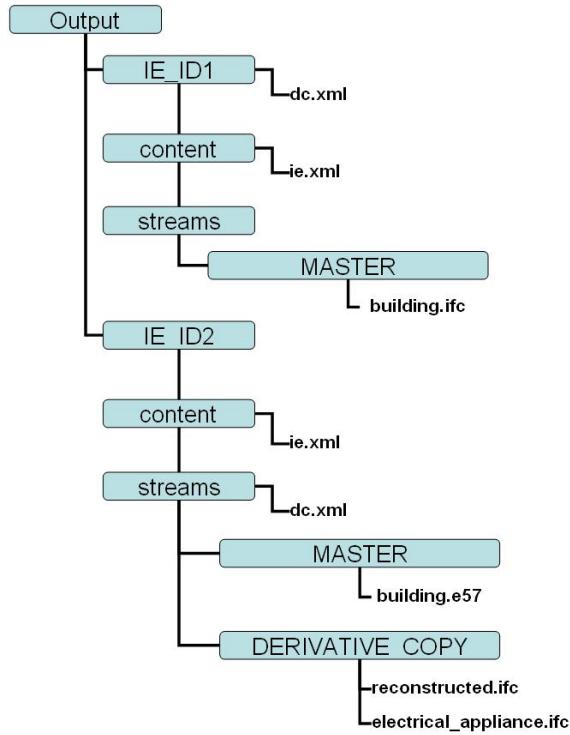


Figure 23: Overview of the *SIP Generator's* output format for Rosetta SIP

BagIt

If the user wants to ingest the provided session files into other digital preservation system, e.g., Archivematica, or exchange it with other institutions the files can be organized as BagIt. Figure 24 shows the structure of the *SIP Generator's* output for a BagIt compliant SIP according to the input structure shown in Figure 22. The BagIt SIP also contains the complete session of the WorkbenchUI. In contrast to the Rosetta SIP the complete session structure will be part of the BagIt specific folder **data**. All containing files are not altered, so the structure for each *Intellectual Entity* stays the same. In the main folder **bag** additional text files are generated by the process. The **bagit.txt** contains information about the current BagIt version and the encoding of the tag files, which has to be “UTF-8“. With the help of the **baginfo.txt** administrative information about the bag and the producing organization respectively tool can be collected. The **manifest-md5.txt** as well as the **tagmanifest-md5.txt** are files for the integrity check of the bag and all associated files. The first one provides md5 checksums for every file which is part of the folder **data**, the second one does the same for every tag file, in this case just the three

other text files.

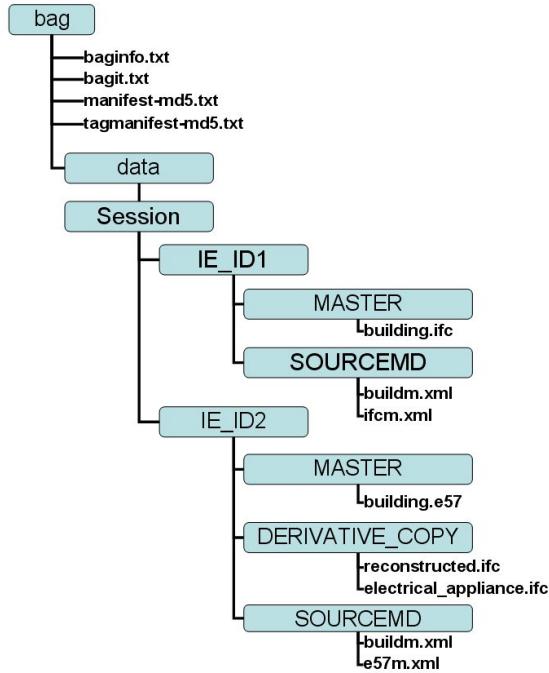


Figure 24: Overview of the BagIt output

4.5.2 ROSETTA DPS

As motivated in deliverable D2.2.3, Rosetta is chosen as a digital preservation system to which the WorkbenchUI acts as a pre-ingest workbench. For detailed information about the suitability of Rosetta as well as the role of the pre-ingest workbench in a digital preservation environment we refer the reader to Section 3.4 in D2.2.3. In Figure 25 an updated view of the WorkbenchUI and Rosetta in an OAIS compliant digital preservation workflow is shown.

Different integration options – for example that of using the WorkbenchUI as a stand-alone pre-ingest workflow in juxtaposition to a modular approach, where single modules such as the E57 metadata extractor are directly integrated into a digital preservation system – have been discussed in D6.2. Here, we refer the reader to Section 6.3 of said deliverable for a detailed discussion. The digital preservation system furthermore provides all entities not natively supported by pre-ingest tools such as the WorkbenchUI, e.g., preservation

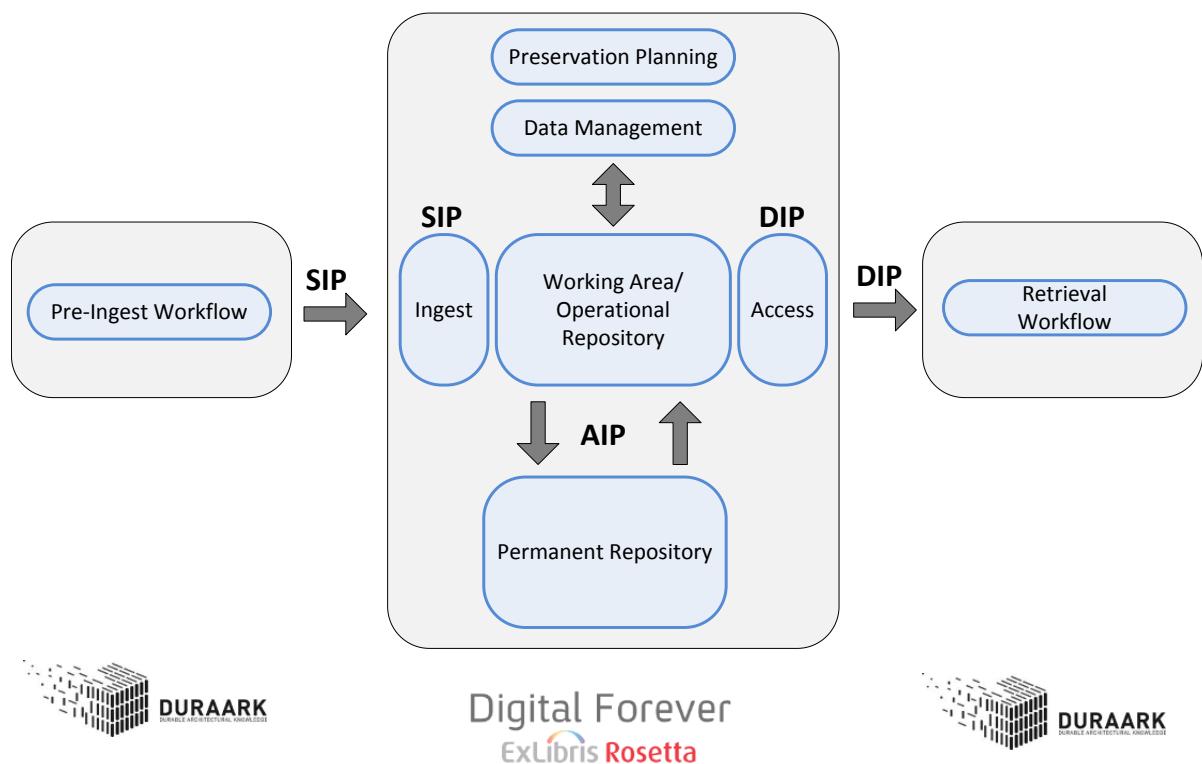


Figure 25: Rosetta and the DURAARK WorkbenchUI in an OAIS compliant digital preservation workflow. The WorkbenchUI's Pre-Ingest Workflow outputs preservation-ready SIPs. For later access and the storage into the SDAS knowledge graph Rosetta passes the DIP (dissemination information package) back to the Workbench UI so that it can be used in the Retrieval Workflow

planning or data management. In the case of the OAIS access entity, DIP (dissemination information package) generation is triggered after the user has conducted a search and retrieval process with the Retrieval Workflow of WorkbenchUI. From the result, the user is then presented with the option to retrieve the object – this call is passed to the digital preservation system, which generates the DIP and passes it back.

As shown in figure 25 the WorkbenchUI and the digital preservation system interact during SIP creation as well as during access to the Dissemination Information Package (DIP). The interaction between WorkbenchUI and the Rosetta web services is managed by the component *Rosetta Connector*. Figure 26 breaks down the interaction between the WorkbenchUI and the Rosetta digital preservation system during the deposit and ingest process. Here, web services offered by the digital preservation system are used to pass the information package to Rosetta, to receive information about the DPS-internal identifier and the status in return and to build the URL needed for the access to the object³¹. This link will be stored in the SDAS knowledge graph. As with M30 the Rosetta Connector is not yet storing this information back to the SDAS, but will be available in M36.

The Rosetta DPS used in DURAARK is hosted at TIB. The API endpoint for the web services used by the *Rosetta Connector* is available at <http://rosetta.develop.lza.tib.eu/dpsws/deposit/DepositWebServices?wsdl>. Note, that the endpoint is only allowed to be accessed by the demo instance of the WorkbenchUI running at <http://workbench.duraark.eu> and selected development hosts.

³¹The web services are provided by the Rosetta software vendor Ex Libris. Further information about the web services is available at <https://developers.exlibrisgroup.com/rosetta/apis/DepositWebServices>

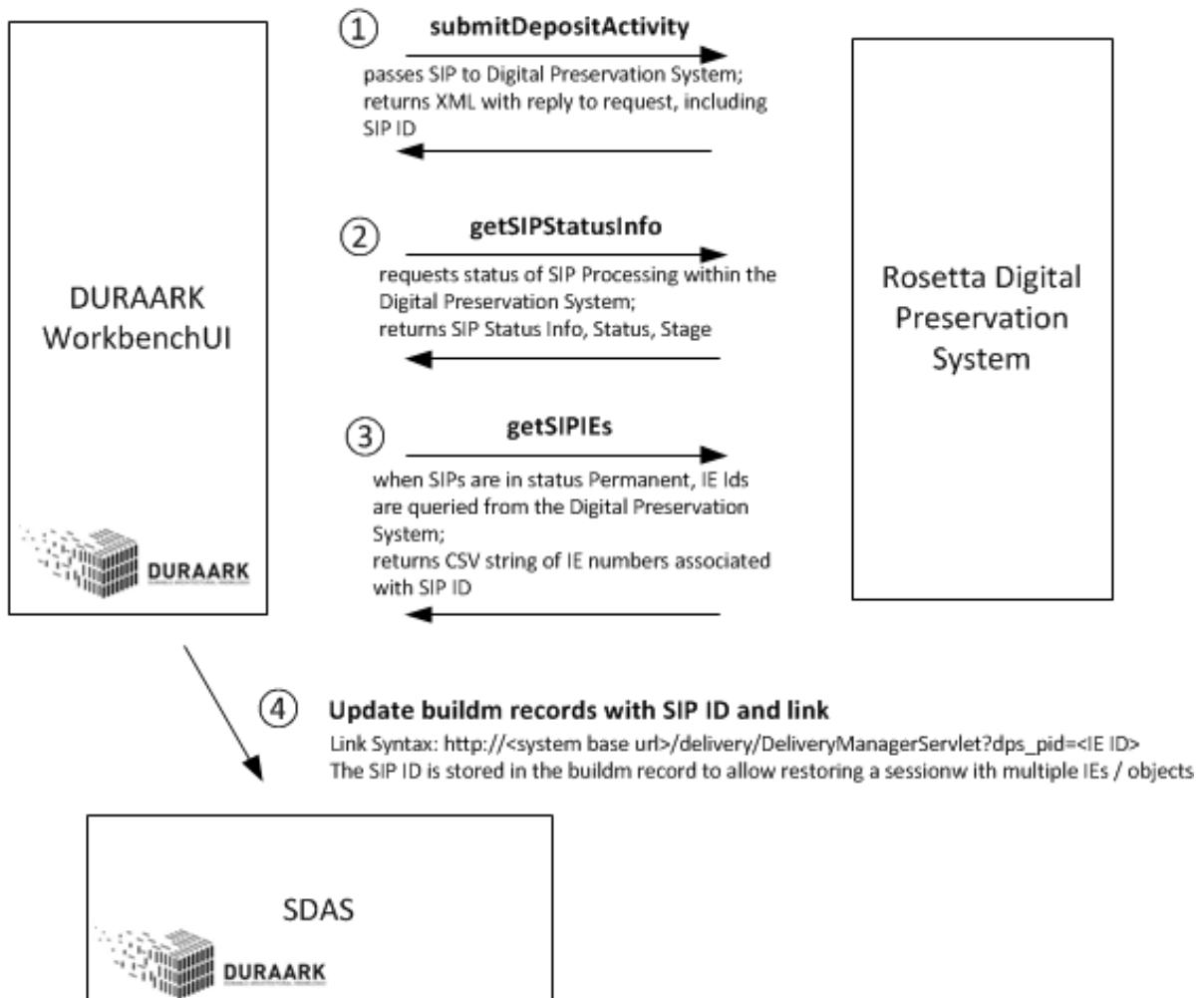


Figure 26: Breakdown of web services used in the communication between the WorkbenchUI and the Rosetta digital preservation system during the deposit and ingest process

5 Decisions & Risks

This section is an update of the respective section in report D2.4. It reassesses the technical decisions and impacts in the current context as well as the risks and provides updates where necessary.

5.1 Technical decisions and impacts

Web-based user interface for the integrated prototype The problem with large file sizes described in D2.4 was mitigated in having a fully distributed service infrastructure. This allows services to be deployed next to the data stores, which allows fast access times and prevents the necessity to upload all files a user wants to work on. The WorkbenchUI as GUI can either be installed next to the data storage, or it is installed on an arbitrary server and connects to the Service Platform remotely. With this setup the services which need file access are still able to access files fast, whereas the communication with the WorkbenchUI only needs to transmit, e.g., meta- or session-data, which is small in size compared to the actual data files.

However, if a user wants to upload local files to the system, the limitation of bandwidth still holds.

NodeJS as runtime environment for web services The experience while building multiple web-services with NodeJS proved its suitability as a server backend language in the context of DURAARK. No problems related to the environment and language itself were encountered until now.

REST API Providing DURAARK's functionality as REST API proved to be advantageous for the development of a distributed, web-based system. The API layer makes it possible to use each service as a standalone service, with no dependency on other services.

JSON as data exchange format JSON as serialization format for transmitting data between the Service Platform and the WorkbenchUI was proving useful, as the native capabilities of Javascript/NodeJS on the server and Javascript in the client's browser make it easy to handle this data format. We use the JSON dialect *JSON-LD* (JSON for Linked Data) now that specifically allows to transmit semantic and linked data as JSON objects, e.g., for metadata encoded in the buildm schema.

5.2 Risk assessment

The risks described in report D2.4 have not materialized. One risk for the period until M36 is added:

Risk Description For the integration plan of the *Difference Detection* component for M36 a fully automated registration between two E57 files or a E57/IFC combination is necessary. A fully automated registration was never part of the description of work (DOW), but UBO is taking this additional effort to allow the WorkbenchUI to utilize the *Difference Detection* component in a natural way, without the need for an additional user interface that takes care of the user-guided registration of two datasets.

Risk Assessment .

Impact Medium

Probability Medium

Description For utilizing the Difference Detection component it is necessary to spatially register the two compared objects with each other. Currently this is done via a graphical tool provided by UBO. UBO is developing a fully automatic registration component towards M36. If the fully automated registry component is not available in M36 the registration has to be done via the graphical tool.

Contingency Solution A graphical tool is provided that allows to perform the registration manually. This is sub-optimal, as this tool does not work within the

WorkbenchUI, but as a standalone tool, making it necessary to download the files to compare to the local computer. Still, it is possible to provide the user with this tool to perform registration. A second solution is to develop a graphical registration tool within the WorkbenchUI. Open source software exists (<http://potree.org/>) that allows to display point clouds in the browser, together with other, mesh-based geometry. This software can be taken as base to develop such a browser-based solution.

6 Software Licenses

The following table gives an updated overview of the software licences generated and used for the DURAARK system. The services are written in Javascript with the server framework “NodeJS“, which is a very common combination in the web development world. The majority of open source projects which are using the same technology stack as DURAARK is preferring the MIT license, as it is a very permissive and community friendly license. Therefore we decided to also use MIT as our main license for the services. The components directly used by the services are licensed either as MIT, BSD, GPL or LPGL, all permissive license, enabling an easy community adoption and extensions for the components.

IPR Type	IP	Software name	License
software	generated	WorkbenchUI	MIT
software	generated	Session Service	MIT
software	generated	Metadata Service	MIT
software	generated	Semantic Digital Archive Service	MIT
software	generated	Geometric Enrichment Service	MIT
software	generated	Digital Preservation Service	MIT
software	used	pyIfcExtract component	LGPL
software	used	E57Extract component	GPL
API	used	SDAS component	CC0 ³²
software	used	Focused Crawling component	LGPL
software	used	IFC Reconstruction component	GPL
software	used	RISE component	BSD
software	used	SIP Generator component (LTU)	BSD
software	used	SIP Generator component (TIB)	BSD
software	used	Rosetta Connector	BSD
software	used	EmberJS framework	MIT
software	used	NodeJS framework	MIT

7 Conclusions & Impact

This report presents the M30 version of the integrated software prototype which provides the integration of all software partner components available at the time of writing³³. In few cases the components are not yet finished in their final version. In the according cases the integrated software prototype provides a stable API for communication with those services, developed together with the respective partner.

The components are summarized into **services**, to group together similar functionality into a coherent unit. Each of those units - Session Management, Metadata, Semantic Enrichment, Geometric Enrichment and Digital Preservation - provides a REST-API to access its functionality in a programming language- and platform-agnostic way. Together the services form the so called **DURAARK Service Platform**, which encapsulates the complete functionality of the developed software prototypes in a coherent layer. A reference implementation for a graphical user interface is provided – the **DURAARK WorkbenchUI**. It utilizes the underlying Service Platform and gives stakeholders a GUI to carry out use cases defined in the project with the help of step-by-step workflows. Additionally, the integration of DURAARK's functionality into a 3rd party software is shown within the AEC domain specific Grasshopper® software, which shows the integration capabilities of the Service Platform. The WorkbenchUI and the Grasshopper®components already provide a sound demonstration system for the features and advantages of the DURAARK system.

This report streamlines prior work in D2.4 regarding the definition of stakeholder workflows. The definition of three main workflows (pre-ingest, retrieval, maintenance) in the WorkbenchUI and the workflows within the Grasshopper® software are a big step forward in the usability of the DURAARK system as it presents the user with easy-to-use tools for performing the set of defined use cases. An evaluation of the usability of the DURAARK system will be done in the remainder of the project, as part of D7.4 in M36.

Regarding the extensibility, maintainability and sustainability of the project a sound development environment is provided for existing and future developers that allows to use the current prototype either as-is, or to extend it with custom functionality.

³³The *Difference Detection* and the *Pointcloud Compression* components will be available in the Service Platform in M36. The *Difference Detection* is already available in the Grasshopper®components.

Together with a robust deployment infrastructure DURAARK provides stakeholders with a straightforward way to test our approach within their own company, and to integrate the functionality in an unobtrusive way. This is a necessary pre-requisite for an adoption of the system in the AEC community.

For interested stakeholders we provide a public WorkbenchUI installation at <http://workbench.duraark.eu>. This is one point of the sustainability program in DURAARK for providing access to the system also after the end of the project. Additional sustainability actions are listed in D8.5, Section 5.

8 Future Work

From the view of the integrated software prototype the remainder of the project until M36 will be dedicated to the following points (also including the points described in Section 1.3). The list is sorted into the respective workflows:

Pre-ingest Workflow

- The remaining components in WP5 – *Difference Detection* and *Pointcloud Compression* – will be integrated into the Service Platform and the WorkbenchUI. The integration depends on the possibility of registering IFC and E57 pointcloud files in a fully automatic way, which is currently developed in WP5 (see Section 5.2 for a risk assessment). With the addition of those two components the DURAARK system will be feature-complete with regards to component integration.
- The RISE component will be enhanced to be usable with arbitrary pointcloud scans. Currently we support the “Nygade“ data set as a showcase in the WorkbenchUI.
- The pointcloud schema extension for the IFC model “HDF5“ (see D3.5) will be added as the third officially supported file format for the DURAARK system, next to IFC and E57.
- The extraction of “building element levels“ information containing properties inspired by the geometric information vocabulary of the IFC standard, e.g., number of rooms, walls or building area, is currently under development. The information will be extracted from pointclouds and will be added to the SDAS knowledge graph. This is a crucial feature of the DURAARK system, as it combines information from the geometric enrichment and the semantic enrichment to allow stakeholders to perform powerful search queries with the system.

Retrieval Workflow

- The Semantic Digital Archive Service will allow to upload and store enrichment topics. This feature allows operators of the DURAARK system to define and maintain information topics which are relevant to their use cases or daily work. The uploaded enrichment topics will then be available in the WorkbenchUI for end-user stakeholders.
- The WorkbenchUI will provide an enhanced interface for leveraging the SDAS knowledge graph via search queries.

Maintenance Workflow

- The missing parts of the maintenance workflow functionality will be added until M36, which will also determine the best strategy on how to store the semantic enrichment information gathered in the pre-ingest workflow (buildM+ data) into the Rosetta system, as well as when to backup the SDAS knowledge graph into Rosetta.

General

- Similar to the proof-of-concept integration of the DURAARK system into Grasshopper® we are currently developing an integration into CATENDA's *bimsync*³⁴ software. *bimsync* is a cloud service for secure storage and distribution of BIM files, including revision control and an intuitive 3D visualization in the browser. It provides an API for interacting with the software. This showcase will demonstrate how to combine the Service Platform API with a 3rd party API. The integration will focus on enabling *bimsync* users to use the pre-ingest workflow for storing IFC files into the DURAARK system. CATENDA as creator of the *bimsync* software is leading this activity, which will be finished in M36.
- The DURAARK system is being evaluated in WP7 (upcoming D7.4). As part of this evaluation the WorkbenchUI will be enhanced regarding the user experience

³⁴<https://bimsync.com/>

for the pre-ingest and the retrieval workflow. We are planning to focus especially on the user interface for the retrieval workflow, to fully leverage the capabilities of the SDAS knowledge graph within the WorkbenchUI. Additionally, sorting and selection mechanisms will be provided to manage the candidates provided by the semantic enrichment process.

The listed points will be carried out in the months towards M36. Their implementation is organized in our release schedule, which is described in Section 2.4. The complete feature-set available in the DURAARK system will be described in D7.4. As the evaluation deliverable it will also assess the usability of those features then.

Appendices

1 Software Design Finalization

This report concludes the design description of the integrated software prototype that started with in report D2.4, Section 3, introducing basic design principles, the overall architecture, as well as the motivation for using a distributed, web-based software architecture for the DURAARK system. An updated description of the overall architecture of the DURAARK system, namely the WorkbenchUI as a graphical user interface and the Service Platform as the micro-service based functionality provider for users of the DURAARK system (either via the WorkbenchUI or via the REST-API) is given in report D8.5, Section 2 (Exploitable outcome). For this reason there will be no repetition of those concepts here in this final design description. This section adds the deeper technical information which are not yet part of other deliverables and also explains changes to the original design that was described in D2.4.

The biggest change to D2.4 is the introduction of the DURAARK Service Platform, which groups together a set of five integral functional blocks that are forming the core of the DURAARK functionality:

- Session Service,
- Metadata Service,
- Semantic Enrichment Service,
- Geometric Enrichment Service and
- Digital Preservation Service.

The Service Platform is accessible via a REST API that delivers information in two serializations: JSON and JSON-LD. The JSON format is a versatile and de-facto standard for exchanging information between web services and consumers, which makes it a natural choice for the communication layer of the Service Platform. Since the initial design description in D2.4 we additionally added the JSON-LD serialization format to the system. JSON-LD³⁵ reads as “JSON for Linking Data“ and is a popular mechanism to transport

³⁵<http://json-ld.org/>

linked, semantic data from a service to a consumer or in-between services. JSON-LD is building upon the principles of the Resource Description Framework (RDF)³⁶ and is capable of representing the same information as any other RDF serialization. The advantage of JSON-LD is that taking JSON as the object model for encoding semantic data is that JSON is understood in web-applications and services in a natural way.

In DURAARK JSON-LD is used for providing metadata information encoded in our DURAARK metadata schemas buildm and buildm+ from the Metadata Service to the WorkbenchUI, where it is edited. The same goes for the Semantic Enrichment Service, which is using JSON-LD for communication with the GUI. Finally, the Digital Preservation Service is taking the metadata as JSON-LD to serialize it into our supported SIP formats: Rosetta SIP and BagIt. The remaining communication channels between the Service Platform and the WorkbenchUI are JSON encoded, e.g., session information or the communication with the Geometric Enrichment Service. Using those two natural formats for their respective tasks is diminishing the entry barrier for new developers in using the system, as they are well adopted and understood in the web community and the tooling support is sound.

Putting together the pieces leads to the final design overview, as depicted in Figure 1.

³⁶<http://www.w3.org/1999/.status/PR-rdf-syntax-19990105/status>

2 Development Quality Assurance

This technical section describes the quality assurance mechanisms accompanying the software development cycle of the project. The section is a guideline for software developers to give them a single source of information on how to start developing for the DURAARK project.

2.1 Source Code and Documentation

Source code management is done on Github . Github is the de-facto standard service for developing open source projects and provides collaborative tooling for revision control via the **git**³⁷ source code management tool, issue tracking and documentation. Because of the popularity of the platform a huge amount of third-party projects are providing additional tools that integrate with Github directly (see an example in Section 2.2). For open source projects Github and the majority of additional tools is freely available.

Documentation for the DURAARK services and for the individual components is also directly provided via the Github platform. This is common and adopted especially in the web development community, where Github forms a central entry point to a project, with source code and documentation in one place.

DURAARK has its own namespace on Github: <http://github.com/DURAARK>. This namespace hosts all software projects in a single spot. The overall project for the DURAARK system is located at <http://github.com/DURAARK/duraark-system>. It contains the documentation on how to deploy the system and relates to the code repositories of each service. The service repositories again link to the respective components. The deployment mechanism is also described in Section 2.3.

The documentation on Github for the DURAARK services contain an overall description of the project, installation instructions, the testing setup (see Section 2.2), a link to the API documentation and a link to the demo servers (e.g., see <http://github.com/DURAARK/duraark-metadata>).

³⁷<https://git-scm.com/>

2.2 Continuous Integration

Continuous integration is a mechanism to check the functionality of a project as part of an automated development process. This is accomplished via a test suite for code units. For DURAARK we implement a test suite that checks the correctness of the service APIs. Services contain the core functionality of the system and their official and supported way for interaction is the respective REST API. This allows us to test each resource and method of the API to guarantee that the exposed functionality works as expected from consumers of the API (e.g., the WorkbenchUI or an external developer accessing the service via REST).

The continuous integration service *CircleCI*³⁸ is a freely available service for open source projects to test source code. It natively integrates with Github. On each code change in the repository the test suite is run with the new code base. A message is sent to the developer in case the new code is causing problems with formerly passing tests.

To check the status of a project open the web page https://circleci.com/gh/DURAARK/REPOSITORY_NAME. REPOSITORY_NAME corresponds to the name of the Github project, e.g., <https://circleci.com/gh/DURAARK/workbench-ui>.

2.3 Deployment

The DURAARK system is heavily based on the idea of distributed **services**. The emerging and highly adopted technology *Docker*³⁹ is an open platform for the packaging of self-contained services, which are executed in a light-weight host environment on the three major desktop platforms Microsoft Windows, Linux and Macintosh. We use docker to package releases of the DURAARK system and publish them on the public **Docker Hub**⁴⁰ platform. There stakeholders can download each DURAARK package and deploy it via a single command line instruction. To get a list of publicly available DURAARK services and the GUI see <https://registry.hub.docker.com/search?q=duraark>.

³⁸<https://circleci.com>

³⁹<https://www.docker.com/>

⁴⁰<https://registry.hub.docker.com/>

For the orchestration of all services and the WorkbenchUI we are using the system **nscale**⁴¹. It makes deployment and management of distributed software systems easy and builds upon docker and Docker Hub. nscale targets service oriented infrastructures and is a sound match for the needs of DURAARK. The Github repository [http://github.com/DURAARK/duraark-system](https://github.com/DURAARK/duraark-system) contains the necessary deployment instructions to get DURAARK up-and-running via the nscale system. This repository is also the recommended starting point for developers to extend the system.

Having a robust and easy to use deployment setup for a distributed system we try to lower the entry hurdle for working with the DURAARK system either as a stakeholder or as a developer.

2.4 Release Planning

Release planning is mainly managed within the WorkbenchUI repository as new features are started as a WorkbenchUI mockup and are then propagated to the respective services and components. Github's *issue* system is used to keep track of open implementation tasks (e.g., the points listed in Section ?? are collected as open issues). Open issues are sorted into *milestones*, which are *released* after all issues have been closed. Together with a release of the WorkbenchUI the services and components which were changed or extended are also released to ensure a consistent versioning of the software. The WorkbenchUI's issue list and according milestone/release planning can be found at the following URL:

- Release Planning:
<https://github.com/DURAARK/workbench-ui/issues>

⁴¹<http://nscale.nearform.com/>

The release history of each service is available at the following repository URLs on Github:

- WorkbenchUI (main repository):

<https://github.com/DURAARK/workbench-ui/releases>

- Session Service:

<https://github.com/DURAARK/microservice-session/releases>

- Metadata Service:

<https://github.com/DURAARK/microservice-metadata-extraction/releases>

- Semantic Digital Archive Service:

<https://github.com/DURAARK/microservice-sda/releases>

- Geometric Enrichment Service:

<https://github.com/DURAARK/microservice-geometric-enrichment/releases>

- Digital Preservation Service:

<https://github.com/DURAARK/microservice-digital-preservation/releases>