ENergy aware BIM Cloud Platform in a COst-effective Building REnovation Context

Project Number 820434

D2.8
Full Knowledge Extraction and Object Enhancement Service (KEES) Prototype

Version 1.0

10/2021

Final

Public Distribution

ATB

Project Partners:
ATB Bremen, TALOS RTD, Universitá delle Marche, Universidad de la Laguna, University of Zagreb, Laurentia, SPA, SmartGateways, ETH Zurich, Junta de Extremadura, CNR-ISTI

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No *820434*

PROJECT PARTNER CONTACT INFORMATION

ATB Sebastian Scholze Wiener Straße 1 28359 Bremen Germany Tel: +49 421 22092 0 E-mail: scholze@atb-bremen.de	**TALOS** Melinda Kuthy Diogenous 1, BLOCK A EGOMI 2404 LEFKOSIA Cyprus Tel: +35 357 224 543 33 E-mail: mk@talos-rtd.com
Universitá delle Marche Andrea Bonci Piazza Roma, 22 60121 Ancona Italy Tel: +39 07122021 E-mail: a.bonci@univpm.it	**Universidad de la Laguna** Norena Martín Dorta Molinos de Agua S/N 38071 La Laguna (Tenerife) Spain Tel: +34 922319545 E-mail: nmartin@ull.edu.es
University of Zagreb Stjepan Bogdan Unska, 3 10000 Zagreb Croatia Tel: +385-1-6129728 E-mail: stjepan.bogdan@fer.hr	**Laurentia** Sergio Muñoz Plaza Honduras 9, 1 46022, VALENCIA Spain Tel: +34 666 708 965 E-mail: sergio.munoz@laurentia.es
SPA Terrence Fernando RUA 24 DE FEVEREIRO LOTE 9, 5000 410, VILA REAL Portugal Tel: +35 193 408 2429 E-mail: t.fernando@salford.ac.uk	**SmartGateways** Dulcidio Coelho 31 SACKVILLE STREET M1 3LZ, MANCHESTER United Kingdom Tel: +44 779 107 2325 E-mail: dfcoelho@smartgateways.co.uk
ETH Zurich Ajad Chhatkuli Raemistrasse 101 8092 ZUERICH Switzerland Tel: +41 4463 45350 E-mail: ajad.chhatkuli@vision.ee.ethz.ch	**Junta de Extremadura** Sagrario Conejero AVENIDA DE LAS AMÉRICAS 2 06800 Mérida Spain Tel: +34 92 400 41 00 E-mail: sagrarioconejero@proyectoedea.com
CNR-ISTI Paolo Cignoni PIAZZALE ALDO MORO 7 00185 Roma Italy Tel: +39 6499 33038 E-mail: p.cignoni@isti.cnr.it	

EXECUTIVE SUMMARY

The ENCORE project aims to increase the share of renovated stock in Europe and worldwide by providing effective and affordable BIM tools that cover the whole renovation life-cycle. The tools shall support all the actors in the renovation process, facilitating cost-effective renovation projects, achieving higher energy efficiency and comfort levels.

This document describes the full prototype of the Knowledge Extraction and Object Enhancement Service (KEES), including the ENCORE Ontology, as part of the ENCORE solution. The KEES module processes data coming from connected systems / services / sensors (data producers) to extract knowledge and use it for enhancing objects in a BIM model by annotating them. Further, it describes the full prototype of the ENCORE ontology, which is based on the BOT and ifcOWL ontologies, enhanced by ENCORE-specific classes.

After describing the KEES module itself, the integration with other ENCORE services as well as KEES usage scenarios are presented. Finally, the deliverable gives an overview about the software tools and frameworks used for the implementation.

The document is divided into the following sections:

- Section 1: describes the purpose of this document and provides a brief overview of the contents of the document.
- Section 2: gives an overview of the full prototype implementation of KEES. Furthermore, this section describes the ENCORE ontology.
- Section 3: describes the integration with other ENCORE services, such as the ENCORE Portal and BMS.
- Section 4: presents a brief user guide on how to configure and run the KEES, including the background technologies used for implementation of the KEES.
- Section 6: gives a summary of the document contents.

INDEX OF CONTENTS

INDEX OF FIGURES

INDEX OF TABLES

ABBREVIATIONS

3D	3-dimensional
API	Application Programming Interface
AR/MR	Augmented Reality/Mixed Reality
AWOPS	Automated Work Planning Service
BIM	Building Information Model
BMS	Building Monitoring and Diagnostics Services
BOT	Building Topology Ontology
BRM	BIM Resources Repository Management Service
CO	Confidential
D	Deliverable
EP	Early Prototype
EU	European Union
GIS	Geographical Information System
H2020	Horizon 2020
HTML	Hypertext Markup Language
HVAC	Heating, Ventilation, Air Conditioning
IDE	Integrated Development Environment
IFC	Industry Foundation Classes
IoT	Internet of Things
IPRS	Image Processing and Reconstruction Service
ISO	International Organization for Standardization
JPA	Java Persistence API
JSON	JavaScript Object Notation
KEES	Knowledge Extraction and Object Enhancement Service
OWL	Web Ontology Language
POJO	Plain Old Java Object
RDF	Resource Description Framework
REST	Representational State Transfer
SOAP	Simple Object Access Protocol
T	Task
WP	Work Package
XML	Extensible Markup Language
XMP	Extensible Metadata Platform

1. INTRODUCTION

1.1 DOCUMENT PURPOSE

This document describes the full prototype of the Knowledge Extraction and Object Enhancement Service (KEES), including the ENCORE Ontology, as part of the ENCORE solution. The KEES module processes data coming from connected systems / services / sensors (data producers) to extract knowledge and use it for enhancing objects in a BIM model by annotating them. Further, it describes the full prototype of the ENCORE ontology, which is based on the BOT and ifcOWL ontologies, enhanced by ENCORE-specific classes.

After describing the KEES module itself, the integration with other ENCORE services as well as KEES usage scenarios are presented. Finally, the deliverable gives an overview about the software tools and frameworks used for the implementation.

1.2 DOCUMENT STRUCTURE

The document is divided into the following sections:

- Section 1: describes the purpose of this document and provides a brief overview of the contents of the document.
- Section 2: gives an overview of the specifications and full prototype implementation of KEES. Furthermore, this section describes the full prototype of the ENCORE ontology.
- Section 3: describes the Dockerisation and integration of KEES with other ENCORE services, such as the ENCORE Portal, the BMS and AWOPS.
- Section 4: describes background technologies used for the implementation of the KEES early prototype.
- Section 6: presents a summary of the document.

2. KEES FULL PROTOTYPE DESCRIPTION

2.1 KEES ARCHITECTURE

The KEES is a backend service, part of the BIM-based Support Tools. In the ENCORE architecture it is located in the Engine level as shown in Figure 1.

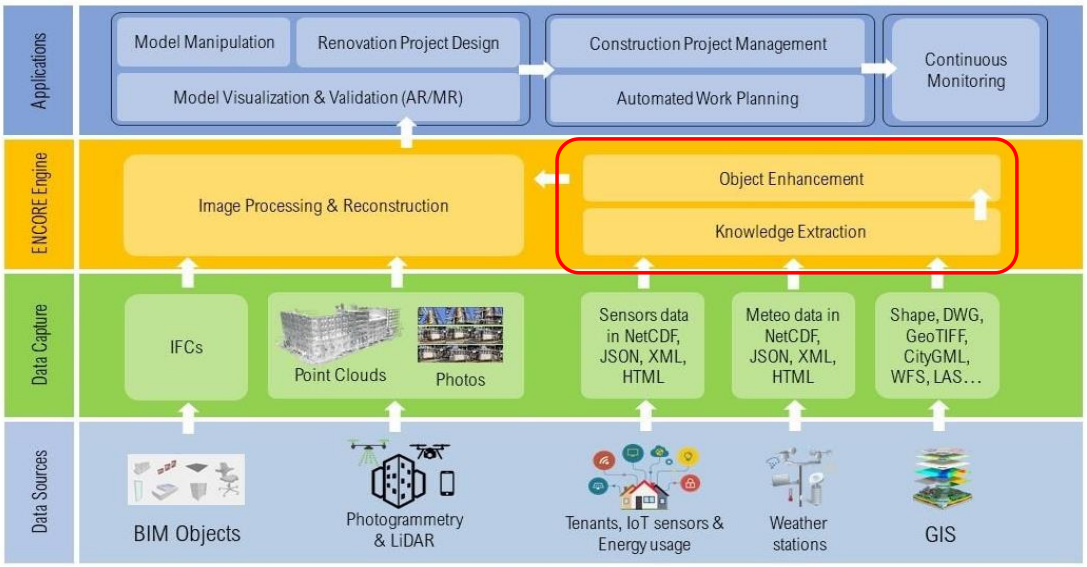

Figure 1: KEES marked in the ENCORE layer architecture

It allows for extracting knowledge from data, such as IFC models, BMS sensor information, which is then used to support the enhancement of identified objects that will take part in a renovation process. Figure 2 shows the conceptual architecture of the KEES as it was shaped for the KEES FP. The main data sources for KEES are other ENCORE services, as indicated on the left side of the figure.

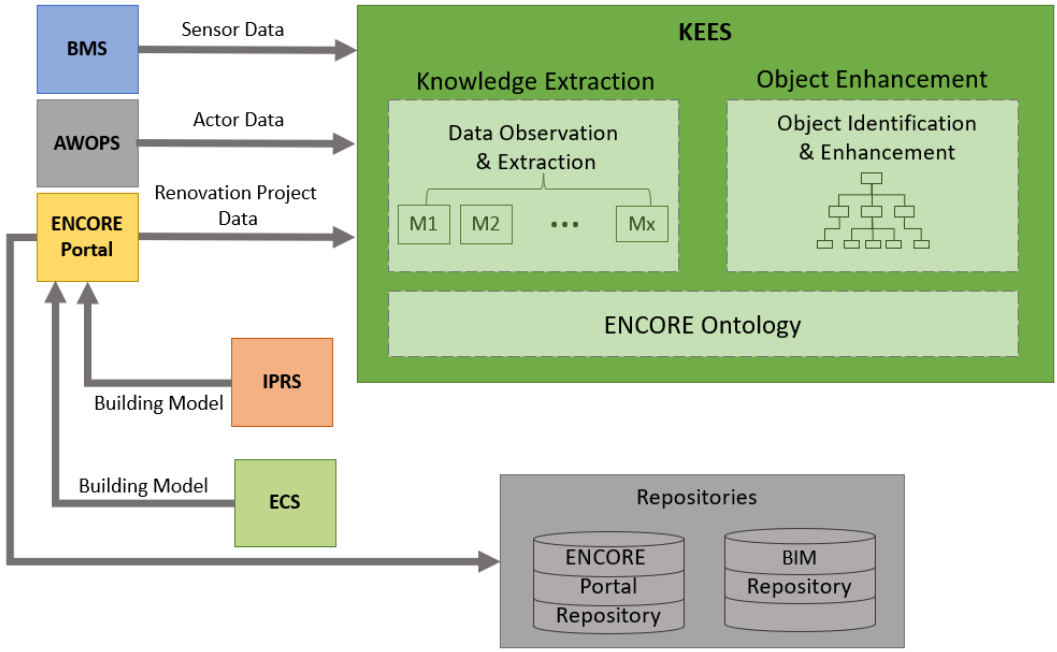

Figure 2: KEES FP Conceptual Architecture

For the full prototype of KEES, the IPRS and the ECS through the ENCORE portal, as well as the BMS are considered as data sources. Due to the communication technologies used, other services can be connected to provide data to the KEES. The repositories relevant for data storage by KEES are indicated at the bottom of the figure. KEES itself comprises three main building blocks, which are presented in detail in the following sub-sections: Knowledge Extraction, Object Enhancement and ENCORE Ontology.

2.2 KEES MODULES

2.2.1 Knowledge Extraction

The objective of the Knowledge Extraction service is to provide annotated data out of the building renovation data it receives from different sources, namely other ENCORE services or systems. Those data include building geometry, building internal structure, sensory information, as well as details on the renovations that are planned to be done.

The Knowledge-Extraction-process correlates such data from distinct systems (e.g. map data from file systems and web-services) and serves as basis data source for object enhancement. The process is divided into several steps performed by the following sub-modules as shown in Figure 3:

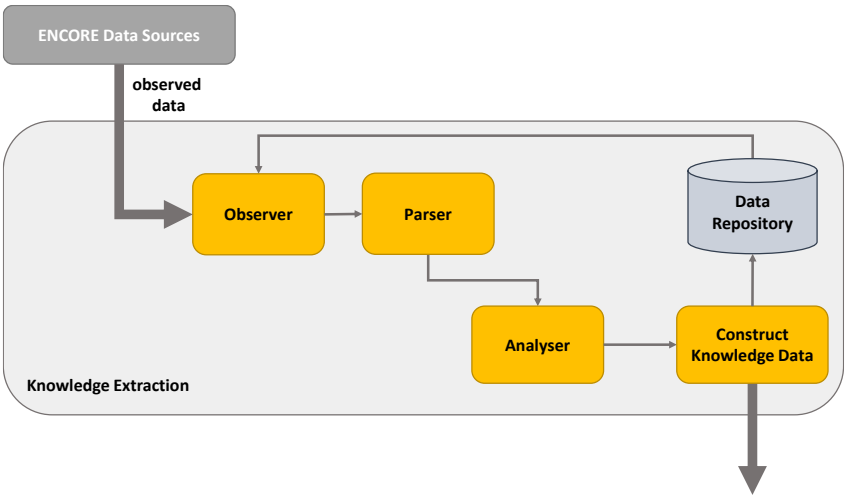

Figure 3: Knowledge Extraction Process

- *Observer* module, which contains all services to observe data sources. The observer module can be further extended and configured for different systems and services serving as data sources.
- *Parser* module, which contains content parsers for the different types of data captured by the observer module.
- *Analyser* module, which correlates the extracted knowledge and constructs the extracted knowledge to be stored and handed over to the Object Enhancement service.

For each data-source (from which knowledge shall be extracted) an index can be specified that holds the data of the resources observed. Depending on the actual resources to be parsed and analysed (e.g. IFC files or sensory data from REST APIs etc.), several resource-specific plugins can be used in the generic knowledge extraction framework. Each of these allows inclusion of parsers and analysers that are specified for additional knowledge sources.

2.2.1.1 Full Prototype Implementation

The full prototype of the Knowledge Extraction module is extended by the analyser to observe rest services, analyse and extract the necessary data which are forwarded to the Object Enhancement module. More specifically, the ENCORE REST Monitor observes the following rest services:

- ENCORE Portal rest services: those services provide with renovation project metadata, such as project ID, project owner and project geometry information (including IFC files), provide direct access to the project IFC files, as well as allows for upload of IFC files.
  The documentation of the services can be found here: https://demo.encorebim.eu/docs/swagger-ui/index.html?configUrl=/docs/api-docs/swagger-config

- BMS rest services: the services provide with monitoring information acquired from sensors installed in the respective building. Such sensor data include metadata to location the sensors in the building elements, sensor value datatypes and units. The documentation of the services can be found here: https://spacloud.eu/public/9786582aaf93e/assets/api_rest_docs.html
- AWOPS rest services: the services will provide with information on the crew types involved in the selected renovation actions.

2.2.2 Object Enhancement

The objective of the Object Enhancement is to identify high-level knowledge from the data observed by the Knowledge Extraction service (services and sensor raw data) and use this information for further object/knowledge enhancement. The service is based on a semantic model (the ENCORE Ontology) for an integrated representation of knowledge.

As shown in the following Figure 4, the process identifies knowledge (i.e. the object to be enhanced) from the observed data (Object Identification) and provides the refined identified knowledge object to further services (Object Provisioning).

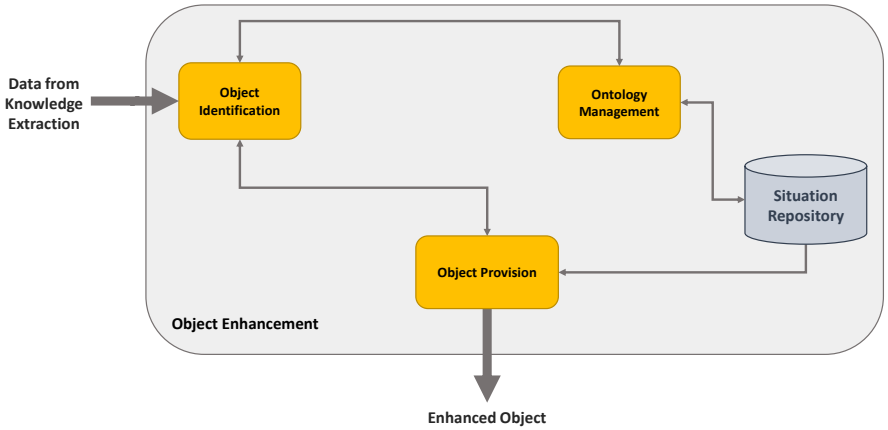

Figure 4: Object Enhancement Process

The Object Enhancement service is based on the services developed within the EU H2020 project SAFIRE, which were extended and adjusted to the needs of the ENCORE services and usage scenarios. The input data is collected from different sources such as IFC, Building Monitoring System, etc. The Object Enhancement Services analyse structured, as well as unstructured data, observed by the Knowledge Enhancement module, to determine the knowledge, and to identify what activity the objects are currently involved in. The knowledge is then stored in the Data Repository as annotation to the content that is used in the observed knowledge.

To achieve this, the Object Enhancement service comprises the following features:

- *Object Identification* module, which analyses the knowledge handed over by the Knowledge Extraction service, and identifies the knowledge, such as information about the involved IFC objects, or information on the environmental characteristics (e.g. temperature, humidity) related to the building to be renovated.
- *Object Provisioning* module, which compares the similarity between the current knowledge and historical knowledge in the repository, and provides the results to other modules.

2.2.2.1 Fully Prototype Implementation

The Full Prototype of the Object Enhancement service is able to enhance objects within an IFC by adding semantic tags to the objects (e.g. windows) that are involved in a renovation process.

The object enhancement retrieves the data snippets that the knowledge extraction extracted from its data sources (e.g. BMS, AWOPS) along with the current IFC of a renovation project.

The object enhancement uses the monitored data and tries to identify the matching objects withing the IFC. If a match between monitored data and IFC object is identified, the IFC is updated with the information from the monitored data.

The enhance IFC will be uploaded to the ENCORE portal after the enhancement process is finished using the ENCORE Portal REST API.
The documentation of the services can be found here: https://demo.encorebim.eu/docs/swagger-ui/index.html?configUrl=/docs/api-docs/swagger-config

2.3 KEES ONTOLOGY

In the ENCORE project and in KEES in particular, the ENCORE Ontology is used to configure the functionality of knowledge extraction and object enhancement services. Additionally, the ENCORE ontology is the baseline for the data model which the KEES uses. By applying an ontological approach, the result is not only reusable, but also understandable by non-experts and can facilitate future adjustments of the KEES to different application scenarios.

Although there are several ontologies that address the aspects of construction and energy management, for the ENCORE solution, the ifcOWL[1] ontology was examined primarily to shape the ENCORE ontology (Figure 5), as it is a well-known representation of the ISO standardised Industry Foundation Classes (IFC). The ifcOWL ontology is able to describe a construction, its components and how those interact, as well as several related parameters, as for example, hardware systems used, energy and occupation aspects.

[1] https://technical.buildingsmart.org/standards/ifc/ifc-formats/ifcowl/

Together with the ifcOWL, the BOT[2] ontology was examined for describing the topology of a construction. The BOT ontology is simple and focuses on the relations between the core elements of a construction, describing its topology. Additionally, the BOT ontology offers an alignment module[3] to connect its entities with those of ifcOWL. Using both ontologies, as well as their alignment module, as basis for the ENCORE ontology served as a means to identify the elements, or parameters, that are necessary to monitor for a complete renovation process, covering different aspects such as building envelope, operation systems, actors involved, etc. At the same time, this allowed a separated focus on the building construction itself, which may be used on demand in cases of construction alteration scenarios.

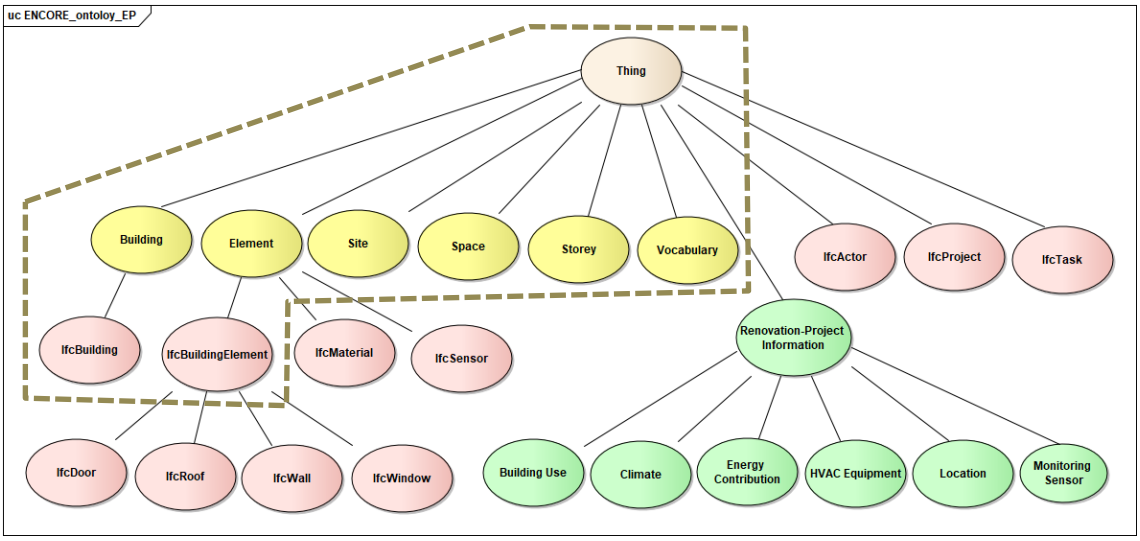

Figure 5: ENCORE ontology for the FP of KEES

(with yellow colour the BOT entities, with red the IFC entities, with green those selected for the ENCORE solution, those grouped with the dashed border are part of the BOT-ifcOWL alignment)

Classes

The descriptions of the ENCORE specific classes follow in the tables below.

Name	Renovation-Project Information
Description	The renovation-project information comprises the parameters necessary to evaluate for cost-effective building-renovation activities taking into account energy and comfort aspects, and are selected based on the scope of the ENCORE project.
Has Subclass	Building Use, Climate, Energy Contribution, HVAC, Location, Monitoring Sensor
Subclass Of	Thing

[2] https://w3c-lbd-cg.github.io/bot/
[3] https://raw.githubusercontent.com/w3c-lbd-cg/bot/master/IFCOWL4_ADD2Alignment.ttl

Name	Building Use
Description	Information that describes aspects on how the building is being used, as for example how many occupants it hosts or the lighting habits of the residents, etc.
Has Subclass	-
Subclass Of	Renovation-Project Information

Name	Climate
Description	Information that describes the weather conditions in the location of the building or site to be renovated.
Has Subclass	-
Subclass Of	Renovation-Project Information

Name	Energy Contribution
Description	Energy contribution describes the parameters or systems that have an impact on the energy demand of the site.
Has Subclass	-
Subclass Of	Renovation-Project Information

Name	HVAC Equipment
Description	Heating, ventilation and air conditioning equipment describes the systems used for adjusting the respective conditions (heat, air flow, etc.) prevailing in the site of renovation.
Has Subclass	-
Subclass Of	Renovation-Project Information

Name	Location
Description	The description of the particular position on the earth of the site to be renovated.
Has Subclass	-
Subclass Of	Renovation-Project Information

Name	Monitoring Sensor
Description	Information that describes a sensor that is attached to a building element, is available to be monitored by the BMS and will be used by the KEES to enrich the building information model (ifc file).
Has Subclass	-
Subclass Of	Renovation-Project Information

The detailed descriptions for the BOT entities can be found in the BOT documentation[4] and the respective definitions for the ifcOWL classes can be found in the IFC documentation[5].

Relations

In order to use the ENCORE ontology for the KEES processing in accordance to the selected usage scenario, relations between the ontology classes have been selected. Those can be enriched to correlate further entities according to the selected ENCORE use cases.

The following table shows the selected relations, as well as examples of their domains and ranges.

Domain (e.g.)	Relation	Range (e.g.)
Building	**has**	**Wall**
Monitoring Sensor	**isAttachedTo**	**Window**
Room	**uses**	**HVAC Equipment**

The selected relations are used by KEES to recognise sub-components of the building elements (“has”), sensors that are related to the identified building components (“isAttachedTo”), as well as energy systems used in the selected areas of the building (“uses”). Using further relations, as for example “describes” (Figure 6), the ENCORE ontology could be connected with further ifcOWL classes, to allow KEES to identify more detailed information or to adapt to different application scenarios.

[4] https://w3c-lbd-cg.github.io/bot/
[5] https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/

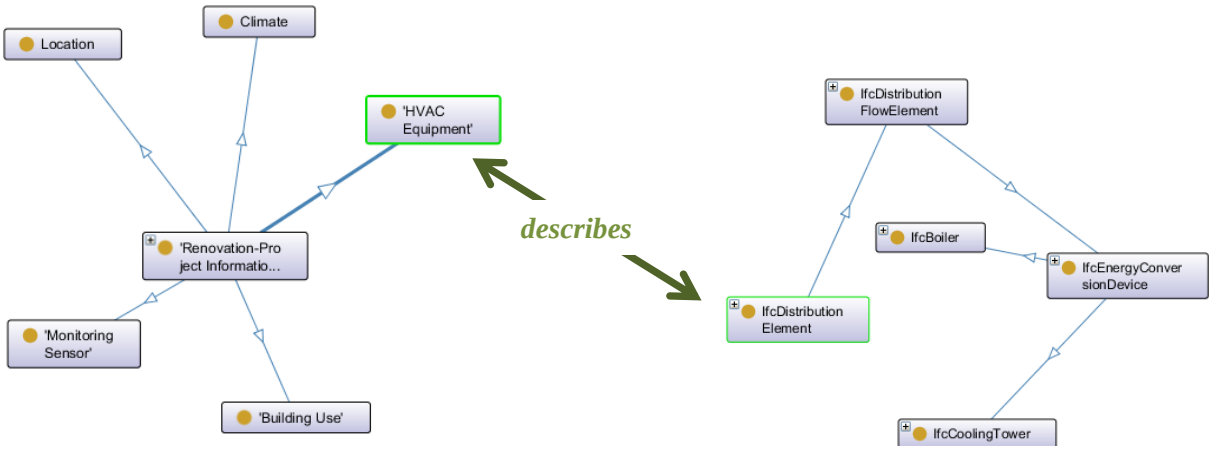

Figure 6: Potential connection between ENCORE ontology and ifcOWL ontology

3. INTEGRATION OF KEES IN THE ENCORE SOLUTION

As shown in the aforementioned Figure 1, KEES is part of the ENCORE Engines working in the backend without having a dedicated graphical interface to interact directly with the user. For acquiring the necessary data for its functionality, KEES, communicates with the ENCORE Portal, the BMS module and AWOPS. As KEES aims to enhance the IFC models of the respective renovation projects, it needs the project information (e.g. building specification, renovations to-be-performed, etc.) which are available in the ENCORE Portal, detailed information about the sensors which are installed to monitor the respective renovation project, available in the BMS, as well as information about the personnel which is involved in the renovation actions, available in the AWOPS module.

For this integration of KEES and communication with the aforementioned modules, Representational State Transfer (REST) libraries provided by the modules are used. In more detail, the interaction of KEES with the ENCORE Portal, the BMS and AWOPS is described by the following workflow in Figure 7.

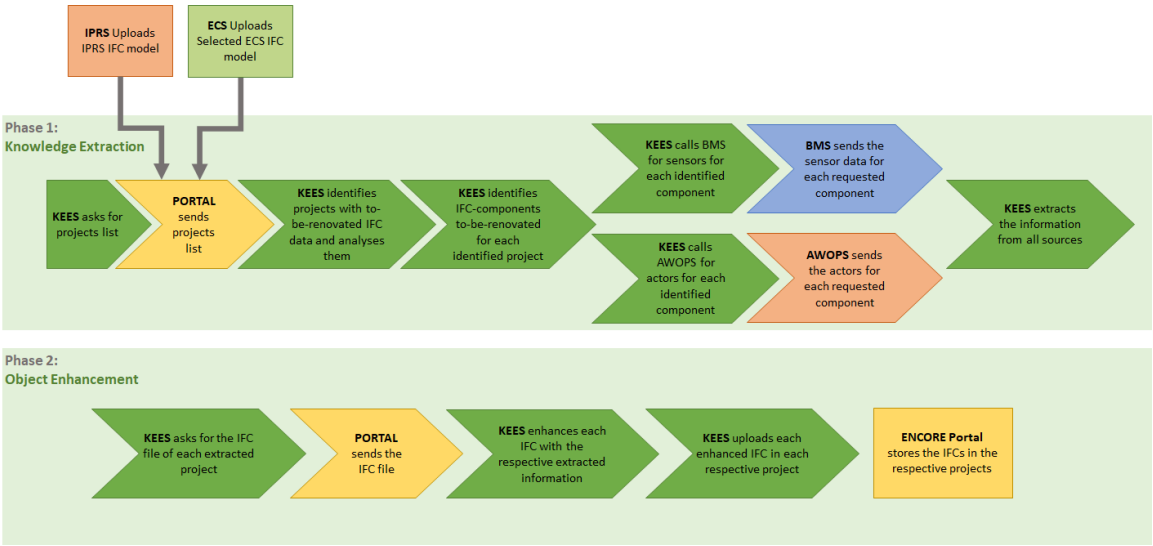

Figure 7: KEES Integration Communication Workflow

The individual interactions of KEES with the other ENCORE modules are being described with more details in the following sections.

3.1 INTEGRATION WITH THE ENCORE PORTAL

The ENCORE Portal, which is being described in detail in D5.2 ENCORE Full Prototype, aims to integrate the individual ENCORE Engines and Applications (Figure 1: KEES marked in the ENCORE layer architecture), offering the users a platform for managing energy efficiency and comfort building-renovation projects. Specifically for the ENCORE Engines (as KEES), the Portal, aims to provide a Graphical User Interface (GUI) from where the users can interact with their results (e.g. give inputs to be conveyed to the engines, download the results, etc.).

As a back-end service, KEES, interacts with the ENCORE Portal using a REST library provided by the Portal. The data acquired through this library, as well as their endpoint URLs, are as follows:

- Renovation projects list: a list of all the available renovation projects created through the portal. Those data include project and IFC model metadata (as explained in the next points).

  The endpoint for this REST call can be reached in:
  https://demo.encorebim.eu/api/projects

- Renovation project metadata: data that include project name and identification number, project owner, building location, building monitoring-gateway identification, and data on the available IFC models for the respective renovation project (e.g. initial building model, renovation-to-be-done model and model enhanced by KEES).

  The endpoint for this REST call can be reached in:
  https://demo.encorebim.eu/api/projects/{projectUUID}

- IFC file list: similar to the project list, this list includes the data (as explained in the next point) for all the available IFC models for a respective project, as this defined by its project identification.
  The endpoint for this REST call can be reached in:
  https://demo.encorebim.eu/api/projects/{projectUUID}/ifcs

- Specific IFC file: the data for a specific IFC file includes Universally Unique Identifier (UUID), upload timestamp, purpose (whether it was uploaded as initial building model by the user, uploaded from the ECS as selected renovation option or uploaded as enhanced by KEES), user that uploaded it, as well as identification of the respective project.
  The endpoint for this REST call can be reached in:
  https://demo.encorebim.eu/api/projects/{projectUUID}/ifcs/{ifcUUID}

The detailed documentation of the REST library can be found here:
https://demo.encorebim.eu/docs/swagger-ui/index.html?configUrl=/docs/api-docs/swagger-config.

3.2 INTEGRATION WITH BMS

The BMS aims to monitor the building which will be renovated, through sensors delivering information such as the temperature or illuminance, etc. These sensors are previously installed in the building and then configured via a gateway system (as described in deliverable D4.6.) This gateway system is identified by a number which is inserted in the ENCORE Portal during the project creation (as project metadata), to connect the BMS specific installation to the respective ENCORE renovation project.

Additionally, each sensor of the respective gateway is configured with a tag corresponding to components of the building found also in the IFC models.

In order for KEES to communicate with BMS it needs to get the project-specific gateway ID from the ENCORE portal, using the Portal REST library. This is available inside the project metadata as already mentioned in section 3.1.

Figure 8 shows the interaction of KEES with the BMS.

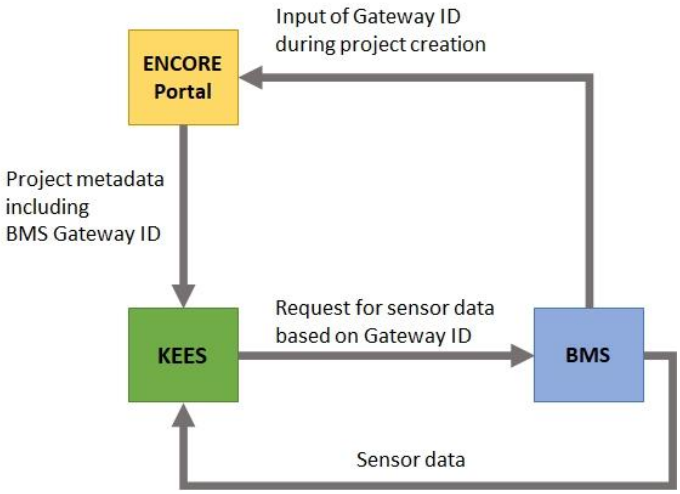

Figure 8: Communication between KEES and BMS

Once the KEES receives the gateway ID from the portal, and analyses a respective IFC model to get the renovation components, it can request sensor data from the BMS using the REST library that BMS provides. Those data, as well as their source endpoints include the following:

- Gateway sensor list: list of sensors connected to a specify gateway.
  The endpoint for this REST call can be reached in:
  https://spacloud.eu/public/9786582aaf93e/gws/{gw-id}/sensors

- Sensor data: data related to a specific sensor requested by its identification number, such as name, type and location-metadata (IFC related components). The endpoint for this REST call can be reached in: https://spacloud.eu/public/9786582aaf93e/gws/{gw-id}/sensors/{sensor-id}

- Sensor data based on IFC components: data of a respective sensor attached to the requested IFC component, defined by its ifc tag.

More detailed description of the BMS REST library can be found here:
https://spacloud.eu/public/9786582aaf93e/assets/api_rest_docs.html.

The information that KEES gets from the BMS is being analysed and connected to those taken from the ENCORE Portal and extracted from the IFC models, in order to create the enhanced information (to be added to the enhanced IFC).

3.3 INTEGRATION WITH AWOPS

The AWOPS aims to provide information on the construction works expected to be performed in the renovation project, as for example, data on the crew types (actors) involved in the respective renovation tasks. More details on the data available in AWOPS can be found in D3.4.

As also described for the other ENCORE services, KEES communicates with AWOPS also through the REST services provided by AWOPS. In order to retrieve the crew data from AWOPS, KEES gets the project IDs and the respective IFC with the renovation actions from the Portal, and calls AWOPS for the identified component to be renovated, as shown in Figure 9.

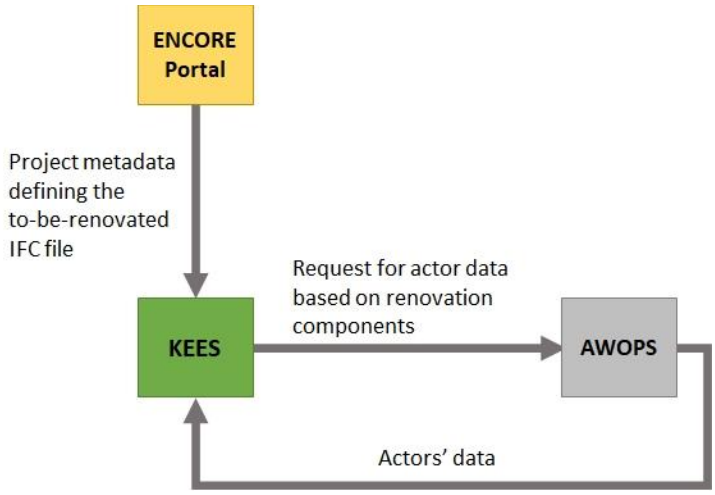

Figure 9: Communication between KEES and AWOPS

Similar to the sensor data from BMS, the crew data retrieved from AWOPS are being connected to the project information retrieved from the Portal and are being stored in the enhanced IFC file.

4. USER GUIDE

4.1 CONFIGURING KEES

The first step in order to start and operate KEES within the ENCORE integrated solution includes the configuration of the respective services. In order to do so, two xml files should be adjusted in the respective needs of the selected pilot scenario. The two files, defines the details of where the services run and which monitoring details should be used.

4.1.1 Configuration of the services starting point

The file for this configuration defines the classes and the server-details of the knowledge extraction, as well as the object enhancement services. As an example, the following services-config.xml can be considered:

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="http://www.atb-bremen.de"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <services>
        <service id="AmI-monitoring">
            <host>localhost</host>
            <location>http://localhost:19001</location>
            <name>AmIMonitoringService</name>
            <server>de.atb.context.services.
                    KnowledgeExtractionService</server>
            <proxy>de.atb.context.services.
                    IKnowledgeExtractionService</proxy>
        </service>
        <service id="AmI-repository">
            <host>localhost</host>
            <location>http://localhost:19002</location>
            <name>AmIMonitoringDataRepositoryService</name>
            <server>de.atb.context.services.
                    KnowledgeExtractionDataRepositoryService</server>
            <proxy>de.atb.context.services.
                    IKnowledgeExtractionDataRepositoryService</proxy>
        </service>
        <service id="PersistenceUnitService">
            <host>localhost</host>
            <location>http://localhost:19004</location>
            <name>PersistenceUnitService</name>
            <server>de.atb.context.services.PersistenceUnitService</server>
            <proxy>de.atb.context.services.IPersistenceUnitService</proxy>
        </service>
        <service id="EnhancedObjectService">
            <host>localhost</host>
            <location>http://localhost:19005</location>
            <name>ContextExtractionService</name>
            <server>de.atb.context.services.
                    ObjectEnhancementService</server>
            <proxy>de.atb.context.services.
                    IObjectEnhancementService</proxy>
        </service>
        <service id="EnhancedObjectRepositoryService">
            <host>localhost</host>
            <location>http://localhost:19006</location>
            <name>ContextRepositoryService</name>
```

```
            <server>de.atb.context.services.
                    EnhancedObjectRepositoryService</server>
            <proxy>de.atb.context.services.
                    IEnhancedObjectRepositoryService</proxy>
        </service>
    </services>
</config>
```

With this configuration, for the four processes of KEES, namely the Knowledge Extraction Service, the Knowledge Extraction Repository, the Object Enhancement Service and the Object Enhancement Repository we define the name-id (service id), the host and address (location tag), a name description (name), as well as server and proxy classes to be used for the execution.

This file is located in the configuration folder of the KEES service. For different operating systems there are different default locations, such as:

- Windows: C:\ProgramData\encore\config
- Linux: /var/lib/encore/config

Alternatively, the config file location can be configured using the environment variable ENCORE_HOME.

4.1.2 Configuration of the monitoring details

The file for this configuration defines the classes used for monitoring the pilot-specific data sources (e.g. REST services in this case), as well as the classes which will be used for the analyses of the incoming information. An example for the ENCORE pilot scenario is the following:

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="http://www.atb-bremen.de"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="monitoring-config.xsd">
       <indexes>
              <index id="index-encoreProjects"
                     location="indexes/encoreProjects"></index>
              </indexes>
       <datasources>
       <datasource id="datasource-encoreProjects" type="restservice"
              monitor="de.atb.context.monitoring.monitors.
                     EncoreProjectsRestMonitor"
              uri="https://demo.encorebim.eu"
              options="restServer=demo.encorebim.eu:443;
                     systemToken=1&amp;systemCookie=2"
              class="de.atb.context.monitoring.config.models.
                     datasources.RestServiceDataSource"/>
       </datasources>
       <interpreters>
              <interpreter id="interpreter-encoreProjects">
                     <configuration type="*"
                            parser="de.atb.context.monitoring.parser.
                                   EncoreProjectParser"
                            analyser="de.atb.context.monitoring.analyser.
                                   EncoreRestAnalyser"/>
              </interpreter>
```

```
        </interpreters>
        <monitors>
              <monitor id="monitor-encoreProjects"
                    datasource="datasource-encoreProjects"
                    interpreter="interpreter-encoreProjects"
                    index="index-encoreProjects"/>
        </monitors>
  </config>
```

This file allows different scenarios to be configured and for each an index is being created defining the scenario id. For each scenario, a data source is defined, describing which class is responsible for reaching the data, as well as possible options that need to be specific (e.g. address parameters). In case the data monitored from a specific data source need specific interpretation, an interpreter class can be defined. Additionally, one or more monitors can be defined, as for example, in the case we need different processing workflows for different combination of data sources, indexes or interpreters.

This file is located in the configuration folder of the KEES service. For different operating systems there are different default locations, such as:

- Windows: C:\ProgramData\encore\config
- Linux: /var/lib/encore/config

Alternatively, the config file location can be configured using the environment variable ENCORE_HOME.

4.2 BUILDING AND DOCKERISATION OF KEES

The KEES module is delivered as a Docker container, configured to interact with other containerized services of the ENCORE Project. The process of image building is automated using Jib Maven Plugin, an open-source tool managed by Google. It handles the packaging of Java applications into a Docker image without maintaining a Dockerfile or requiring a built JAR with all dependencies.

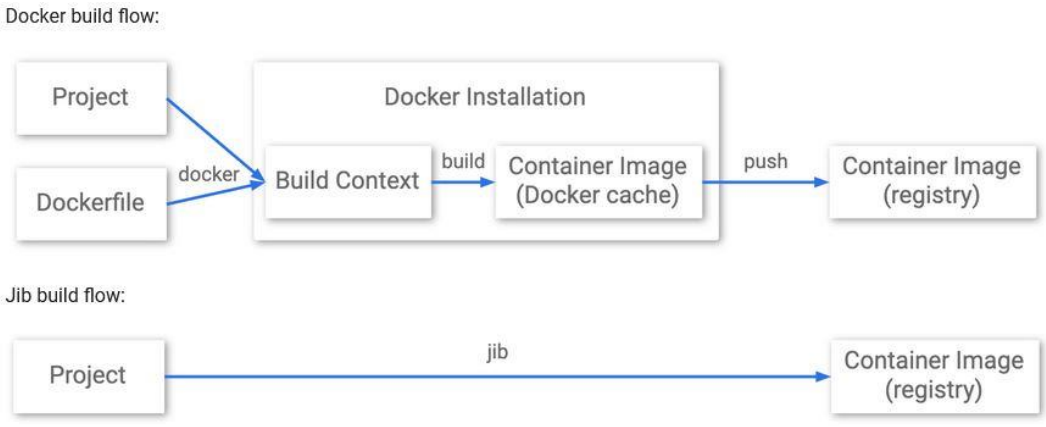

Figure 10 Jib build flow

Using the Jib functionality, new versions of KEES module are automatically built to images and pushed to the corresponding project folder at ATB Docker registry. Using Docker Compose functionality, the latest version of KEES module can then be picked up and redeployed on a server. The earlier image versions are also kept in the registry, allowing to quickly redeploy any stable version for test, backup or comparison purposes.

Version control and redeployment is further supported by Gitlab Multi-project Pipelines functionality. Pipelines are part of the inbuilt Gitlab CI/CD suit and are used to streamline build steps on the way to deployment. Since ENCORE Project contains multiple modules that get updated separately, it is important to keep track of all the changes and to deploy the newest version of each module on arrival.

Once the module build pipeline is activated and successfully creates new image, the central Deployment project is notified by a cross-project bridge job and starts a pipeline of its own. The new deployment pipeline stays associated with the given image version and can be activated to run a container with that specific module version on the server at any time.

4.3 RUNNING

4.3.1 Standalone installation

The KEES services can be downloaded from https://www.atb-bremen.de/artifactory/libs-releases-local/de/atb/encore/

After downloading the jar files, the services can be started with the following commands:

```
java -jar kees-services-2.0.0.jar
```

4.3.2 Docker container deployment

The docker containers for Situation Monitoring and Situation Determination can be downloaded from http://gitlab.atb-bremen.de by using the following commands:

```
docker pull gitlab.atb-bremen.de:5555/encore/deployment/kees
```

After downloading the container image, it can be started with:

```
docker start kees
```

4.4 RESULTS

The purpose of KEES is to analyse building-renovation data from different sources of a respective renovation-project, to correlate the data to each other and enhance the building model (renovation IFC file) with the findings. For the full prototype of KEES the results

are being added to the IFC file in the tag IfcProject, as additional description of the already created renovation project.

The current format of the added description is a json string, including information of the project itself, a list of the components that will be renovated (named as “tasks”), as well as related information for the connected sensors (acquired from BMS) and the involved actors (acquired from AWOPS). An example of the json string is the following:


```
{
   "projectID": "6ced4942-daf9-4c43-a16d-8022c3a30075",
   "tasks": [
     {
          "taskID": "15142",
          "sensorLocator": "IfcWindow",
          "sensorsAttached": [
                {
                      "sensorID": "0015BC001D0238B2",
                      "sensorDataType": "null",
                      "sensorDataUnit": "60"
                },
                {
                      "sensorID": "0015BC001B021376",
                      "sensorDataType": "null",
                      "sensorDataUnit": "115"
                }
          ],
          "actorsAttached": [
                {
                      "actorID": "456",
                      "actorRole": "Project Manager"
                },
                {
                      "actorID": "123",
                      "actorRole": "Architeckt"
                }
          ]
     },
     {
          "taskID": "47214",
          "sensorLocator": "IfcWallStandardCase",
          "sensorsAttached": [
                {
                      "sensorID": "0015BC001A006005",
                      "sensorDataType": "null",
                      "sensorDataUnit": "97"
                },
                {
                      "sensorID": "0015BC001D0238B2",
                      "sensorDataType": "null",
                      "sensorDataUnit": "60"
                },
                {
                      "sensorID": "0015BC001B021376",
                      "sensorDataType": "null",
                      "sensorDataUnit": "115"
                }
```

```
            ],
            "actorsAttached": [
                  {
                        "actorID": "456",
                        "actorRole": "Project Manager"
                  }
            ]
      }
   ]
}
```

Inside the enhanced IFC file, the above json is saved in the IfcProject tag (in the description property) as shown in Figure 11 below.

```
#102= IFCGEOMETRICREPRESENTATIONSUBCONTEXT('Body','Model',*,*,*,*,#97,$,.MODEL_VIEW.,$);
#103= IFCGEOMETRICREPRESENTATIONSUBCONTEXT('Box','Model',*,*,*,*,#97,$,.MODEL_VIEW.,$);
#104= IFCGEOMETRICREPRESENTATIONSUBCONTEXT('FootPrint','Model',*,*,*,*,#97,$,.MODEL_VIEW.,$);
#105= IFCPROJECT('3xRAmGI7HA_QHdb14Bh01d',#41,'1808938','{"projectID":
"6ced4942-daf9-4c43-a16d-8022c3a30075", "tasks": [{"taskID": "15142",    "sensorLocator":
"IfcWindow", "sensorsAttached": [{"sensorID": "0015BC001D0238B2", "sensorDataType":
"null","sensorDataUnit": "60"}, {"sensorID": "0015BC001B021376","sensorDataType": "null",
"sensorDataUnit": "115"} ], "actorsAttached": [ {"actorID": "456", "actorRole": "Project
Manager"}, {"actorID": "123", "actorRole": "Architeckt"}]}, {"taskID": "47214",
"sensorLocator": "IfcWallStandardCase", "sensorsAttached": [    {   "sensorID":
"0015BC001A006005", "sensorDataType": "null", "sensorDataUnit": "97"}, {"sensorID":
"0015BC001D0238B2", "sensorDataType": "null", "sensorDataUnit": "60"}, {"sensorID":
"0015BC001B021376", "sensorDataType": "null", "sensorDataUnit": "115"}], "actorsAttached": [
{"actorID": "456", "actorRole": "Project Manager"} ] } ] }',$,'Posthuset','FOREL\S\XBIGT
TRYK',(#97),#92);
#111= IFCPOSTALADDRESS($,$,$,$,('Matr.Nr.: 11gu Ballerup By,
Ballerup\X2\000D\X0\\PA\\X2\000A\X0\Baneg\S\erdspladsen 7\X2\000D\X0\\PA\\X2\000A\X0\2750
Ballerup'),$,'','','','<Default>');
#115= IFCBUILDING('3xRAmGI7HA_QHdb14Bh01c',#41,'',$,$,#32,$,'',.ELEMENT.,$,$,$);
#121= IFCAXIS2PLACEMENT3D(#6,$,$);
#126= IFCCARTESIANPOINT((0.0,0.0,28755.0));
```

Figure 11: KEES Result in the Enhanced IFC File

4.5 TECHNOLOGIES AND SOFTWARE TOOLS USED

To create the KEES based on its functional specifications, several technologies and tools were used. These tools support the creation of specific services.

The KEES services described in this deliverable were implemented utilising the Java programming language. For developing and maintaining the ENCORE Ontology, Protégé is used. Protégé is a widely accepted open-source platform that offers the ability to develop and visualise complex models, including properties and relations, and allows for export of the model in different formats (e.g. RDF, OWL, Turtle, etc.). To monitor the information out of files and web services several additional Java compliant libraries were used.

The repositories are using a POJO-based approach, which is serialised into an abstraction layer that covers many legacy database systems to work with (e.g. MySQL or H2). The specified KEES services rely on the abstraction layer as persistence system and do not require a mandatory database, but only a compliant one as technical foundation.

For the implementation of the specified KEES services, different development tools and IDE are used. For the overall development and orchestration of all system modules and components, the Eclipse IDE is used. The tested and widely accepted Open Source development environment for Java offers a modular system and a large plug-in community.

The repositories are based on a model- and object-oriented POJO approach. These are the foundation of the data flow and storage and can be easily extended or altered according to the changing requirements and embedded into the system again. Relations and models are used in correlation with the serialisation for sending and hibernation. The storage layer is, as described, based on a Java JPA solution. The JPA specification is referenced by the Hibernate project which gives the ability to create and use storage systems based on models, JPA-Specifications and POJOs.

The software tools used to develop and run the KEES service, together with their version and a website address are listed in the following Table 1.

Table 1: Overview of used software tools

Functionality	Software	Version	Website
Programming Language	Java	>= 11	http://www.java.com
XML Configuration Wrapper	Simple XML	>= 2.7.1	http://simple.sourceforge.net/
Database	H2 Database	>= 1.4	http://www.h2database.com/
RDF / OWL API	Jena	>= 2.8.7	http://jena.apache.org/
RDF Storage	SDB / TDB	>= 1.4 / 1.1	http://jena.apache.org/
Indexing	Apache Lucene	>= 8.8	http://lucene.apache.org/
Ppen source services frame-work	Apache CXF	>= 3.4.2	https://cxf.apache.org/
IFC framework	IFC Toolbox	>= 4.5.0	http://www.apstex.com/

5. SUMMARY

The KEES, currently being developed in T2.4, is a part of the BIM-based support tool for digital modelling and information generation (WP2). KEES is one of the so-called ENCORE engines, working in the backend and providing services to other modules of the ENCORE solution. KEES provides functionality for observing different types of data sources, extracting information from those sources, and correlating it, thus generating additional knowledge that is used to annotate and thus enhance objects in the BIM of a renovation project.

This deliverable describes the progress of the Knowledge Extraction & Object Enhancement Services (KEES), as it is prepared for the Full Prototype. This module, which is one of the ENCORE Engines (Figure 1), aims to analyse the building models (IFC files) of a selected renovation option and provided enhance information for the particular components to be renovated including sensor and related actor information. In order to do so, it communicates with the ENCORE Portal and the other services (BMS and AWOPS using REST libraries. More specifically, from the ENCORE Portal KEES acquires the project and IFC metadata, while from the BMS it acquires sensor information for the respective renovation tasks, as those defined by the IFC components, and from AWOPS it acquires the data for the actors related to each renovation action.

Additionally, information and guidance for the users to configure and run the KEES is being given, as well as a description of the results expected from its operation. As KEES does not include a dedicated graphical user interface, the indirect communication of the user is done through the ENCORE portal from where the IFC files are being uploaded. On those files the results of KEES are being stored and forwarded back to the Portal from where the user can have access to them.