



**Conférence Européenne  
des Directeurs des Routes**  
**Conference of European  
Directors of Roads**

**Connected Data for Effective Collaboration  
(CoDEC)**

# **Pilot projects report and consolidated implementation resources**

**Deliverable D3A**

**October 2021**



Project acronym: **CoDEC**

Project title: **Connected Data for Effective Collaboration**

## **CODEC Deliverable D3A**

### **Pilot projects report and consolidated implementation resources**

Due date of deliverable: 30.06.2021

Actual submission date: 07.09.2021

Start date of project: 01.10.2019

End date of project: 30.09.2021

**Author(s) this deliverable:**

José Barateiro  
J Barateiro, LNEC, Portugal  
A Antunes, LNEC, Portugal  
V Marecos, LNEC, Portugal  
D Kokot, ZAG, Slovenia  
S Bhusari, RHDHV, The Netherland  
S Wijdeven, RHDHV, The Netherland  
J Clark, TRL, UK  
J Proust, TRL, UK  
S Biswas, TRL, UK  
A Wright, TRL, UK  
F Luković, Bexel, Slovenia  
J Petrović, Bexel, Slovenia

## Table of Contents

Executive Summary	3
1 Introduction	4
1.1 Problem and objectives	4
1.2 WP3 and this report	4
2 The CoDEC Ontology	6
3 CoDEC technical architecture	9
3.1 High level overview	9
3.2 Overview of layers	10
3.3 Initial testing	12
3.4 Test concept	12
3.5 Methodology	12
3.6 Outcomes of the Test case	13
4 Pilot Project 1: Integration and 3D visualisation of monitoring data within a BIM Model	16
4.1 Concept and objective	16
4.2 Methodology and implementation	16
4.3 Outcomes of Pilot Project 1	26
4.4 Recommendations	28
5 Pilot Project 2: Linking and visualizing risk and condition data with a BIM model	29
5.1 Concept and objective	29
5.2 Methodology and implementation	30
5.3 Outcomes of Pilot Project 2	38
5.4 Recommendations	41
6 Pilot Project 3: Enhancing legacy data by linking BIM models to GIS systems	42
6.1 Concept and objective	42
6.2 Methodology and implementation	43
6.3 Outcomes of Pilot Project 3	55
6.4 Recommendations	56
7 Further outputs associated with this report	57

8	Conclusions	58
9	References	61
	Appendix 1 – Additional Deliverables	62
	Appendix 2 - Dynamo script	73
	Appendix 3 – Technical Tools Glossary	74

## Executive Summary

### The CODEC project

The CoDEC project aims to provide a better understanding of how BIM principles could be practically applied, within the European highways industry, to manage asset data during the operational phase. In particular, the project aims to develop a specification that will support the establishment of connections between asset management systems and BIM platforms - to make best use of the legacy and sensor/scanner data. CoDEC will deliver a “Master Data Dictionary” for key infrastructure Assets that can form the base for the data structure for integration between different data management systems. CoDEC therefore aims to free-up and enrich the data flow to and from BIM and asset management systems.

The research is divided into 6 Work Packages. Of these, Work Package 1 (WP1) has examined the aspect of legacy data and Work Package 2 (WP2) has examined the aspect of new data from sensors and scanning systems and developed a standardised specification for a “Data Dictionary” for three key infrastructure asset types. Work Package 3 (WP3) has developed a “Data Ontology” based on the “Data Dictionary” and has demonstrated the application of the data dictionary and ontology through three pilot projects.

### **This Deliverable (D3A - Pilot projects report and consolidated implementation resources)**

This report captures the methods and outcomes achieved through the pilot projects as an outcome of the WP3. The report also provides the developed software components, along with their documentation.

This report demonstrates the process of:

- Implementing the Data Dictionary and Ontology developed in WP1 & WP2.
- Visualization of the integrated data (including sensor/scanned data sets)
- Demonstration of the CoDEC approach within three Pilot Projects.

The pilot projects are being conducted with the support of three implementation partners:

- Pilot project 1- Agentschap Wegen & Verkeer (AWV, Belgian (Flemish) NRA) - integration and 3D visualisation of monitoring data on Tunnel.
- Pilot project 2 –Rijkswaterstaat (RWS, Dutch NRA) - to integrate data from multiple bridge specific sensors, and
- Pilot project 3 –FTIA (Finnish NRA)- Linking BIM & GIS for Road Asset Management.

The outcome of the pilot projects includes the developed methods and tools to implement data connectivity between BIM platforms and asset management systems. These are provided with this report.

# 1 Introduction

## 1.1 Problem and objectives

Building Information Modelling (BIM) is a model-based process of generating and managing digital representations of physical infrastructure and its (functional) characteristics across the whole asset life cycle. BIM concepts and workflows are well accepted in the Architecture, Engineering and Construction (AEC) industries, but transport infrastructure asset management has not yet reached the level of usage of these other industries.

In transport infrastructure, 3D virtual design to support construction projects (as provided by BIM) is generally implemented into current practice to enhance collaboration, sharing and exchange of infrastructure related information between stakeholders. This may resolve issues arising during the design, planning and construction phases. However, one of the most common problems in transport asset management is to have access to integrated information to support decisions in the operational phase. This challenge is becoming increasingly important and complex as more data is captured and recorded. Additionally, the increase in data captured leads to a greater number of data sources, which must be combined in order to provide an integrated environment for decision making.

The CoDEC project has explored the use of BIM as a framework for decision support, not only as a visual interface for the user, but also as a repository for information for decision-making. To enable the integration of several data sources in CoDEC it has been necessary to make the connection between BIM and operational data via semantic web and linked data principles, through the definition of a data repository based on ontologies.

## 1.2 WP3 and this report

The first two work packages (WP1, WP2) of CoDEC have defined data dictionaries to support links between BIM and operational activities (D1B and D2B). The challenge for the work of WP3, as described in this report, has been to define how this data should be organised and structured in a practical way to support its use in a BIM environment.

WP3 has then attempted to solve the problem of data integration with BIM via an independent, scalable, accessible and integrated solution that allows interoperability between the data and visualisation tools.

Finally, the practical implementation of the WP3 processes is presented via three case studies, with the overall objective of demonstrating an integrated visualisation environment, which helps stakeholders make decisions through access to information within a BIM model and a linked data environment.

This report is organised as follows.

- Chapter 2 describes the CoDEC ontology.
- Chapter 3 presents a summary of the technical architecture used by CoDEC and the initial testing of this in a case study related to light posts.

- Chapters 4 to 6 present pilot projects developed in 3 real-world scenarios, namely tunnels, bridges and roads.
- Chapter 7 provides the additional outputs through the Pilot Projects
- Chapter 8 presents the main conclusions of the application of CoDEC in 3 real pilot projects, discussing the main challenges and lessons learned.

## 2 The CoDEC Ontology

The CoDEC ontology<sup>1</sup> is based on the EurOTL framework<sup>2</sup>. The EurOTL provided a starting point (<https://www.roadotl.eu/static/eurotl-ontologies/index.html>) containing (already defined) concepts and properties that could be used or extended in the CoDEC ontology to achieve the necessary level of detail (for WP3 the level of detail being defined by the requirements of the pilot projects). This also ensures that the CoDEC ontology will work within any EurOTL interface or application. Hence, as alignment with EurOTL Framework was important, the first step in developing the Ontology was to map each CoDEC<sup>3</sup> on the EurOTL ontology. If a mapping was not found, a new CoDEC class or property was created. However, this was always a sub-class of an EurOTL concept (meaning that the new concept is an extension of the previous, thus ensuring interoperability between the two ontologies). The CoDEC ontology was developed using Stanford's Protégé (<https://protege.stanford.edu/>). Table 1 shows the relation between Data Dictionary and Ontology. If a mapping was not found, a new CoDEC class or property was created.

Table 1: Sample of Data Dictionary and Ontology relation

Data Dictionary			Ontology		
Property	Description	Format	Domain	Object/Data Property	Range
Bridge ID	The unique reference identifier for bridge	String	<a href="#">bridgelD</a>	is-a	Bridge
Bridge name	The name of the bridge	String	<a href="#">bridgelD</a>	<a href="#">rdfs:label</a>	<a href="#">xsd:string</a>
Environment	Classification of surrounding environment (e.g.: Rural/Urban)	String	<a href="#">bridgelD</a>	<a href="#">inEnvironment</a>	<a href="#">xsd:string</a>
Region/District/Area	Relevant geographical situation	String	<a href="#">bridgelD</a>	<a href="#">prov:atLocation</a>	<a href="#">eurotl:LocationByI dentifier</a>
Owner	Owner of the asset	String	<a href="#">bridgelD</a>	<a href="#">hasOwner</a>	<a href="#">prov:Agent (Person or Org.)</a>

As an example, "Bridge" already exists in the EurOTL Framework (AM4Infra: Bridge). However, a "Structural Element", or equivalent, is not found in EurOTL. For that reason, a new Structural Element class was created in the CoDEC ontology, as a sub-class of an already existing EurOTL concept (EurOTL:PhysicalObject).

<sup>1</sup> Linked Data and Semantic Web use ontologies to encode and share knowledge. An ontology is a "formal, explicit specification of a shared conceptualization" (Studer, et al, 1998), meaning that concepts, their constraints, and relationships should be encoded in a way that is explicit and machine-readable. This allows ontologies to be used for information integration and retrieval, and for semantically enhanced content, knowledge management and inference (Stephan, et al, 2007). The Resource Description Framework (RDF), RDF Schema (RDFS) and Ontology Web Language (OWL) were developed by the World Wide Web Consortium (W3C) and are used as standard in Semantic Web. RDF provides the base for "creation, exchange and use of annotations on the web" (Pan, 2009), using statements in the form of triples (subject property object). RDFS introduces class and hierarchy concepts, while OWL provides additional vocabulary and expressiveness (e.g., disjointedness, cardinality, object and data properties).

<sup>2</sup> Appendix 4 contains a short glossary with tools and technologies used for WP3

<sup>3</sup> Properties can be defined either as an object property or data property, meaning, respectively, a semantic relation between classes or between the class and literal data (e.g., strings or numbers).





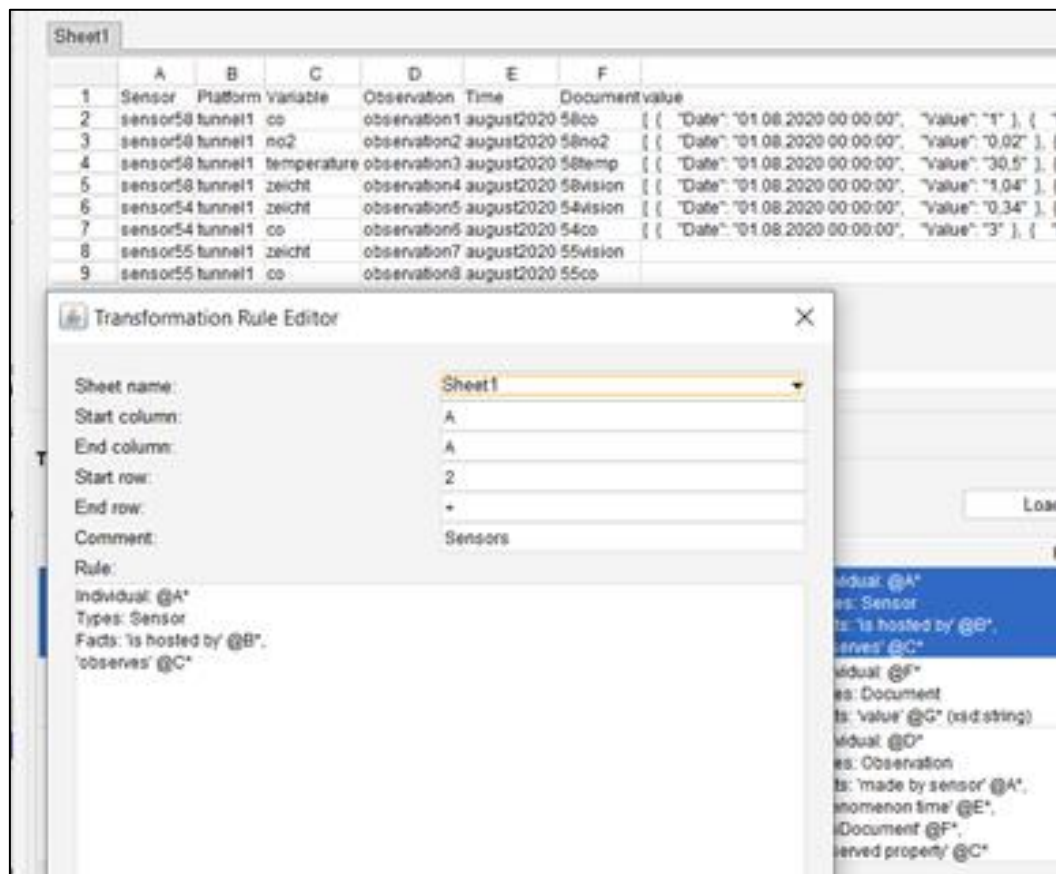
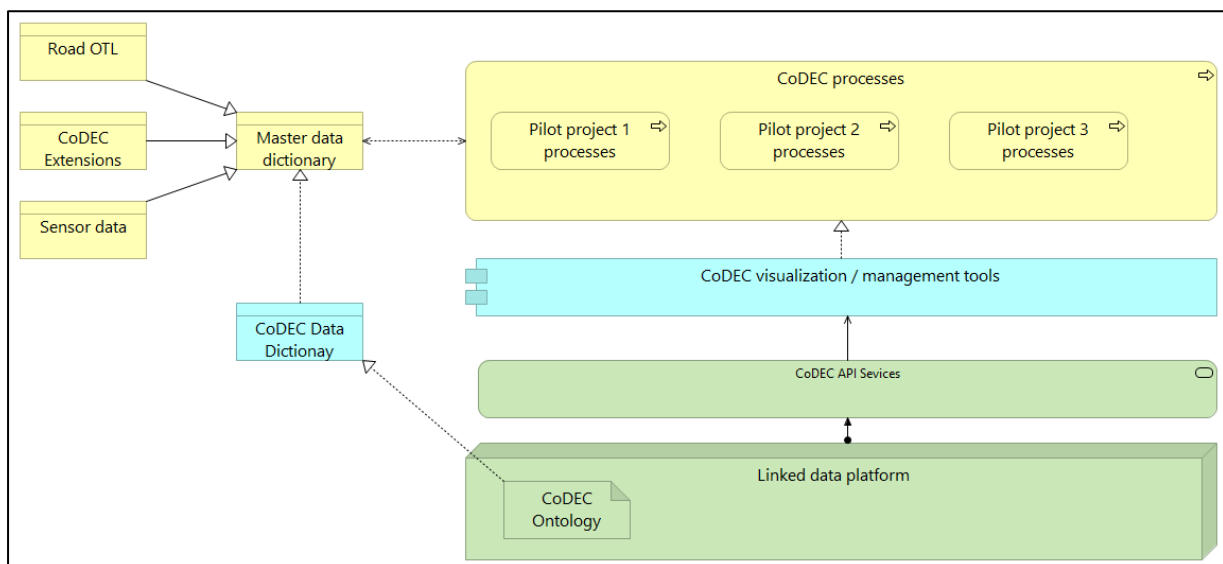


Figure 2: Cellfie Interface with Transformation Rules

## 3 CoDEC technical architecture

### 3.1 High level overview

Figure 3 presents the high-level architecture of CoDEC (for the Pilot Projects), following the Archimate notation, showing three layers. The first layer highlights the existing information on which the technical solution was developed - namely, the Road OTL ontology of the Interlink project (making it possible to implement the CoDEC ontology from the Road OTL implementations), and the data dictionary developed in WP1 and WP2 (which include the definition of the data concepts for pavements and structures and data provided by sensors). Note that to constrain the scope of CoDEC the dictionaries have been guided by the expected requirements of the 3 pilot projects. Details of the CoDEC ontology have been presented in Chapter 2. The ontology instances are stored in a Linked Data Environment, so they can be accessed and manipulated to meet the requirements of the different pilot projects.



**Figure 3: CoDEC technical environment**

The access to the ontology is via an Application Programming Interface (API). The CoDEC API is a critical component of the technical solution: It creates a layer of abstraction and independence between the data and logical levels; it allows any technological solution to access the linked data environment; it eliminates any technical dependency to access the linked data environment; it allows the ontology to evolve without changing the applications that access it through the API; it allows the complexity of the data to be isolated, i.e. it can be used by any application without needing to know the details of the implementation; it allows faster development; and it simplifies the entire process of testing and validation.

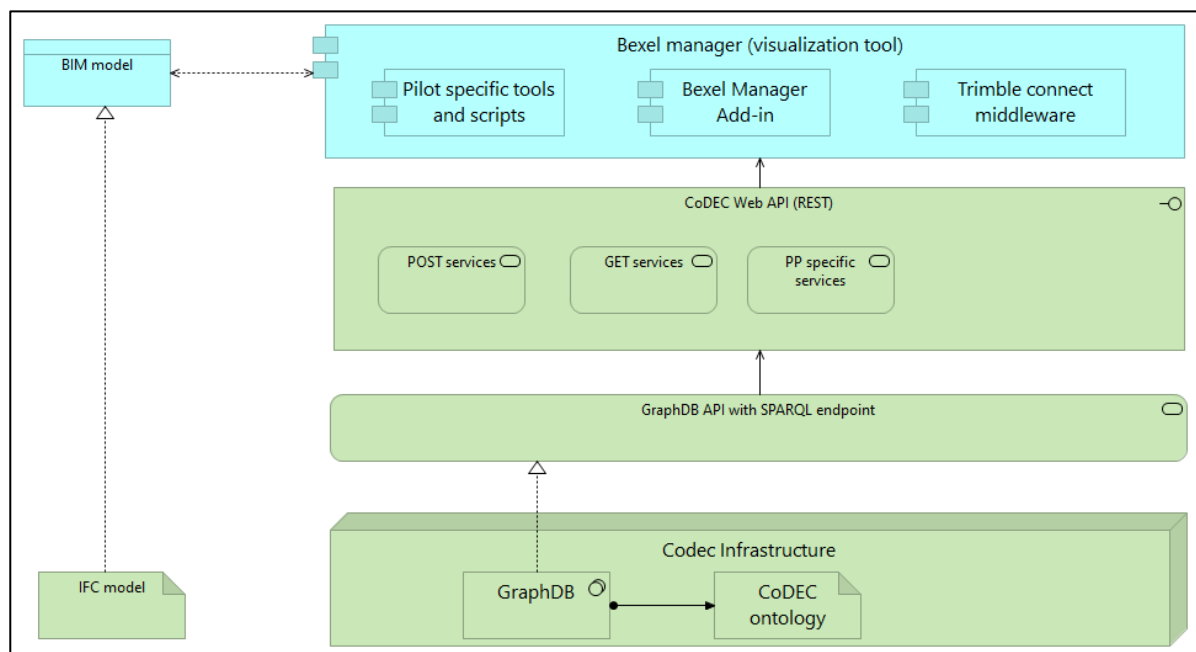
Finally, visualisation and data management tools allow access to the API to manipulate and access data in the linked data environment. For the end user, the only interface required with the CoDEC solution is the visualisation / data management tool, hiding all the complexity of the linked data environment.

### 3.2 Overview of layers

Figure 4 shows the CoDEC architecture has 3 “layers”.

- The data (lower) layer is where all data is stored.
- The logical (intermediate) layer implements the logical functionalities of the System and accesses and manipulates the data layer.
- The presentation (upper) layer provides the interface with the end user.

Following the best practice of systems architecture, these layers do not communicate directly with each other, but through a set of services that provides a level of abstraction between the different layers, hides the complexity of the implementation of each layer, and creates technological independence between the layers, i.e., each of them can be implemented using distinct programming languages and/or diverse applications. E.g., the presentation layer can use any tool or visualisation system by invoking the services provided in the CoDEC API.



**Figure 4: Technical architecture**

To provide the layered architecture, it was necessary to adopt a linked data environment that provides access to the CoDEC ontology through services, namely through a SPARQL endpoint. For this purpose, the ontologies were stored in a graph database (GraphDB, <https://graphdb.ontotext.com/>), which automatically provides a SPARQL endpoint through rdf4j (<https://rdf4j.org/>).

The layer provides data access and manipulation services via communication with the linked data environment. For example, a data service could obtain all the elements of a structure that were inspected in a given inspection. The logical layer is responsible for accessing and manipulating all the data and making it available to the user (typically the visualisation application). Once again, the access to the functionality of the logical layer is provided through a set of services, which were developed to support specific functionality required by the

CoDEC pilot projects. Note that this set of services, which can be developed incrementally, composes the CoDEC API.

In fact, the CoDEC API services are exposed for visualisation and analysis applications. From a technological point of view, these services are mainly web services that follow the pattern of a REST API, as well as an API with methods developed in Python, which are exposed to other applications and are documented using Python docstrings.

Finally, the visualization layer contains the tools that interface with the end user. In CoDEC these have been implemented to meet the requirements of each of the pilot projects. In general the functionality to manipulate or access data in the CoDEC ontology follows the following: (1) the user selects a certain functionality in the visualization environment; (2) the visualization tools identifies that this functionality requires access to a data service, isolated in the CoDEC API; (3) the visualization tool invokes the service made available in the CoDEC API; (4) the visualization tool processes the service response; (5) the visualization tools presents the results to the end user in the visualization interface.

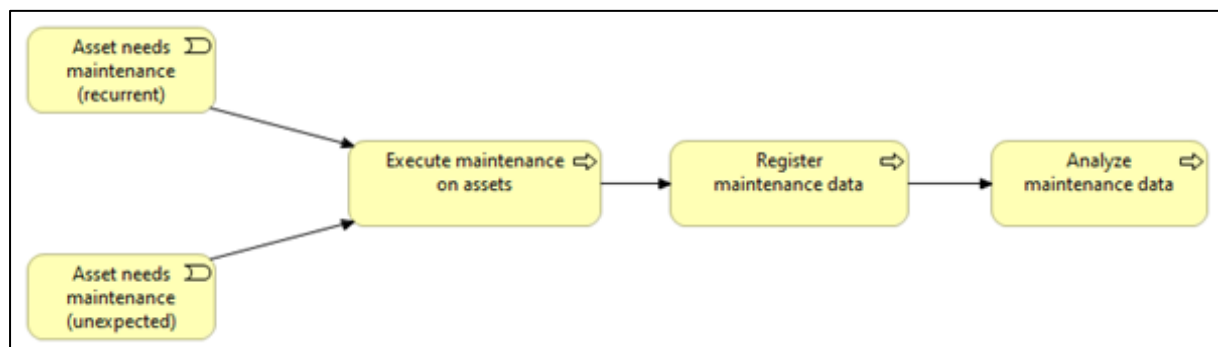
Despite the simplicity of the process described above, it should be noted that the CoDEC API services provide the functionality through the implementation of logic flows and the generation of SPARQL queries that communicate with the linked data environment. Once again, this access is ensured by a REST API, provided by the linked data environment through a SPARQL endpoint.

### 3.3 Initial testing

Development of the CoDEC technical architecture began with initial testing using smaller scale data, before exploring the Pilot Projects. This focussed on the information used in the maintenance of light posts (produced in the INTERLINK project). The testing aimed to check the feasibility of using information stored in an ontology in a linked data platform and to provide tools for the asset manager to access this information in the BIM environment. We have used Bexel Manager (see Appendix 3) as the BIM environment for this testing, Pilot Project 1 and 2. For Pilot Project 3, we have used 3D Repo as the BIM environment. This Chapter outlines the process and the outcome.

### 3.4 Test concept

Figure 5 presents an asset maintenance scenario. The maintenance lifecycle starts with an event that indicates the need to perform certain maintenance action on an asset (this event, can be a recurring event, i.e., occurring after a certain period of time, or an unexpected event, i.e., an event arising from an unscheduled situation). In both cases, the event leads to one or several maintenance actions (preventive or corrective) that are subsequently registered in the system. Maintenance information is then available, allowing asset managers to perform various analyses to support decision-making.



**Figure 5: Asset maintenance lifecycle**

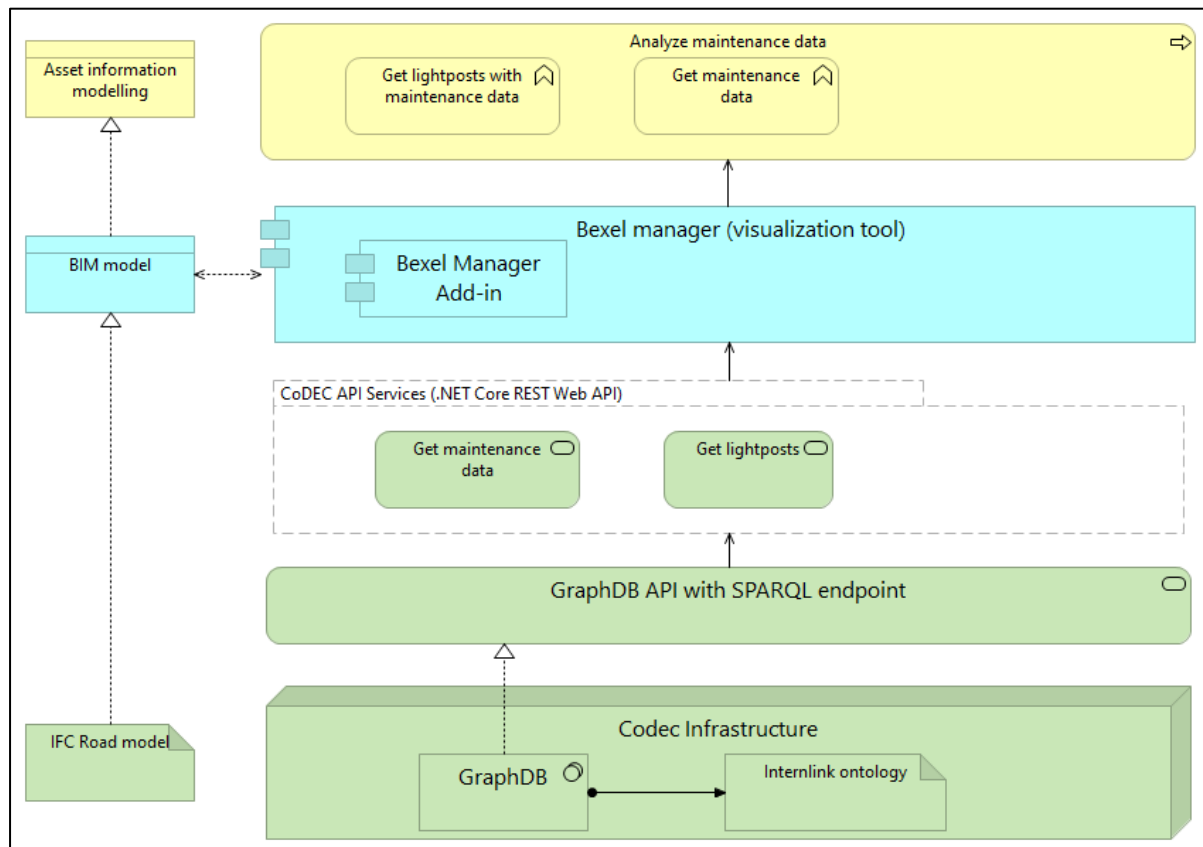
For this case, it is assumed that maintenance information is recorded in a given system and that it is accessible for analysis. In this particular case this corresponds to the instance of the ontology developed within the interlink project that includes maintenance information on light posts. Thus, the process that demonstrates the use of the technical solution from the CoDEC project is the process "Analyse maintenance data."

### 3.5 Methodology

As shown in Figure 6, the "Analyse maintenance data" process includes two functions, namely: "Get light posts with maintenance data" and "Get maintenance data". The visualisation that supports these functionalities is performed through an add-in to the Bexel Manager BIM model analysis tool. To enable Bexel Manager to access the light posts maintenance information, two services were developed in the CoDEC API, namely: (i) "Get

light posts" - gets all light posts containing maintenance data; and (i) "Get maintenance data" - gets maintenance data for a specific light post.

The services composing the CoDEC API were implemented in a Web API developed in Microsoft .Net Core. This layer is responsible for understanding and communicating with the Linked Data environment. In this case, through the generation of SPARQL queries executed using the SPARQL endpoint of the GraphDB graph database. Note that the database was instantiated with the ontology example provided by the Interlink project.

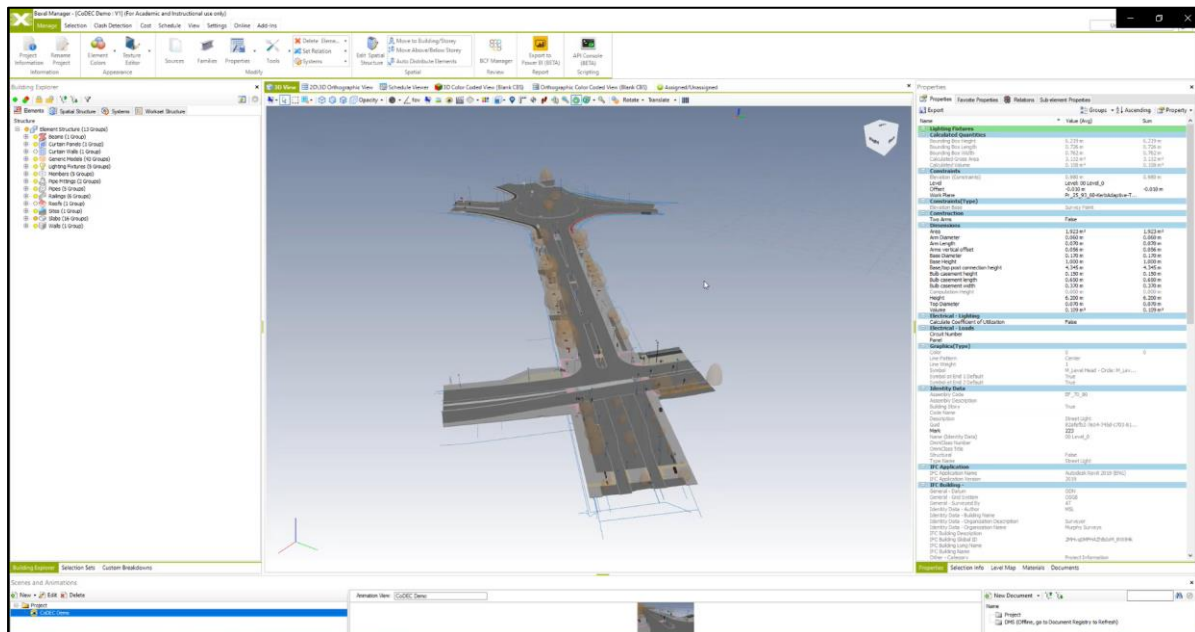


**Figure 6: Light post demonstration technical view**

### 3.6 Outcomes of the Test case

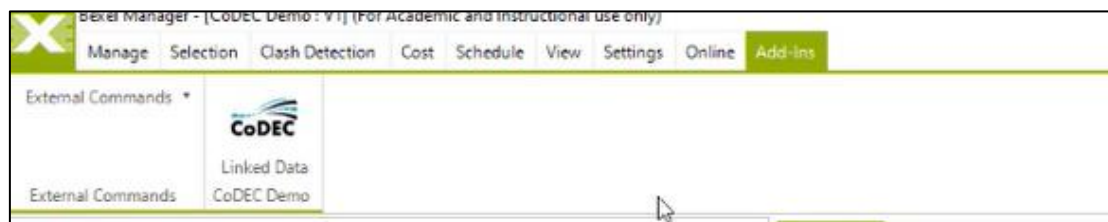
The test case successfully integrated data in the visualization environment (Bexel manager tool) via the CoDEC API to demonstrate the CoDEC approach. Figure 7 shows the integrated environment presented when the user opens the BIM model in Bexel Manager.





**Figure 7: Bexel Manager environment**

To access the integration functionalities provided by the CoDEC API services, the user accesses the Bexel Manager Add-ins, having access to the CoDEC Add-in directly from the normal Bexel Manager viewing environment, as illustrated in Figure 8.



**Figure 8: Bexel Manager CoDEC Add-in integration**

By using the Bexel Manager Add-in, the user gains access to the data stored in the linked data environment (through the CoDEC API). As illustrated in Figure 9 and Figure 10, the user has access in the same environment to all the BIM model information, together with the data stored in the linked data environment - which in this case corresponds to information about the light posts which have maintenance data, and the details of that maintenance data.



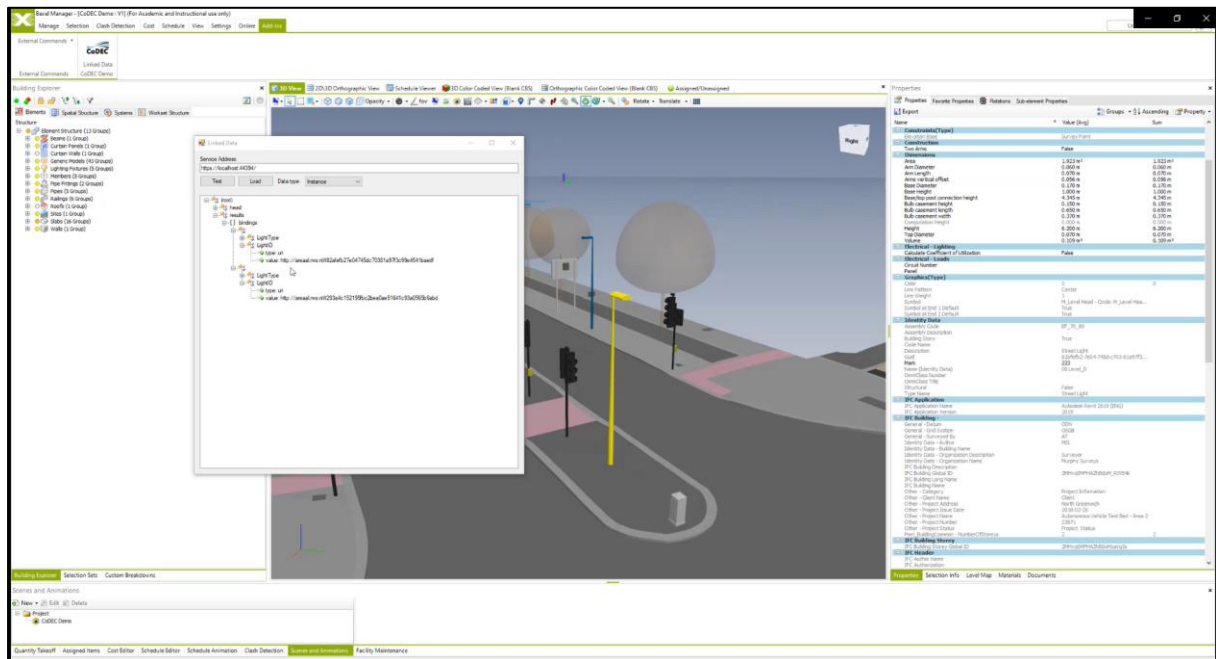


Figure 9 – Bexel Manager CoDEC Add-in: light posts

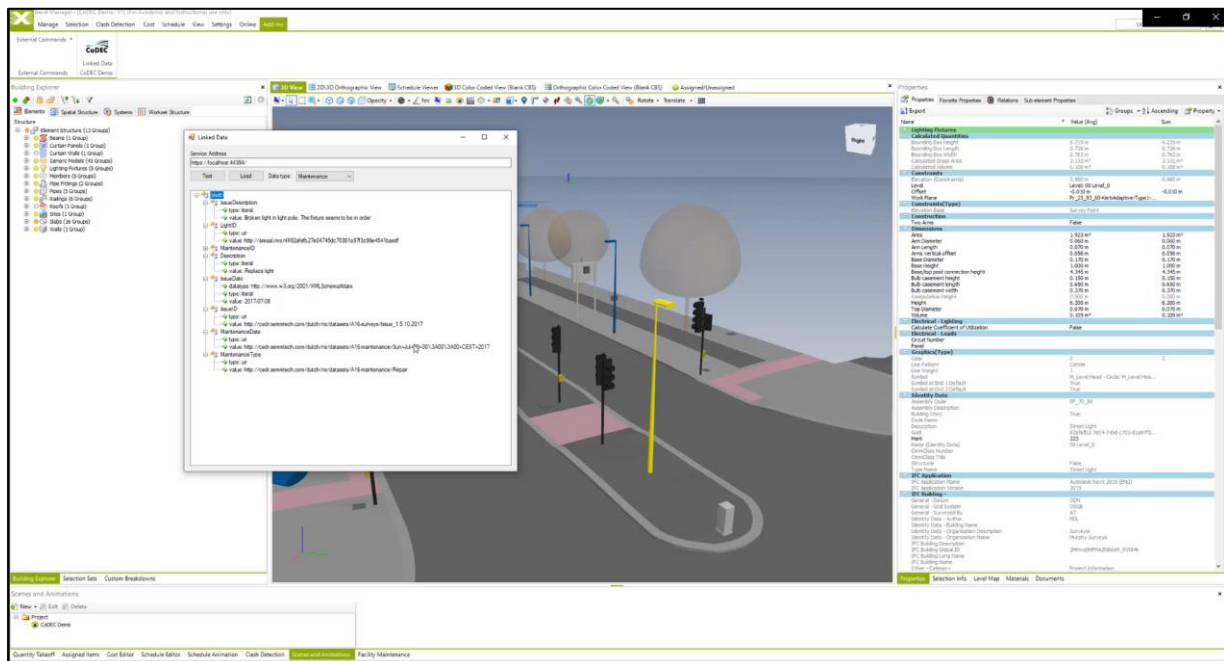
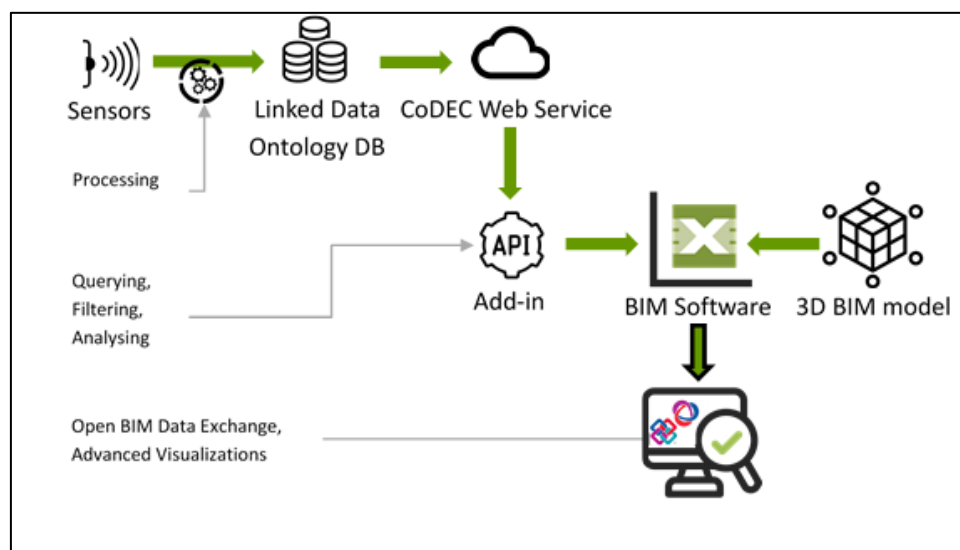


Figure 10 – Bexel Manager CoDEC Add-in: maintenance

## 4 Pilot Project 1: Integration and 3D visualisation of monitoring data within a BIM Model

### 4.1 Concept and objective

The objective of Pilot Project 1 (PP1) was to demonstrate the potential for improvement to inspection, monitoring and maintenance that could be achieved through the integration of monitoring data within a BIM model of a road tunnel (see Figure 11). The objective was to provide up-to-date information from a selected monitoring system in a 3D visualization in the BIM environment. This could be a great advantage during the operational phase, providing enriched analyses and assessment to help determine maintenance requirements. The concept of Pilot Project 1 is shown in Figure 11.



**Figure 11 – Pilot Project 1 concept**

### 4.2 Methodology and implementation

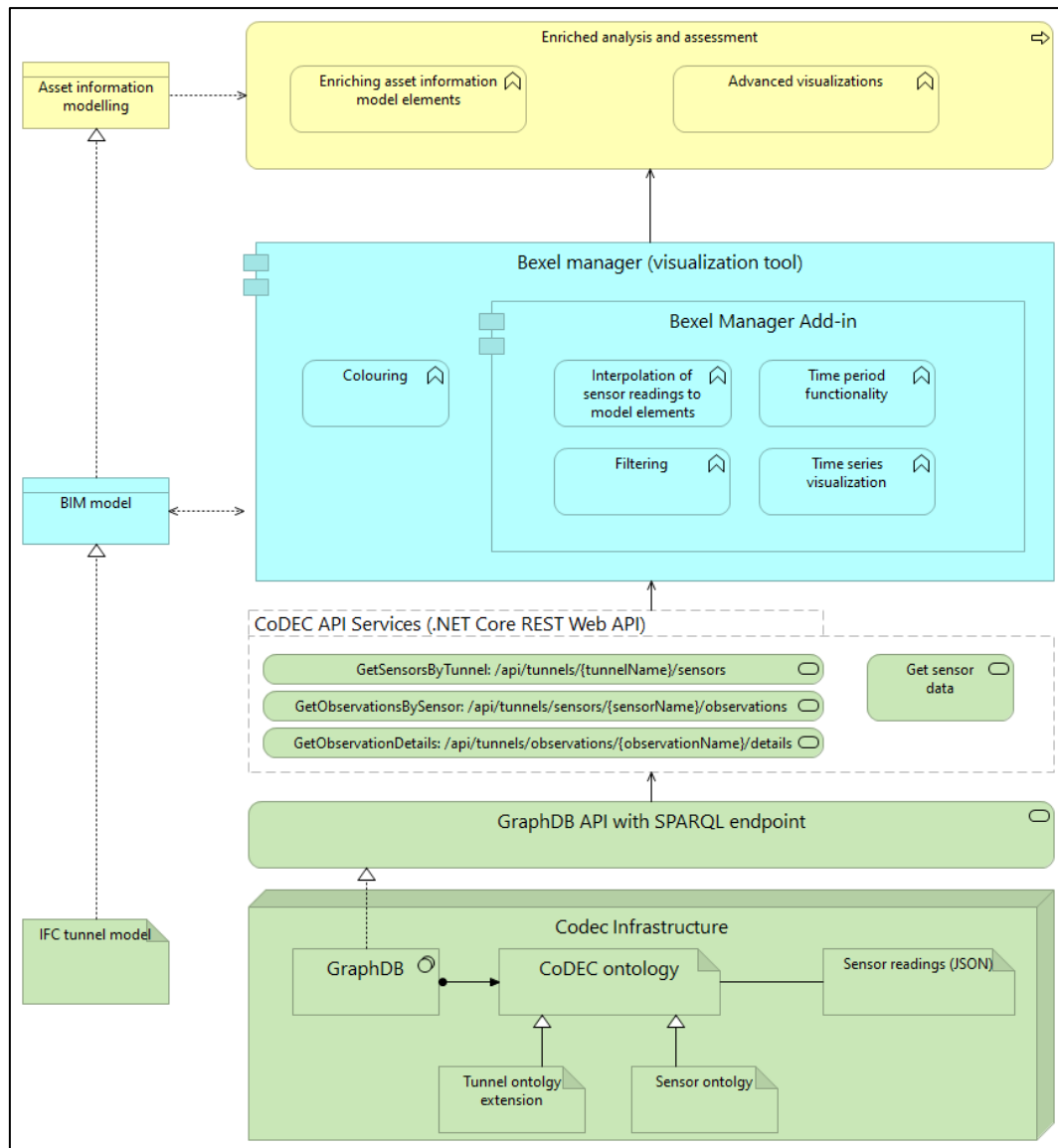
Asset managers need insight into the performance of the infrastructure and assets they manage and operate. Condition and other data are needed to understand the current and future performance of assets, predict their life expectancy, and plan maintenance actions. In addition, real-time monitoring is becoming increasingly important to identify problems in the infrastructure and its elements and take immediate action when needed.

The aim of PP1 was to develop a holistic procedure for handling monitoring and inspection data in an example tunnel, data processing and integration into the information model (Figure 12).

There were four so called use cases considered within PP1:

- Establishing the link between BIM model and monitoring data;
- Querying the external data (via CoDEC Add-in);

- Assessing performance through advanced 3D visualisation of the entire BIM model with monitoring data;
- Enhancing BIM model of a tunnel with tunnel OTL.



**Figure 12** - Pilot Project 1 technical architecture

The chapter discusses the challenges encountered in PP1 and how they were addressed on the way to completing practical implementation of a BIM platform software module successfully integrating sensor data and relevant analytics into the BIM environment. It also aims to demonstrate various use cases of the extended BIM software platform.

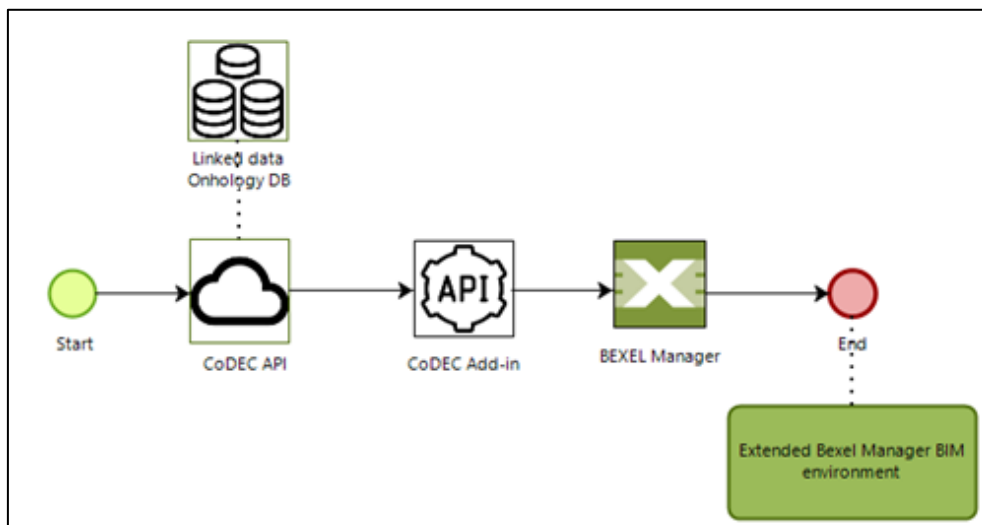
#### **4.2.1 *Link between BIM model and monitoring data and advanced visualisation of monitoring data***

Pilot Project 1 used Bixel Manager (a BIM project management platform) as the demonstration tool for data enrichment of the 3D BIM model. As noted above Bixel Manager allows easy 3D BIM model reviewing, updating and further 3D model enrichment with non-geometrical data. It can be also extended by improving existing or adding entirely new features through Add-ins developed using an open application programming interface (API). Particular functionality of this platform suited to CoDEC Pilot Project 1 included:

- Ability to import and federate multiple IFC model files
- Support for handling and analysis of BIM metadata
- Ability to extend data (add new attributes) directly in the system
- An interface for reading, analysing and automating the exchange of model metadata (such as scripts, extensions, add-ins) — this is critical because importing additional information from external data sources, such as structured JSON, XML or CSV files, is the envisioned concept of sensor data integration
- Ability to export integrated BIM with sensor data and links to external documentation, in open IFC file format (preferably both IFC2x3 and IFC4) to ensure compatibility with other supported BIM software

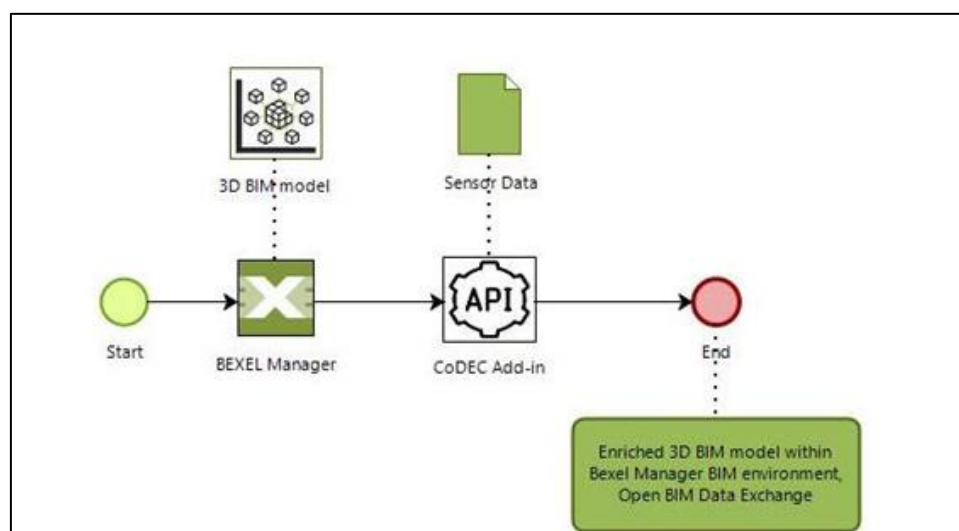
The first input needed for the implementation of PP1 was a 3D BIM model of the tunnel. The 3D BIM model was obtained from the implementation partner. It contained more than 27,000 elements, with a broad structure of categories, families and element types. This model was imported into Bixel Manager as an IFC open BIM format (ISO 16739-1). In addition, the implementation partner provided further information on sensors and the data they provided (CO, NO<sub>2</sub>, temperature, sight distance) in one month for upload to the CoDEC OTL.

After importing the 3D BIM model into Bixel Manager, a software Add-In allowed the user to connect to the data source to import the processed sensor readings into the BIM environment (Figure 13). In this particular case, the data source is represented by the CoDEC ontology database, which stores sensor readings. The user could also use sensor data provided in Excel format.



**Figure 13 - Process of extending Bexel Manager BIM environment with developed add-in**

Using simple filtering functions within the developed add-in and after generating properties, the sensor data was connected to corresponding sensor BIM model elements within Bexel Manager. Connecting sensor data to BIM elements was a 1-1 mapping process via a unique sensor identifier (Figure 14). On the BIM side, this identifier is defined as a BIM element attribute (property), while on the sensor data side it is defined as a data column. This kind of mapping enables an automatic, bi-directional relationship between the BIM elements and the related sensor data.



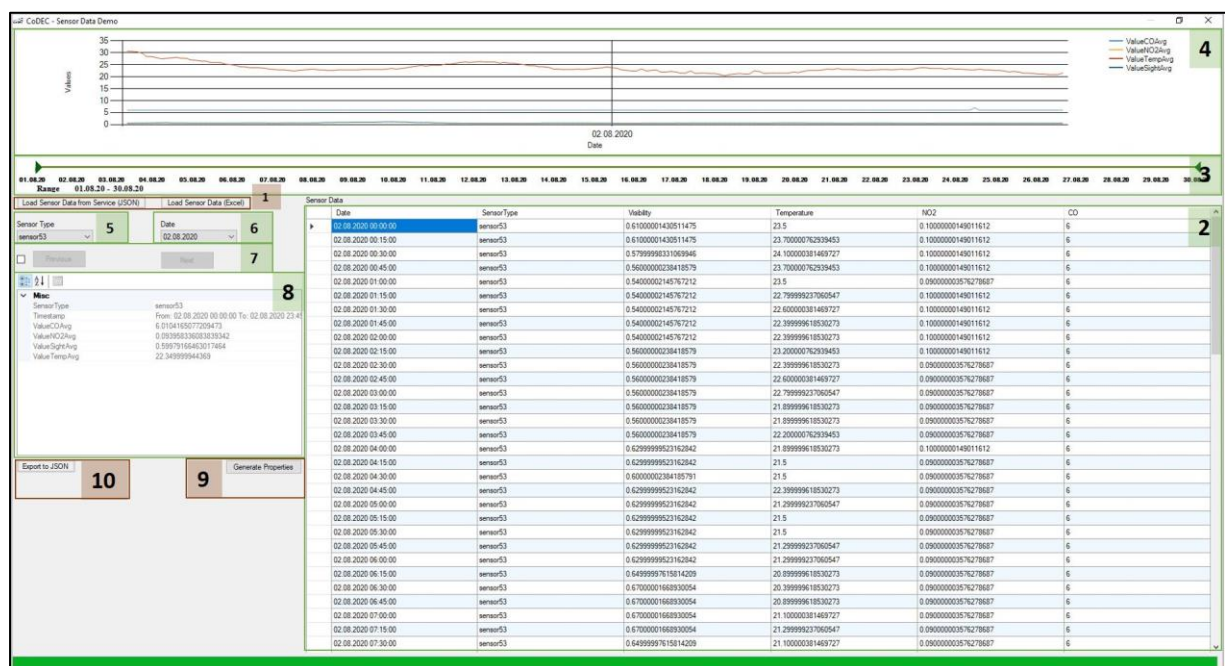
**Figure 14 - Process of integrating sensor data with BIM model**

Several filtering options for importing the data into the BIM environment were provided within the **CoDEC Add-in**. The user could select a specific sensor or all sensors. They could also select the time period, date and time for the data. The period could be selected by a developed timeline or by selecting a specific date. If the date is selected, there is an option to select a specific time as well. There is also a graph within timeline that shows values for the

filtered date/period. Once selected within the add-in interface (sensor, date, time period, time), the attributes that are going to be associated with BIM model elements are displayed. The selected data is then loaded into the BIM environment. Figure 15 shows the functionality for delivering sensor data into BIM environment, the numbers referring to:

- 1 – Importing Sensor Data
- 2 – List of filtered sensor data
- 3 – Timeline for choosing period of sensor data reading
- 4 – Graph that shows filtered sensor data readings
- 5 – Section for Sensor type selection
- 6 – Date selection
- 7 – Time selection
- 8 – Filtered Sensor type and Sensor data
- 9 – Generate Properties button
- 10 – Create JSON files (from XLSX format) for further use

The values of sensor readings and period/date/time, appear as attributes for specific sensor elements in the Bexel Manager BIM environment. These attributes appear in the Properties palette, under General section of attributes, such as *Timestamp*, *ValueCOAvg*, *Value NO2Avg*, *ValueTempAvg*, *ValueSightAvg*.



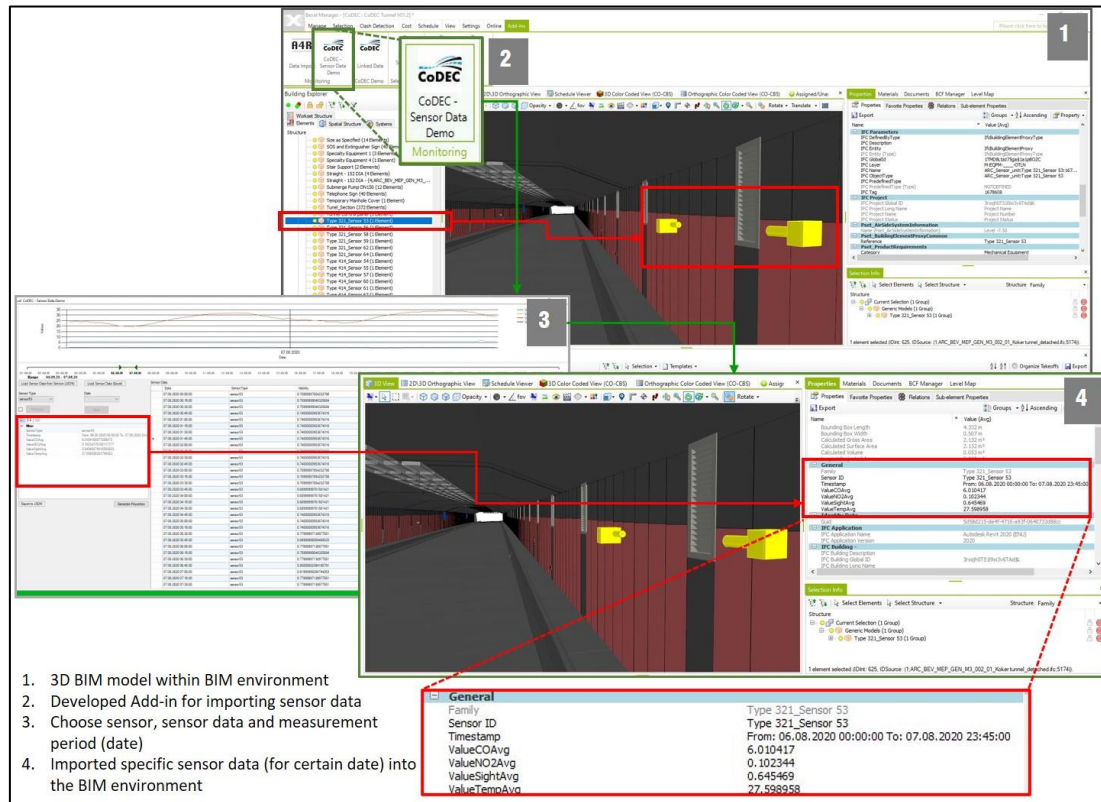
**Figure 15 – Developed CoDEC Add-in functionalities for generating sensor data into BIM environment**

The process of linking sensor data and BIM model elements can be repeated as many times as it is needed. Only the most recently imported sensor data values are persisted in the BIM model. This enriched BIM model can be exported into open standard format such as IFC and used in other BIM applications which support open standards. This allows the BIM model



(elements, geometry, properties, relations), with the additional integrated sensor data, to be accessible in an open, standardized, vendor-neutral way across all IFC supported software.

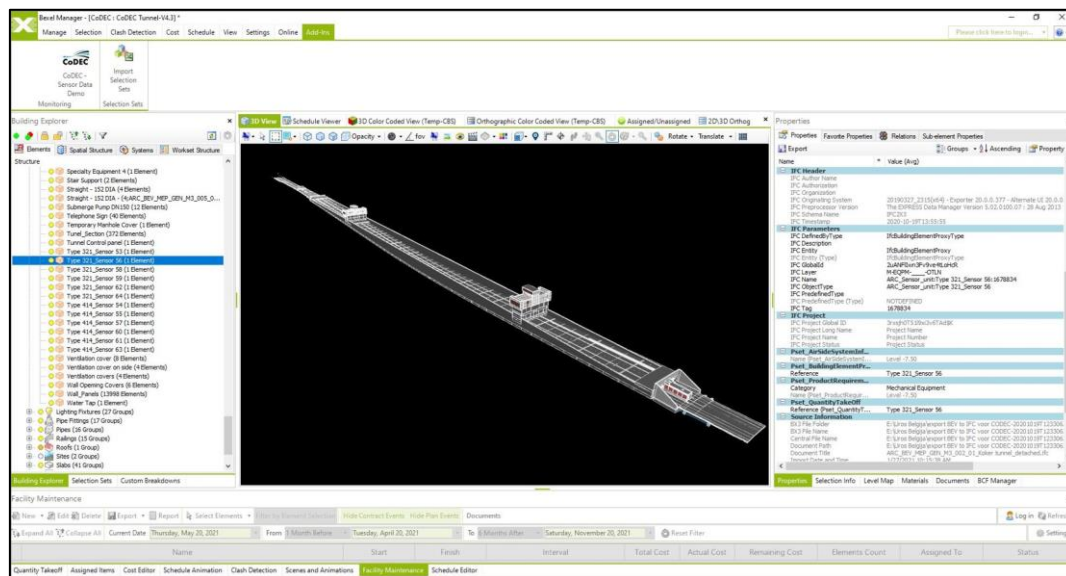
The following chapters describe requirements and technical information for successful implementation of PP1. Figure 16 shows the visual representation of the sensor data within a BIM Model.



**Figure 16 - Process of integrating sensor data with BIM model within Bexel Manager BIM environment**

#### 4.2.2 The BIM model

The BIM model of the tunnel was initially developed in Autodesk Revit 2020. The exchange files used to import the model into Bexel Manager were IFC 2X3 file format. In total 16 IFC files were imported into the Bexel Manager BIM environment, unified as a federated BIM model of the tunnel that served as a base for the Pilot Project 1 use case *Link between BIM model and monitoring data and advanced visualisation of monitoring data*. Figure 17 shows the BIM model of the tunnel within Bexel Manager.



**Figure 17 – BIM model of the tunnel within Bexel Manager BIM environment**

The level of development of the BIM model was suitable for associating sensor readings with specific BIM model elements. The sensor BIM model elements were *IfcBuildingElementProxy* type (*IFC Entity*). A particular type, *IFC ObjectType* (similar as the *Reference* attribute) that indicates the sensor further, is unique for each sensor. These attributes are used for linking sensor readings and BIM model.

As the sensor elements were distributed over large distances in the tunnel, it was challenging to find the right way to visualize the imported data. It was determined that the wall panels themselves would be a good way to visualise the sensor data as these wall panels, *IfcBuildingElementProxy* type were distributed along the entire tunnel and were very visible in the visualisation (see 4.2.6 below).

It is worth noting how the implementation partner that helped CoDEC prepare the pilot project on tunnels - Agentschap Wegen & Verkeer (AWV) - handles identifiers. AWV has well defined Object Type Library (OTL) principles and techniques that are the standard for BIM and AIM. OTL reflects the requirements of domain experts based on the relevant properties of objects. Unified Modelling Language (UML) and Object-Oriented Analysis and Design (OOAD) combine real world objects (types) with relevant properties such as attributes, classes and instances of classes, their associations and inheritance.

#### 4.2.3 Sensor Inputs and ontology validation and population

The Semantic Sensor Network (SSN) ontology is an ontology for:

- describing sensors and their observations,
- the involved procedures,
- the studied features of interest,
- the samples used to do so, and the observed properties,
- as well as actuators.



SSN is used as basis of the tunnel part of the CoDEC Ontology to demonstrate the link between BIM and the sensor readings through Linked Data. The sample data provided by the implementation partner was used to validate and populate the ontology. In order to consume the sensor data from CoDEC Ontology Database through the CoDEC API:

- The *Sensor* is hosted by the *Platform*, the location where the sensor is connected to (the tunnel or a more specific part of the tunnel (element))
- The sensor measures the set of properties and makes the *Observations*.
- The Observation concept is such that each observation is a time series of a particular ObservableProperty made by a certain sensor in a particular time.

Initial sensor data was received as XLSX file with sensor readings. This original file was converted to JSON format for easier use within ontology. JSON data structure:

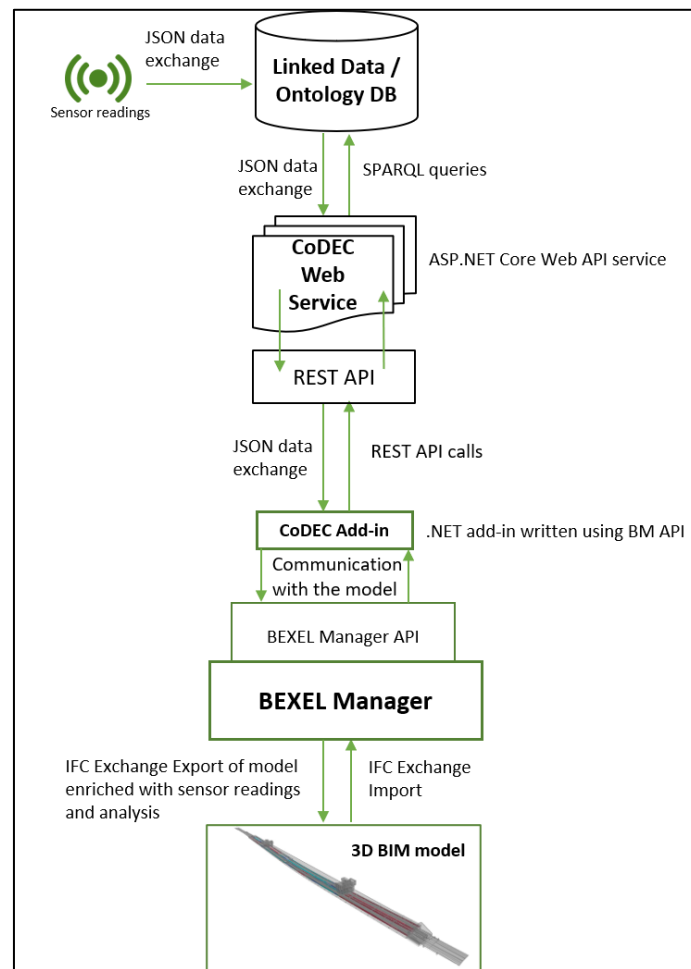
- Each Sensor contains several JSON files (one JSON file for each specific sensor attribute). Each JSON file gets information about characteristics for certain sensor (time period of measurement, values for specific attribute in certain time).

#### **4.2.4 Sensor Data Integration and connection with BIM through linked data**

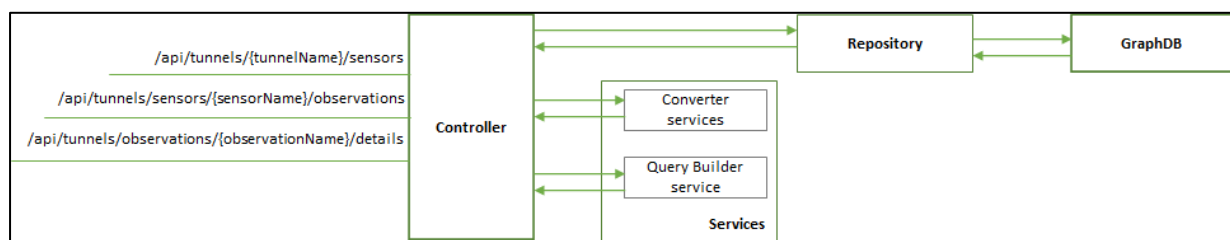
Custom add-ins can be developed to expand the functionality using the open C# API in Bexel Manager. Hence custom software modules can be developed for exchanging data from any external data source.

Sensor data was integrated with the BIM model through the client/server solution (Figure 18). On the server side there is an ASP.NET CORE web service (CoDEC API) which communicates with the Ontology database where all sensor readings are stored. On the client side there is an add-in which communicates with the BIM model as well as the web service. The web service is implemented in accordance with REST API rules. Based on a request made from the client application (e.g., certain sensor and interval), the service responds with the filtered sensor data in JSON format. In this particular case the user (client) would get information about all sensors, Observations they have made and the Observation data, using REST API calls. Once received, data will be processed and stored inside the model as attributes (properties).

Web services have been developed using Microsoft's ASP.NET Core framework. The stable, clean and safe architecture of Pilot Project 1 web services is achieved using the well-known Controller-Service-Repository pattern. Separation of concerns enables easily expanding the existing solution with new features (Figure 19).



**Figure 18 – PP1 implementation schema**



**Figure 19 – Developed Web API services**

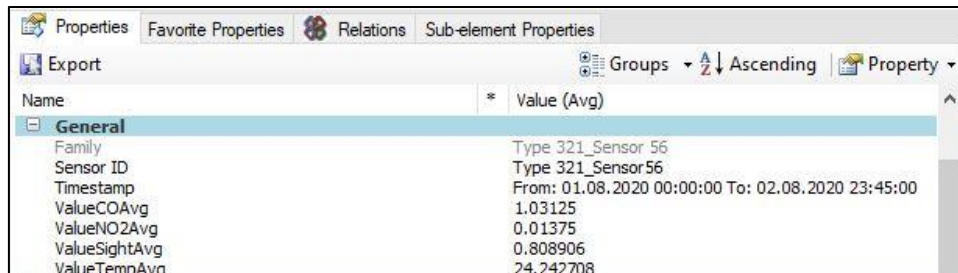
#### 4.2.5 Data Structure within the BIM

As an output of the integration process between the sensor data and the BIM model, five additional attributes were generated (for BIM model elements representing sensors):

- *Timestamp* – Date Time attribute representing the exact time the value was sampled;
- *ValueCOAvg* - Numerical attribute representing the (processed) CO value of the sensor;
- *ValueNO2Avg* - Numerical attribute representing the (processed) NO<sub>2</sub> value of the sensor;

- *ValueTempAvg* - Numerical attribute representing the (processed) Temperature value of the sensor;
- *ValueSightAvg* - Numerical attribute representing the (processed) Sight distance value of the sensor.

Each attribute appears after generating sensor data through the developed CoDEC Add-in (Figure 20).



Name	* Value (Avg)
<b>General</b>	
Family	Type 321_Sensor 56
Sensor ID	Type 321_Sensor56
Timestamp	From: 01.08.2020 00:00:00 To: 02.08.2020 23:45:00
ValueCOAvg	1.03125
ValueNO2Avg	0.01375
ValueSightAvg	0.808906
ValueTempAvg	24.242708

**Figure 20 - Example of generated properties in Bexel Manager - Output of the integration process between sensor data and BIM model**

#### 4.2.6 Sensor Data Advanced visualisation

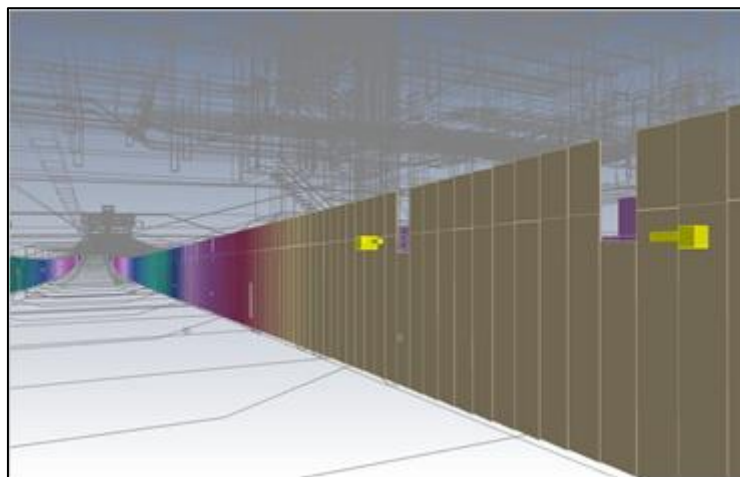
As noted above, questions were raised over how best to visualise the sensor data within the complex mode. To achieve this, the BIM elements *Wall panels* were given additional attributes, *ValueCOAvg*, *ValueNO2Avg*, *ValueTempAvg* and *ValueSightAvg*. The values of these attributes in the corresponding near zone of the sensor corresponded to the values of the (data) attributes for the observed sensor. The values of attributes for the *Wall panels* elements between two sensors were automatically linearly interpolated so that a continuous visual representation of sensor data could be made. Hence all *Wall panels* elements received these additional attributes (Figure 21):

- The *Start Chainage* and the *End Chainage* that represent station of the elements within the tunnel;
- The *Sensor ID* to recognise corresponding sensor area;
- The *Side(Wall\_Panels)* with defined four strings *Right-LTube*, *Left-LTube*, *Left-RTube*, *Right-RTube* which serves to define specific side of the tunnel.

Using the Custom Breakdowns (CBS) tool and Bexel Manager's 3D colour-coded view, the sensor values could then be shown in different colours (Figure 22).

Properties	
Export	Groups 2 Ascending Property
Name	Value (Avg)
<b>General</b>	
End Chainage	80801.868287
Family	Wall_Panels
Sensor ID	Type 321_Sensor 58
Start Chainage	80801.390798
Timestamp	From: 13.08.2020 00:00:00 To: 13.08.2020 23:45:00
ValueCOAvg	1.260417
ValueNO2Avg	0.048854
ValueSightAvg	0.828021
ValueTempAvg	29.270833
<b>Text</b>	
Section	LWPS80801.3907976718L0.477489698634599
Side(Wall_Panels)	Left-LTube

**Figure 21 – Additional Wall panel properties generated for advanced visualisation implementation purpose**



**Figure 22 - The Wall panels and sensor BIM model elements within Bexel Manager BIM environment (3D colour coded view)**

### 4.3 Outcomes of Pilot Project 1

The end result of Pilot Project 1 was sensor data successfully integrated into a BIM model of a tunnel, and advanced visualisation of this data. Once the sensor data has been generated into the BIM environment (using the developed Add-in) all the advantages of the BIM software can be used. In this particular project, the colour-coded 3D view shows the generated sensor values in different colours depending on numerical thresholds – e.g., using “red” to highlight locations in the tunnel where there are unacceptably high levels of emissions/pollution.

An issue that had to be overcome was - how actually can the sensor data be visualized? As the BIM model is not traditionally used to show “condition obtained from ‘real time’ measurements”, there was a need to determine how it could be adapted for this - what elements of 3D model can be used for such purpose? This issue was solved by developing an

automated process to apply sensor values to colour code the appearance of specific wall panels (Figure 23).

The Custom Breakdowns (CBS) tool is used to create a structure that divides our project according to a chosen criterion. The CBS used for this purpose are based on the continuous property which enables the user to have an overview of different ranges of the same property, such as *ValueCOAvg*, *ValueNO2Avg*, *ValueTempAvg* and *ValueSightAvg*. The user can easily change the colour and range of the observed property within Edit Custom Breakdown Property.



Figure 23 – Advanced visualisation of 3D BIM model elements

## 4.4 Recommendations

Pilot Project 1 demonstrated some of the potential benefits of linking operational data to the BIM model:

- It can support novel automated/improved inspection methods to enable faster and more timely inspection of infrastructure;
- It supports the development of centralized information models for data collection and processing (i.e., a single source of truth);
- The advanced 3D visualization tools help users to identify components that require (maintenance) attention. This addresses the need expressed by asset managers for tools to help to identify critical zones and maintenance issues in a reasonable timeframe.

For this work we have had to apply a novel solution to visualise data as this is not a core function of BIM models or their visualisation tools. To create a flexible solution where data can be visualized according to the user needs, designers of BIM are allowed to develop elements that better represent the subject of maintenance/monitoring. For example, a model be created within a tunnel describing the smallest possible segments (resolution relating to the smallest size of the Tunnel section) that represent the area of the sensor location to be able to connect the data from the sensor. Additionally, BIM designers can extend elements with the definition of properties to support advanced features, such as customized visualizations.

To take advantage of Bexel Manager visualization features, sensor data is imported from the CoDEC ontology via the CoDEC API into the BIM model, creating additional properties in this model. The main limitation of this solution is the lack of separation of concerns between information managed by the BIM model and those from the linked data environment, requiring that the BIM model is continuously updated to provide a visualization feature.

Automating the linking of data to BIM model elements requires standardisation, both on the part of BIM model development and on the part of maintenance operators. The success of Pilot Project 1, as demonstrated in this chapter, relies on standardization implemented by Agentschap Wegen & Verkeer. BIM model development and naming conventions are as important as standardisation of measurement data and corresponding BIM model properties.



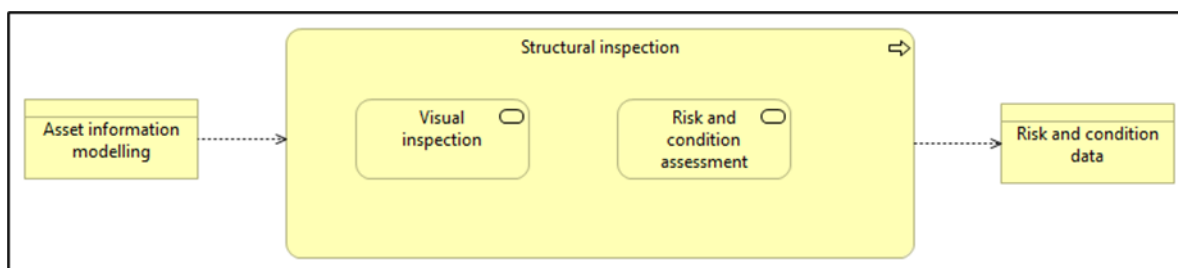
## 5 Pilot Project 2: Linking and visualizing risk and condition data with a BIM model

### 5.1 Concept and objective

Bridges exist in a large variety of shapes and span sizes based on a multitude of structural systems and materials. This results in many components, which, depending on the bridge and component type, can be composed of an even larger number of subcomponents. The number of defects and damage that occur through the service life of a bridge is also large and depends on the traffic, environmental conditions, ageing and various degradation phenomena, in addition to the structural relationships between components and sub-components.

The management of bridge safety and maintenance is therefore complex, with inspections having to consider each sub-component. Monitoring can be used to directly acquire data on local and global structural behaviour and hence continuous monitoring and inspection is one of the most important toolset, generating data from which meaningful information regarding the structural condition, ageing and operational capacity can be obtained (Figure 24).

The key objective of pilot project 2 was to integrate data on bridge structural condition into the BIM model of the bridge, to demonstrate the potentialities of a BIM platform as a framework to store information and provide a visual interface for bridge condition. The project focuses on the integration of condition data with a 3D visualization on top of the BIM model, and complements the developments of PP1 in the integration of sensor data.



Condition indicator/state		Definition	Procedures
1	Good	<p>Normal state of conservation.</p> <p>Existence of anomalies that do not affect structural behaviour but may compromise durability</p>	No repair work is required.
2	Regular	<p>Satisfactory conservation status.</p> <p>Existence of anomalies with some importance in terms</p>	Non-priority repair works are defined to be carried out in the long-term (6 to 10 years).

		of durability and/or functionality but with an insignificant impact on its structural behaviour.	Complementary diagnostic or monitoring actions can be recommended to measure the mid- and long-term evolution of detected anomalies.
<b>3</b>	Average	<p>Deficient conservation status.</p> <p>Existence of anomalies that significantly reduce durability and/or affect structural behaviour, or whose rapid evolution may affect safety.</p>	<p>Repair works are defined to be carried out in the medium term (2 to 6 years).</p> <p>If no significant evolution of its condition or service capacity is expected, the intervention may be reassessed at the next main inspection.</p> <p>The option as to the date of intervention or reassessment in the next inspection must be expressly indicated and duly justified. Additional diagnostic or monitoring actions may be recommended for measuring the short- and medium-term evolution of detected anomalies</p>
<b>4</b>	Insufficient	<p>Severe conservation status.</p> <p>Existence of anomalies that severely affect behaviour, resistant capacity and structural safety, with importance for integrity.</p> <p>It only meets the requirements to perform the function for which it was designed to a limited extent.</p>	<p>The onset of short-term intervention (up to 2 years) must be specified.</p> <p>The need for a reinforcement/rehabilitation project must be specified. Its exploitation can be restricted by conditioning the circulation or other preventive interventions such as shoring or temporary reinforcements.</p> <p>Complementary diagnostic or monitoring actions should be recommended. to measure the short-term evolution of detected anomalies.</p>
<b>5</b>	Bad	<p>Limit conservation status.</p> <p>Imminent failure situation, with the existence of anomalies that severely affect structural safety and integrity.</p>	<p>The start of the intervention urgently or very short term (up to 1 year) must be specified.</p> <p>The need for a reinforcement/rehabilitation project must be specified. Circulation restrictive measures, in terms of load, speed or mode of circulation, or other preventive interventions that mitigate the risk of exploitation, such as shoring or temporary reinforcements, must be implemented.</p> <p>In the extreme case, circulation can be prohibited.</p>

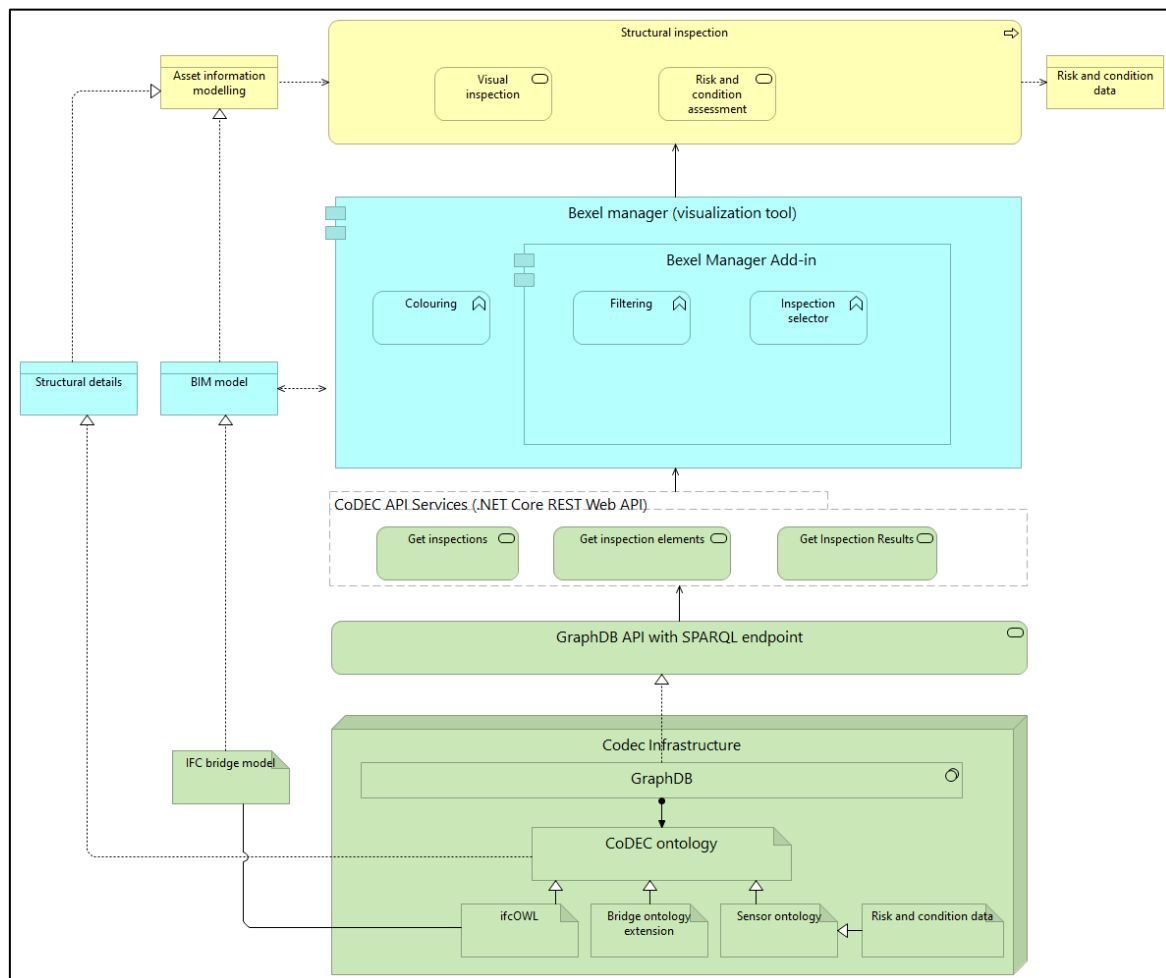
**Figure 24 – Structural Inspection Process**

## 5.2 Methodology and implementation

As noted above, CoDEC Pilot Project 2 aimed to demonstrate the use of the CoDEC Technical Architecture (Figure 25) to support the Structural Inspection process for bridges. This should



handle (and make accessible) risk and condition data, integrated and linked to the infrastructure asset (Bridge and its elements) in the BIM model.



**Figure 25 - Pilot Project 2 technical architecture**

### 5.2.1 Visualization tool development

Similar to PP1, PP2 uses Bexel Manager as the visualization tool. This tool was extended by the development of an Add-in to connect to the ontological environment (through the CoDEC API), allowing data filtering and selection, and interaction with the BIM model (colouring the elements based on risk and condition indicators).

To develop this pilot a 3D BIM model of a bridge was provided by the implementation partner (Rijkswaterstaat). The model contained 496 elements of four different IFC Classes. This model was imported into Bexel Manager BIM environment in IFC open BIM format (ISO 16739-1).

### 5.2.2 CoDEC API in PP2

A new set of ASP.NET CORE web services were added to the CoDEC API to support PP2. These services were related to accessing inspection information and risk and condition data on structures. Three services were added to enable the retrieval of information from the CoDEC

ontology without the need for SPARQL (query language) or data model knowledge. The services are described in the following sections.

**Service: Get Inspections:** Returns all Inspection instances. Inspections need to be instantiated as both an Inspection Activity (<http://www.roadotl.eu/def/InspectionActivity>), and a Procedure (from the SSN ontology), in order to be related to Observations. Currently, the service returns all inspections that are defined in the ontology connected to the CoDEC API for pilot project 2 (but could be updated to include filters, such as structure or a structural element). The service does not require any input parameters. It lists all inspections based on the schema of Figure 26. Inspection IDs and descriptions are the attributes retrieved for each instance. As an example, the JSON string of Figure 27 provides information about two inspections, including a unique identifier (inspectionID) that is critical to invoke other services.

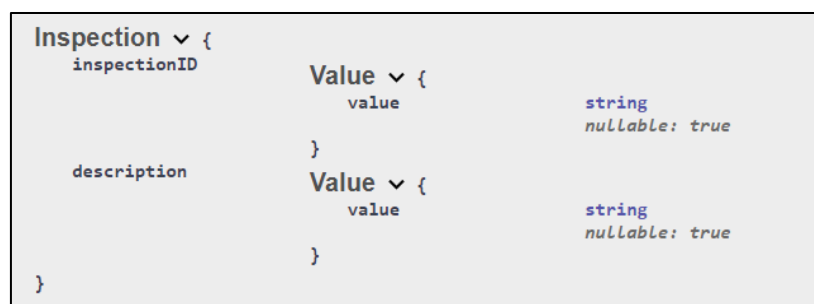


Figure 26 – Inspection schema

```

[
  {
    "inspectionID": {
      "value": "http://www.codec-project.eu/def#Inspection01Feb21"
    },
    "description": {
      "value": "Principal Inspection. 1st Of Feb, 2021. All Elements State And Condition"
    }
  },
  {
    "inspectionID": {
      "value": "http://www.codec-project.eu/def#Inspection06Mar21"
    },
    "description": {
      "value": "Routine Inspection, 6th Of March, 2021 Focus On Hanger State And Condition"
    }
  }
]

```

Figure 27 – “Get Inspections” response example

**Service: Get Inspection Results:** Returns all results from Observations related to an Inspection. Having the InspectionID as a parameter, this service returns pairs of elements GUID and their Risk and condition indicator (the inspection result). This allows the user to get a quick glance at the Inspection Results, and, since the connection between the model and the ontology information was secured by the use of ifcOWL, the BIM tool can make use of this data directly. This service requires the InspectionID as a parameter. The process is to get all inspections from the “Get Inspections” service, and then select one inspection from the result provided. Thus, after calling the “Get Inspection Results”, using a valid InspectionID, the

service provides a list of all inspected elements and the condition indicator (result), using the schema of Figure 28. The JSON string of Figure 29 shows a truncated list of elements inspected on inspection with InspectionID: <http://www.codec-project.eu/def#Inspection06Mar21><sup>4</sup>

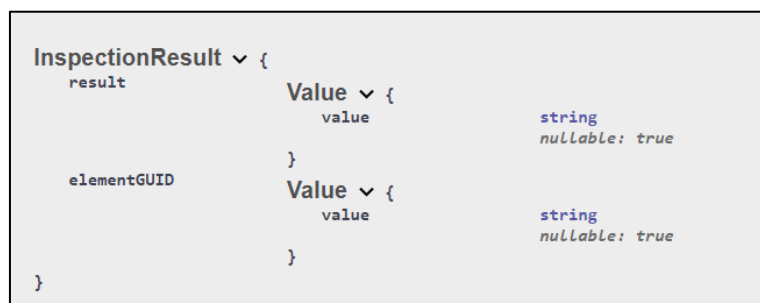


Figure 28 – InspectionResult schema

```

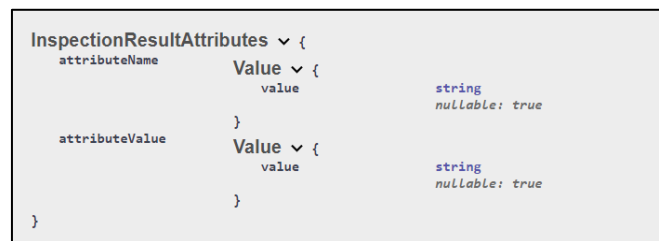
[
  {
    "result": {
      "value": "2"
    },
    "elementGUID": {
      "value": "23wH3G9sL3GgDaiKQFq9t1"
    }
  },
  {
    "result": {
      "value": "2"
    },
    "elementGUID": {
      "value": "23wH3G9sL3GgDaiKQFq9sB"
    }
  },
  {
    "result": {
      "value": "4"
    },
    "elementGUID": {
      "value": "23wH3G9sL3GgDaiKQFq9mA"
    }
  },
  {
    "result": {

```

Figure 29 – “Get Inspections Results” response example

**Service: Get Inspection Element:** Returns risk and condition data from a structural element concerning an inspection. Using the Inspection ID and Element GUID as filters, this service retrieves all the information contained in the Risk and Condition Result (indicator, next inspection date, etc.), using pairs of attribute name and value, according to the schema of Figure 30. The provision of results as pairs (attribute, value) does not limit the number of provided attributes. As an example, the JSON string of Figure 31 shows a truncated list of results for element with GUID: 23wH3G9sL3GgDaiKQFq9s3, with InspectionID: <http://www.codec-project.eu/def#Inspection06Mar21>.

<sup>4</sup> This link is IRI (Internationalized Resource Identifier) from the CoDEC Ontology. This allows to uniquely identify the instance in the knowledge base. However, since the knowledge base is not published, the IRI does not resolve, only serving its purpose in the linked data environment.



**Figure 30 – InspectionResultsAttributes schema**

```

[
  {
    "attributeName": {
      "value": "http://www.codec-project.eu/def#conditionIndicator"
    },
    "attributeValue": {
      "value": "2"
    }
  },
  {
    "attributeName": {
      "value": "http://www.codec-project.eu/def#conditionState"
    },
    "attributeValue": {
      "value": "Regular"
    }
  },
  {
    "attributeName": {
      "value": "http://www.codec-project.eu/def#nextInspectionDeadline"
    },
    "attributeValue": {
      "value": "2021-07-31T00_00_00"
    }
  },
  {
    "attributeName": {

```

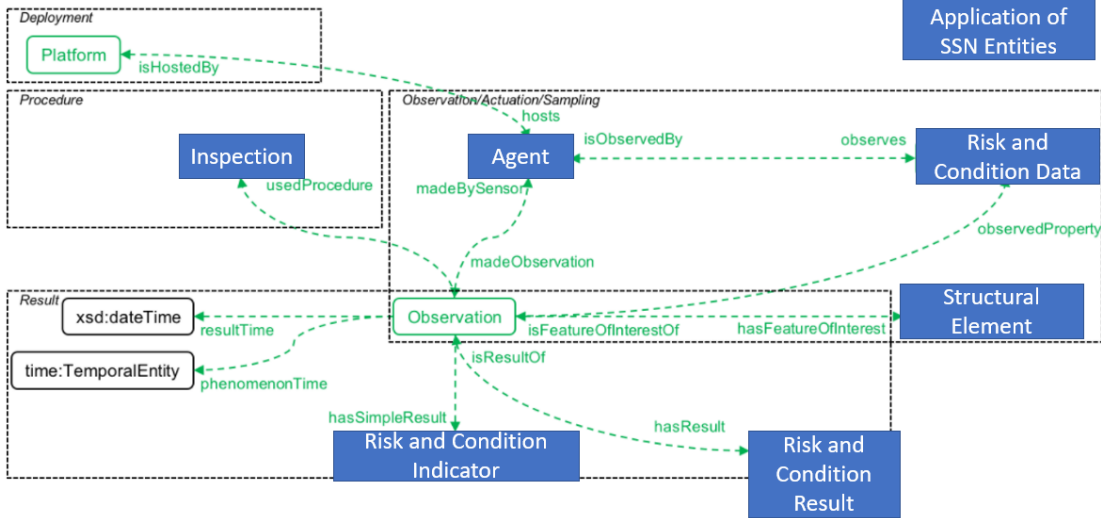
**Figure 31 – “Get Inspection Element” response example**

### 5.2.3 CoDEC Ontology

The CoDEC ontology is used in PP2 to achieve the following: a) Store knowledge about the Structure and its elements, b) Store information about inspection data (Risk and Condition Data) and c) Ensure the connection between the BIM model and the above information.

The CoDEC ontology is able to store information about the bridge and structural elements such as pylons and cables. The attributes necessary to define these entities are presented in the CoDEC data dictionary. They were developed (added to the ontology) based on the requirements of the pilot project.

The SSN ontology was used to represent “sensors and sensor data” (see Chapter 4). PP2 utilizes a similar approach to PP1 as Risk and Condition data is considered to be fundamentally similar to sensor data. The process of acquiring bridge condition includes a visual inspection of structural elements followed by a risk and condition assessment. In this process, an expert (more precisely, their eyes) takes the roll of the “sensor”. The Agent is responsible for implementing the Inspection procedure (the visual inspection and assessment), which in return creates a set of Observations (each one related to a structural element) (Figure 32).



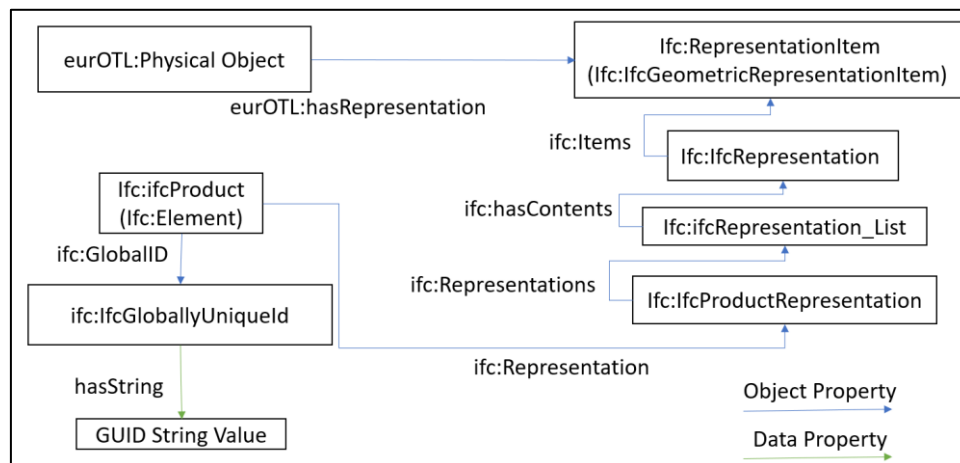
**Figure 32 – CoDEC Application of SSN Entities for Risk and Condition Data Analysis**

Whilst the result of each Observation in PP1 was stored as a JSON file in the ontology, in PP2 the results can be saved either as a simple value or as a full result. To use a simple risk scale (e.g., 1-5), the relation *hasSimpleResult* is used. However, if a more complete analysis on the risk and condition of an entity is required a different approach is used. In this case, the Result class of SNN is extended to obtain a Risk Analysis Result which in turn is able to represent all necessary information concerning the Pilot Project (Figure 33), including the URLs for the Photographs taken during the inspection.

Entry	Name	Data type	Examples
1	<u>Datetime</u>	POSIX (time)	2021-03-06 11:20:00
2	<u>Condition indicator</u>	<u>integer</u>	NI, 1, 2, 3, 4 and 5
3	<u>Condition state</u>	<u>string</u>	" <u>Excellent</u> " to " <u>bad</u> "
4	<u>Maintenance indicator</u>	<u>string</u>	"A", "B", "C", " <u>Sufficient</u> ", Insuficiente"
5	<u>Maintenance state</u>	<u>string</u>	" <u>Drainage blocked</u> , <u>need cleaning</u> "
6	<u>Alert</u>	<u>Boolean</u>	<u>Yes</u> or <u>No</u>
7	<u>Alert description</u>	<u>string</u>	( <u>reason for alert</u> )
8	<u>Awareness/surveillance state</u>	<u>string</u>	"normal", " <u>high</u> "
9	<u>Next inspection type</u>	Vector of strings	"principal" " <u>routine</u> "

**Figure 33 – Risk and Condition attributes example**

The last challenge was to make sure there was a connection between the ontological knowledge and the BIM model (IFC model). To achieve this connection ifcOWL ontology was utilized. The ifcOWL ontology is part of the eurOTL framework, relating an IFC model representation to a Physical Object, such as CoDEC's Structural elements. This geometry representation is then connected to an ifcElement in the ontology, with a defined GUID (Global Unique Identifier), following a complex path with several relationships (Figure 34).



**Figure 34 – Connection between eurOTL's Physical Objects and ifcOWL's GUID**

The ontology for PP2 was populated using Cellfie, as explained in Chapter 3. However, there was a challenge when using ifcOWL. The first step was to translate the IFC model to ifcOWL using official tools (<https://technical.buildingsmart.org/standards/ifc/ifc-formats/ifcowl/>). Despite being successful, this resulted in a significantly large file which led to complications (since the ifcOWL file contains all information from the BIM file). To overcome this problem, the ifcOWL entities were manually added to the ontology (with the correct GUIDs and other information from the original model), utilizing only the entities necessary to ensure the correct connection between the BIM model and the ontology (an “ifcOWL lite” of sorts).

#### 5.2.4 Data Preparation – Ontology Instances

For the purposes of the PP, knowledge about the structure, structure elements, inspections and risk and condition data need to be added to the project ontology. In addition, ifcOWL entities also need to be created to correctly link the knowledge to the BIM model. The import was done using Cellfie, a Stanford's Protégé plugin that allows the creation of new axioms from an excel spreadsheet. With the CoDEC project ontology opened, Cellfie is loaded with the Workbook (Figure 35) containing the necessary information for the use case. Dummy data was created based on existing BIM model elements (referenced by Global Unique Identifiers - GUID). To proceed to the import, Cellfie utilizes Manchester Syntax to define a set of Transformation Rules. Rules were defined to create class instances and the corresponding relationships of each individual. E.g., Figure 36 creates the individuals (column A) for *InspectionActivities* and relates them to a description string (column B) using a data property (description). In this case, the rule creates eight new axioms: 2 axioms define the new Individuals, 4 axioms assert the individuals' type (both *InspectionActivities* and *Procedure*) and, lastly, 2 axioms for the data property.

Workbook (C:\Users\anton\Desktop\testDataPC2.xlsx)

	A	B	C	D	E
1	Inspection	InspectionDescription	Name	Bridge	Element Type
2	Inspection 01Feb21	Principal Inspection, 1 of Feb of 2021. All elements state and condition	_03-Hangers_SFR_Hanger_Strand system_Hanger 1. (I-1)_18750356	Bridge1	Tie Beam
3	Inspection 01Feb21	Principal Inspection, 1 of Feb of 2021. All elements state and condition	_03-Hangers_SFR_Hanger_Strand system_Hanger 1. (I-1)_18750384	Bridge1	Tie Beam
4	Inspection 01Feb21	Principal Inspection, 1 of Feb of 2021. All elements state and condition	_03-Hangers_SFR_Hanger_Strand system_Hanger 1. (I-1)_18750462	Bridge1	Tie Beam
5	Inspection 01Feb21	Principal Inspection, 1 of Feb of 2021. All elements state and condition	_03-Hangers_SFR_Hanger_Strand system_Hanger 1. (I-1)_18750466	Bridge1	Tie Beam
6	Inspection 01Feb21	Principal Inspection, 1 of Feb of 2021. All elements state and condition	_03-Hangers_SFR_Hanger_Strand system_Hanger 2. (I-II)_18750358	Bridge1	Tie Beam
7	Inspection 01Feb21	Principal Inspection, 1 of Feb of 2021. All elements state and condition	_03-Hangers_SFR_Hanger_Strand system_Hanger 2. (I-II)_18750380	Bridge1	Tie Beam
8	Inspection 01Feb21	Principal Inspection, 1 of Feb of 2021. All elements state and condition	_03-Hangers_SFR_Hanger_Strand system_Hanger 2. (I-II)_18750386	Bridge1	Tie Beam
9	Inspection 01Feb21	Principal Inspection, 1 of Feb of 2021. All elements state and condition	_03-Hangers_SFR_Hanger_Strand system_Hanger 2. (I-II)_18750408	Bridge1	Tie Beam
10	Inspection 01Feb21	Principal Inspection, 1 of Feb of 2021. All elements state and condition	_03-Hangers_SFR_Hanger_Strand system_Hanger 3. (II-2)_18750360	Bridge1	Tie Beam
11	Inspection 01Feb21	Principal Inspection, 1 of Feb of 2021. All elements state and condition	_03-Hangers_SFR_Hanger_Strand system_Hanger 3. (II-2)_18750378	Bridge1	Tie Beam
12	Inspection 01Feb21	Principal Inspection, 1 of Feb of 2021. All elements state and condition	_03-Hangers_SFR_Hanger_Strand system_Hanger 3. (II-2)_18750388	Bridge1	Tie Beam
13	Inspection 01Feb21	Principal Inspection, 1 of Feb of 2021. All elements state and condition	_03-Hangers_SFR_Hanger_Strand system_Hanger 3. (II-2)_18750406	Bridge1	Tie Beam
14	Inspection 01Feb21	Principal Inspection, 1 of Feb of 2021. All elements state and condition	_03-Hangers_SFR_Hanger_Strand system_Hanger 4. (2-II)_18750362	Bridge1	Tie Beam

Figure 35 – Excel Workbook opened in Cellfie

Generated Axioms	Rule:
Cellfie generates 8 axioms:	Individual: @A*
Individual: Inspection01Feb21	Types: Procedure, InspectionActivity
Individual: Inspection06Mar21	Facts: Description @B* (xsd:string)
Inspection01Feb21 Type InspectionActivity	
Inspection01Feb21 Type Procedure	
Inspection06Mar21 Type InspectionActivity	
Inspection06Mar21 Type Procedure	
Inspection01Feb21 Description	
"PrincipalInspection,1OfFebOf2021.AllElementsStateAndCondition"	
Inspection06Mar21 Description	
"RoutineInspection,6OfMarchOf2021.FocusOnHangerStateAndCondition"	

Figure 36 – Example of generated axioms from a rule in Cellfie

The set of rules utilized is available in the project repository (see Chapter 7). These rules are used to import information about the bridge, its elements, inspections, and inspections data. Figure 37 shows two examples, for Observations and Risk Analysis.

Rule:	Rule:
Individual: @L*	Individual: @O*
Types: Observation	Types: 'Risk Analysis Result'
Facts: 'used procedure' @A*,	Facts: conditionIndicator @H* (xsd:integer),
'observed property' @M*,	conditionState @I* (xsd:string),
'has feature of interest' @C*,	nextInspectionType @J* (xsd:string),
'result time' @G* (xsd:dateTime),	nextInspectionDeadline @K* (xsd:dateTime),
'has simple result' @H* (xsd:integer),	photoDetailURL @P* (xsd:string)
'has result' @O*	

Figure 37 – Rule definition for Observations (left) and Risk Analysis Results (right) import


EurOTL defines the connection between Physical Elements and *IfcGeometricRepresentationItem* as a connection point between element information and ifcOWL, by using the "hasRepresentation" object property. This connection is already instantiated when importing the element information (so each element contains information about its representation on the model). However, the *ifcGeometricRepresentationItem* is a small component of ifcOWL with no direct connection to the ifcElement's GUID.



A set of relations is needed to connect the *ifcGeometricRepresentationItem* to the *IfcGloballyUniqueId* value. Dummy data with values based on the initial ifcOWL model and additional import rules were created for this purpose. The new set of rules instantiates these relations for each of the elements, thus ensuring the connection between the elements and BIM model.

In the end, 12400 new axioms are added to the ontology with individuals' information. One bridge and 306 bridge elements (such as Tie Beams, Bracings, Arches), each one connected to a valid GUID. Information about two different inspections and the respective Risk and Condition Results were also imported to the ontology (Figure 38).

Once the knowledge was fully imported to the ontology on Protégé, the ontology was transferred to a GraphDB repository and made available through the CoDEC API.

<div> <div>obs227 — http://www.codec-project.eu/def#obs227</div> <div>Annotations Usage</div> <div>Annotations: obs227</div> <div> <div>Annotations +</div> <div>'has feature of interest'</div> <div> <div></div> <div>Floor_Steel_24_19340954</div> </div> <div>'has result'</div> <div> <div></div> <div>rar227</div> </div> <div>'has simple result' [type: xsd:integer]</div> <div>2</div> <div>'observed property'</div> <div> <div></div> <div>RiskAndConditionData</div> </div> <div>'result time' [type: xsd:dateTime]</div> <div>2021-02-01T00_00_00</div> <div>'used procedure'</div> <div> <div></div> <div>Inspection01Feb21</div> </div> </div> </div>	<div> <div>rar227 — http://www.codec-project.eu/def#rar227</div> <div>Annotations Usage</div> <div>Annotations: rar227</div> <div> <div>Annotations +</div> <div>conditionIndicator [type: xsd:integer]</div> <div>2</div> <div>conditionState [type: xsd:string]</div> <div>Regular</div> <div>nextInspectionDeadline [type: xsd:dateTime]</div> <div>2021-07-31T00_00_00</div> <div>nextInspectionType [type: xsd:string]</div> <div>Principal</div> <div>photoDetailURL [type: xsd:string]</div> <div>  </div> </div> </div>
--	--

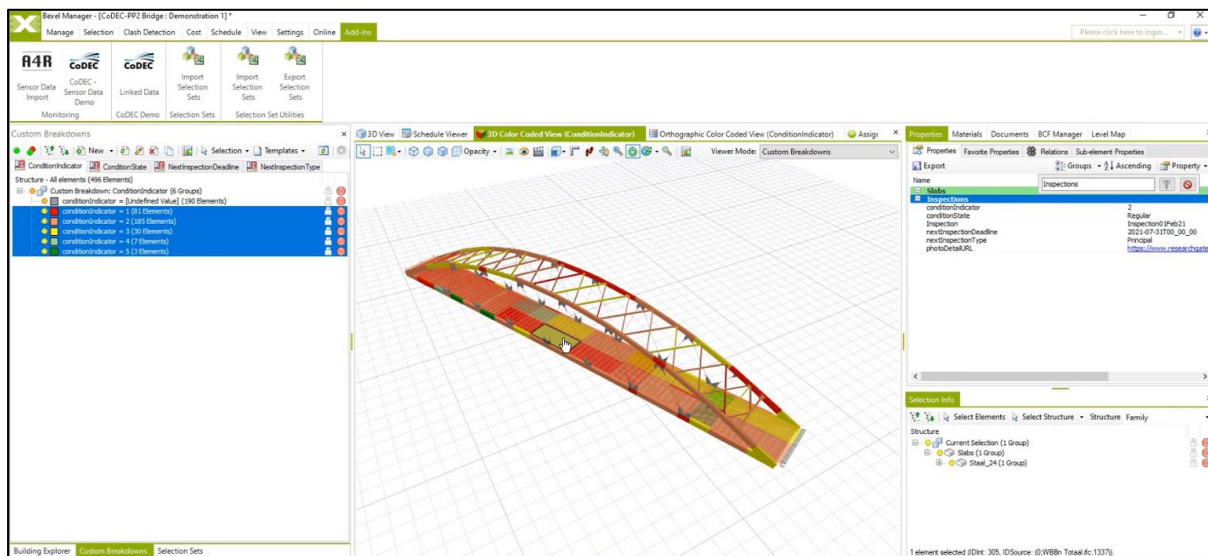
**Figure 38 – Ontology Individuals: Observation “obs227” (left) and Risk Analysis Result “rar227” (right)**

### 5.3 Outcomes of Pilot Project 2

The outcomes of Pilot Project 2, enable visualisation and risk analysis of condition data directly in the BIM model, using the Add-in developed for this pilot project. After opening the BIM model in Bexel Manager, all the typical functionality of the tool is available. Once the linked data add-in is installed, the user can get the list of inspections associated with the structure and get the risk and condition data associated with that inspection.

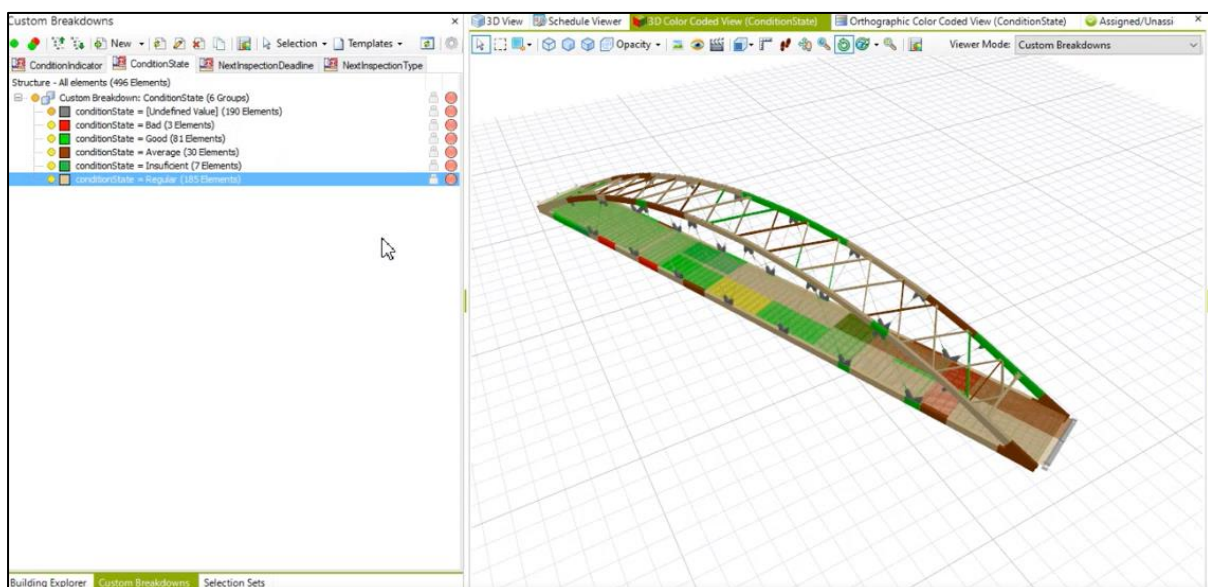
Figure 39 shows the use of the Custom Breakdown (CBS) functionality to assign different colours to the elements of the structure, according to the condition level determined for each element in the selected inspection.



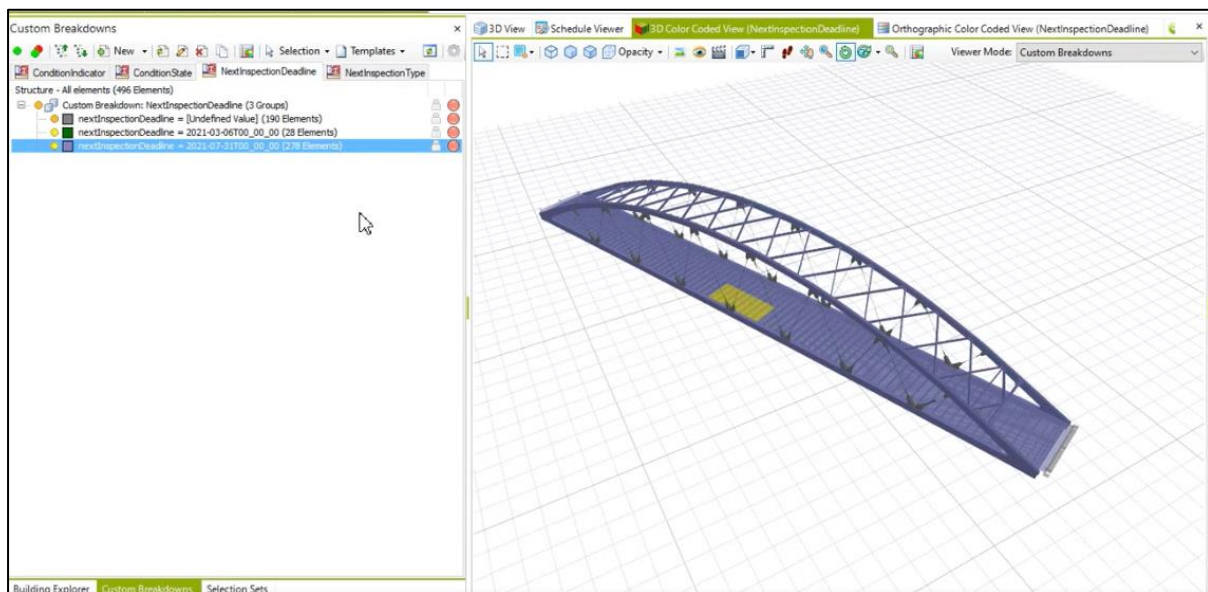


**Figure 39 – BIM bridge model coloured by condition indicator**

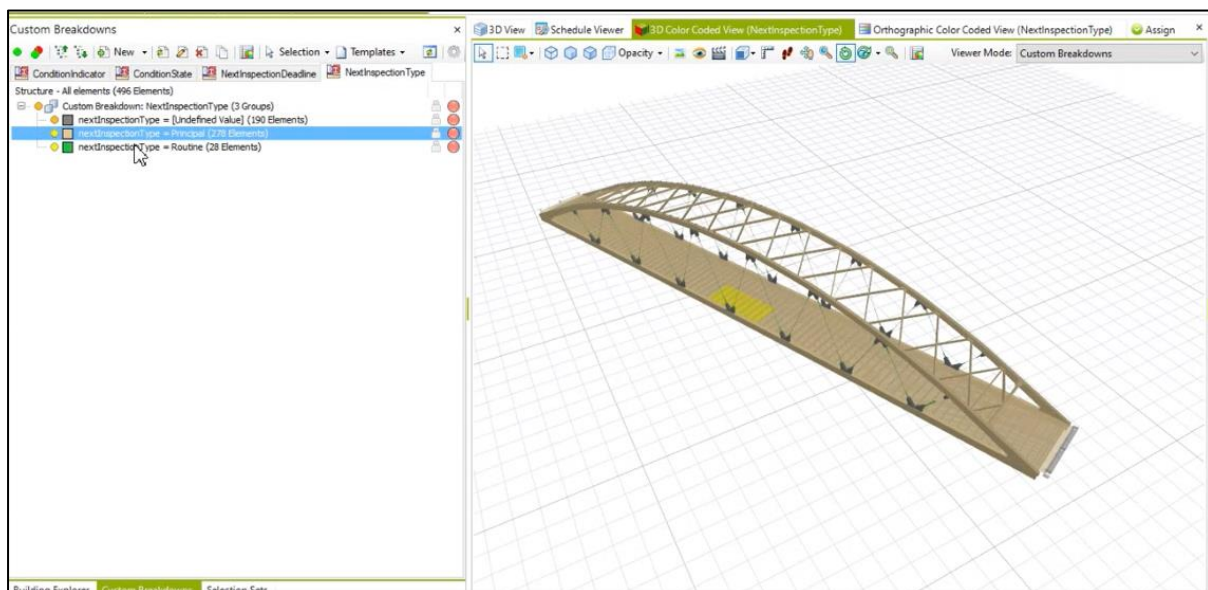
Figure 40, Figure 41 and Figure 42 illustrate the use of the same functionality for other values associated with that inspection, namely, the qualitative assessment of the condition state of the elements, the deadline for the next inspection and the type of the next inspection. Note that using the Custom Breakdown functionality of Bexel Manager, the user has total flexibility to edit the colour scheme and define rules and limits to assign colours according to the values associated to each element of the structure.



**Figure 40 – BIM bridge model coloured by qualitative condition state**

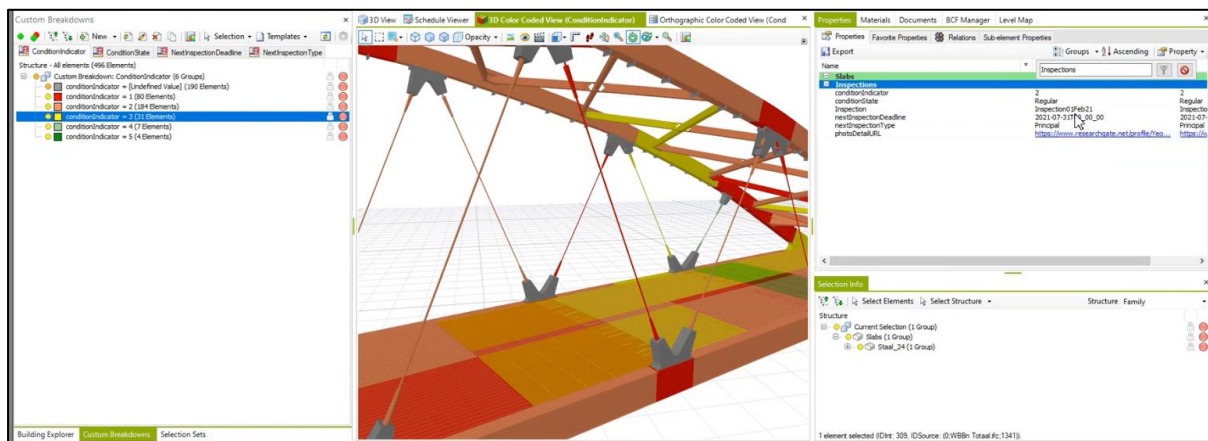


**Figure 41 – BIM bridge model coloured by next inspection deadline**



**Figure 42 – BIM bridge model coloured by next inspection type**

Finally, as illustrated in Figure 43, it is possible to select an element (or set of elements) and view all the details of the attributes associated with those elements. In the particular case of images (especially those related to photographs), a link is provided to the image or folder of images that were captured in the inspection selected for each element of the structure. Note that this information is included in the risk and condition data ontology extension and there may be several images associated with each structure element in an inspection.



**Figure 43 – Element inspection details, including links do images**

## 5.4 Recommendations

This pilot project has demonstrated the connection between data stored in a Linked Data Platform and BIM models, using the CoDEC API to provide access to that data. In this particular case, it was possible to successfully demonstrate the integration of risk and condition data from a bridge inspection. The visualisation of risk and condition data in a BIM model provides quick and effective analysis through the use of colour schemes in the BIM model.

One of the main difficulties in this approach is related with the connection between the risk and condition data and the BIM model. Typically, these data are captured and recorded in external systems, with no connection to the BIM models. The approach adopted in this pilot project involved the use of the ifcOWL ontology, thus allowing the connection between the BIM elements and the data sources (with this approach, we can use any field to make the connection, when importing to the ontology, as long as there is one or more fields that allow making the association).

The project has tested the conversion of the BIM model into ifcOWL. However, since it is not possible to filter the elements during the conversion, the ontology generated was quite heavy and complex, and difficult to manipulate. Although this problem does not arise in simpler models, it is a major limitation in complex and/or detailed models. Thus, one of the main recommendations would be that BIM tools should provide export functionalities to ifcOWL, with advanced filter mechanisms, both at the level of the elements to be exported and the details of the properties to be included in the export.

Finally, to take advantage of Bexel Manager visualization features, risk and condition data is imported into the BIM model through the CoDEC API. The main limitation of this solution is the lack of separation of concerns between information managed by the BIM model and those from the linked data environment, requiring that the BIM model is continuously updated to provide a visualization feature.

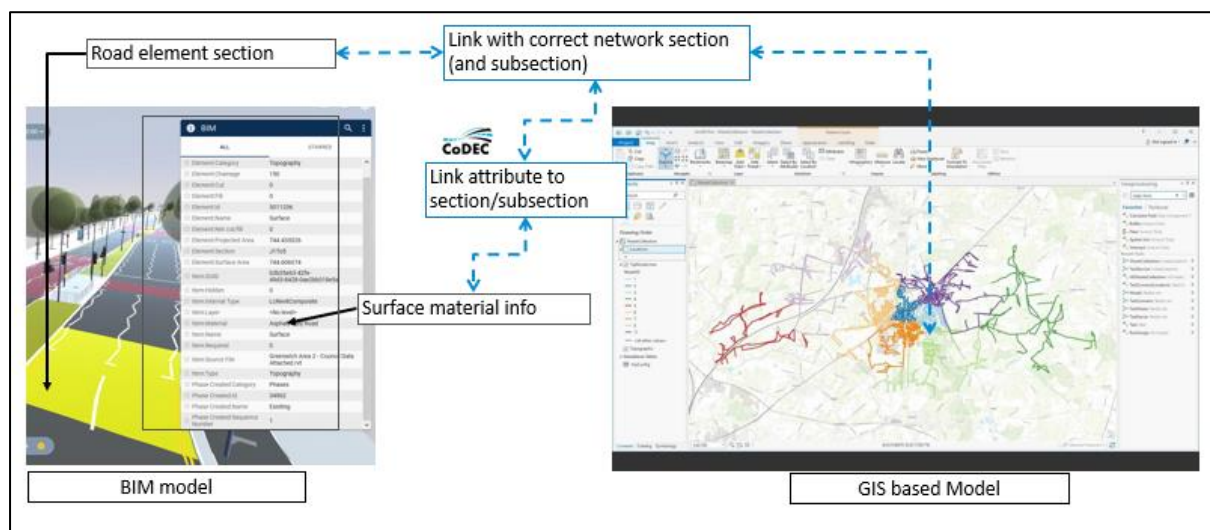


## 6 Pilot Project 3: Enhancing legacy data by linking BIM models to GIS systems

### 6.1 Concept and objective

Whilst BIM models are often created for the design/construct phase, highways are primarily managed during the operational phase using GIS-based Asset Management Systems (AMS). Likewise, LiDAR scans – increasingly being used to gather physical data about highways assets – are often translated into BIM models rather than GIS-based information. In all such cases, BIM models often hold information useful for asset management, but which is not made available in the GIS-based AMS where these assets are managed. In other words, BIM models represent a new source of data which can be used to enrich (and/or complement) the data held within legacy systems – a particular focus of the CoDEC project.

This pilot project focuses on the carriageway pavement as the key asset. Although some of the methods developed are specific to the challenges of linking pavement asset data, the general approach is more widely applicable to other non-linear highways assets. Pilot Project 3 aimed to address this issue by providing methods to link data between the BIM and GIS-based environments (see Figure 44). The concept was agreed with the FTIA (Finnish NRA) who were the implementation partner for this pilot. However, the BIM model was provided by The Smart Mobility Living Lab (SMLL), London and the asset data was provided by the Royal Borough of Greenwich.



**Figure 44 - Pilot Project 3 Concept**

Pilot Project focuses on two specific cases (see Figure 45):

- **Use case 1:** This use case falls under the category of ‘inventory data’ that an asset manager would require mostly for one-time assessments generally in the initial stages of construction/ maintenance. For the case study we will focus on the longitudinal gradient of the pavement. However, the 3D information we use is less important than the method we use to calculate/extract the information. Lidar point cloud scans available from TRL’s SMLL lab have been used as input for gradient calculation. Nevertheless, the use case can also be extended to other data points that need to be inventoried at different stages of the life cycle of the asset.
- **Use case 2:** This use case focusses on asset data points that change during the lifecycle of the asset and require regular monitoring by asset managers. We refer to this as road construction data (as depicted in Figure below). For this use case, to demonstrate the CoDEC framework we chose to include the following information about the road surface: Position of layer (top, bottom etc), material type, thickness of layers.

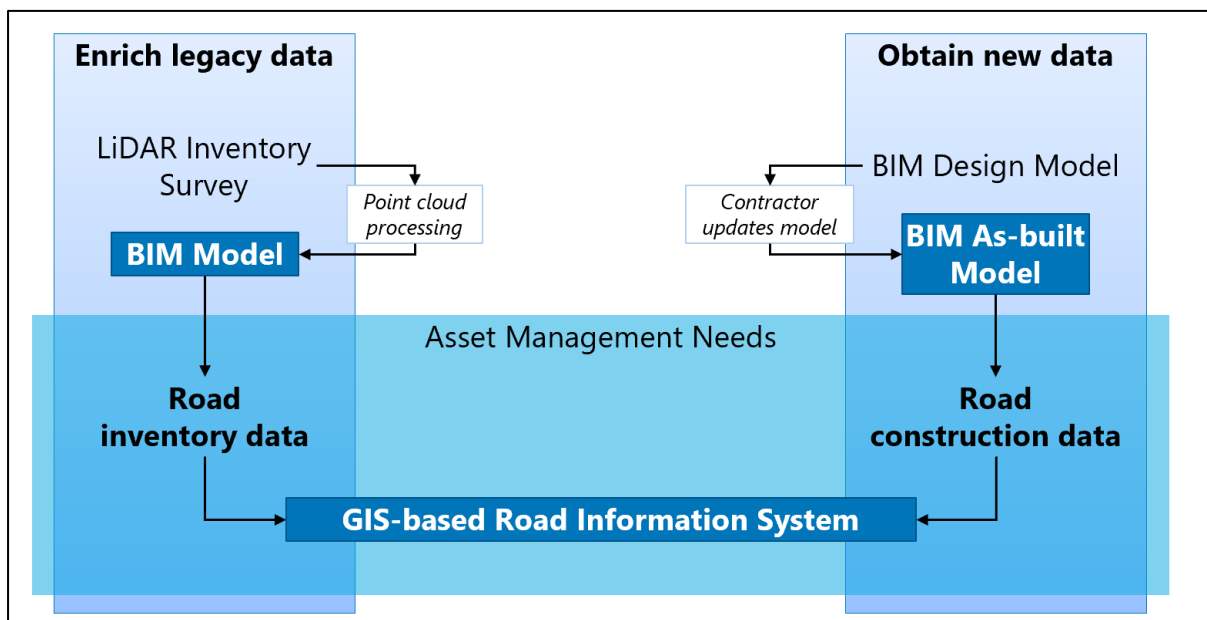
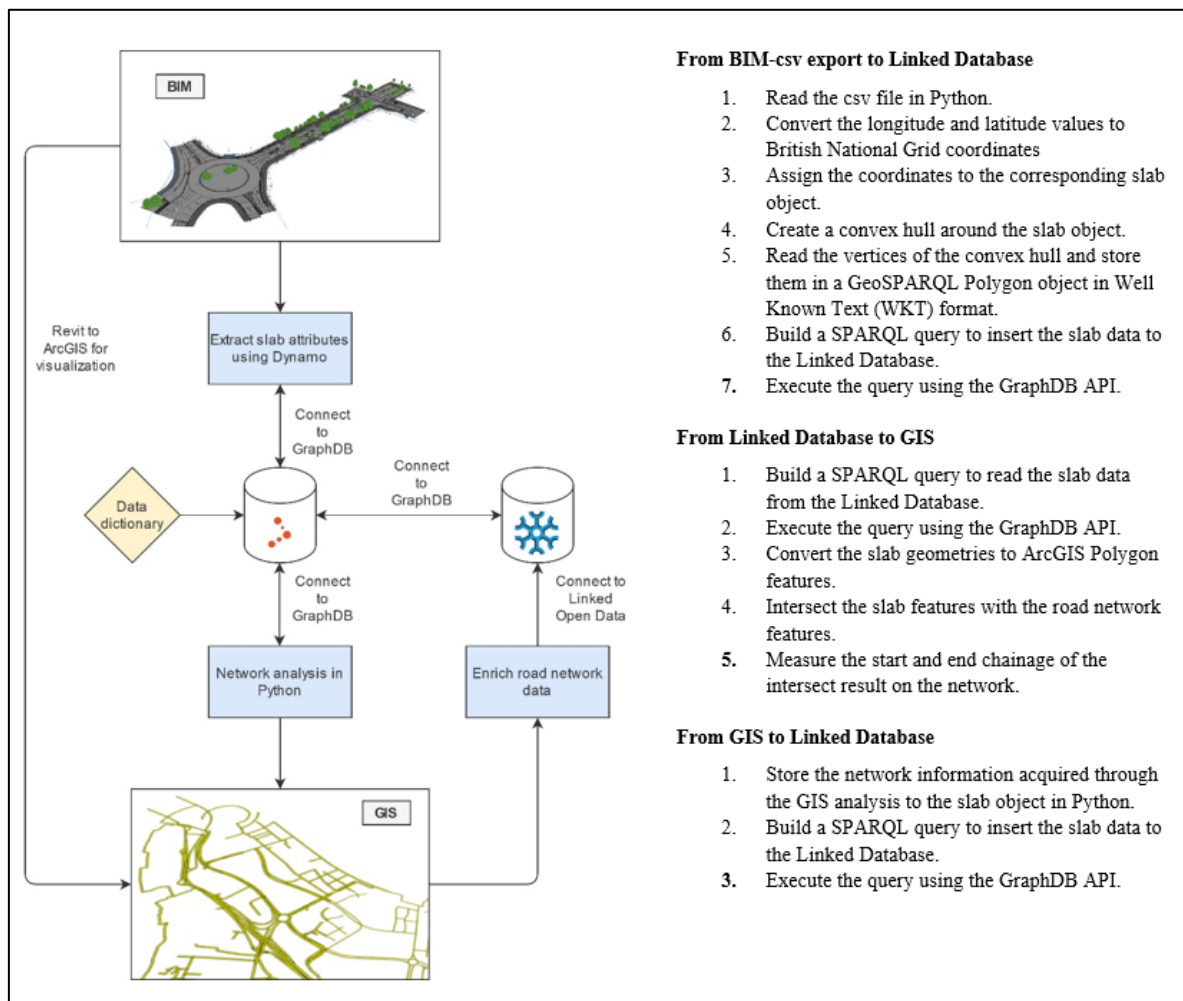


Figure 45 - Use cases for PP3

## 6.2 Methodology and implementation

The CoDEC ontology is being used to provide a framework for organizing and linking the relevant data. The linking methodology is summarized in Figure 46 below. The pilot project has provided a practical opportunity to test the content and structure of the ontology (which is based on the data dictionary) – this has directly resulted in improvements and additions to the ontology.



**Figure 46 - Steps involved in linking data between the BIM models and the GIS using the CoDEC-developed tools and schema**

Below, we firstly describe the BIM model, then the various software and tools used in each step, followed by a description of each step and key lessons learnt/ challenges encountered and mitigated.

### 6.2.1 The BIM model

The BIM model of the Smart Mobility Living Lab, London (SMLL) was used for the Pilot Project. The model covered the entirety of the SMLL routes in the Royal Borough of Greenwich and the Queen Elizabeth Olympic Park both in London. The following software and tools were used in association with the model.

- Revit: 

Autodesk Revit is a commonly used BIM modelling tool, allowing users to create digital assets for design and construction projects with emphasis on building construction. It was selected for this project due to the SMLL BIM model being created within Revit and the

software having an established mechanism (Dynamo) for interacting with the model.




- Dynamo:

Dynamo is an open-source visual programming language designed for Revit and comes as standard as a plugin to the main Revit program. It facilitates users to automate and iterate processes, access metadata within the model and import and extract information from Revit.

- Python: 

Python is a high-level general-purpose programming language and was used in this process in a few ways. Dynamo also enables a python API to Revit which enables more complex dynamo programs to be written. Due to its general-purpose nature and extensive resource and library of packages available, it was used to process the extracted data and upload it to the database.

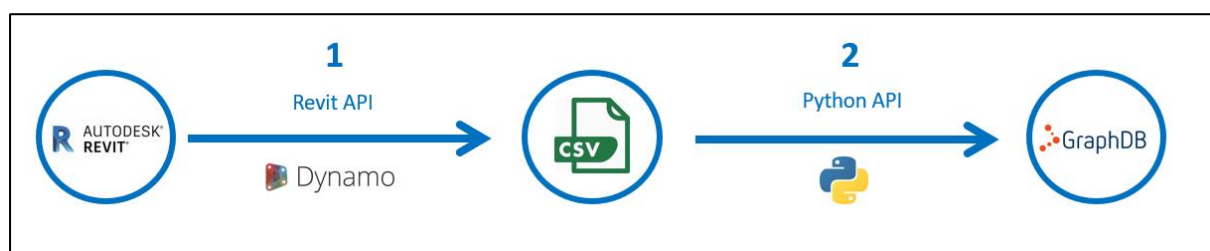
- GraphDB: 

Graph DB is a cloud-based enterprise-ready semantic graph database used in the project to host linked data and demonstrate the structure of the CoDEC ontology. The database can be queried via the CoDEC API using a SPARQL request, making the information accessible to the GIS software.

- ArcGIS Pro: 

ArcGIS Pro is a commercial GIS application developed by Esri. ArcGIS is the industry standard for working with desktop GIS. In this pilot project, ArcGIS is used for data visualization, data management and spatial analysis. To automate the spatial analysis, we used the python package ArcPy, which is provided with the ArcGIS Pro installation. ArcGIS Pro uses Python 3.

## 6.2.2 The process (BIM to linked data)



**Figure 47 - Diagram of the BIM to linked data high level process**

The two-stage process of uploading BIM data to a linked data platform shown in Figure 47 is described in high level below:

1. The data began in a Revit model either inherently as the geometry information or with target data assigned to each element (road slab) as parameters.



- a. To extract geometry information such as gradient from the model, this can be calculated using Dynamo and then exported or assigned as a parameter to the relevant element in the model.
  - b. A Dynamo file and Python module extract the parameters and the bounding coordinates of each of the slabs or centre point of an object to a CSV file.
2. A Python module reads data from the CSV and applies a complex hull algorithm to the coordinates to reduce the number and simplify the polygon before generating a SPARQL query to upload all the information into the ontology within the database GraphDB.

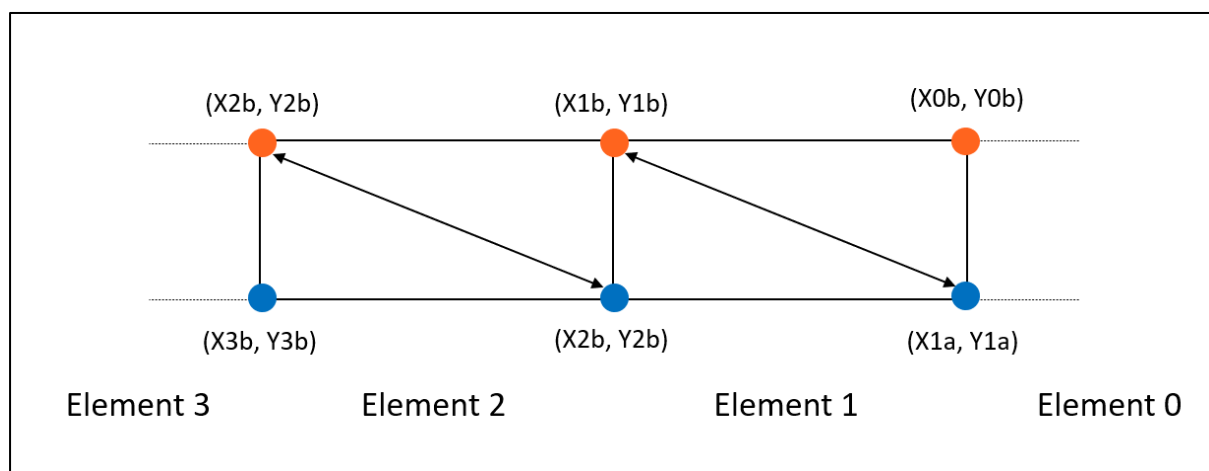
### 6.2.3 Stage 1 – Revit to CSV

#### 6.2.3.1 Dynamo – Calculating and extracting geometry information

A BIM model is a 3D digital representation of the assets (in this case the road and street furniture). Therefore, it inherently holds geometry attributes about the assets which can be obtained through calculation. One of these attributes is road surface gradient. For proof of concept purposes a gradient value was calculated for each of the road slabs and reassigned as a parameter to the respective slab element.

The road surface was divided into 10m sections allowing the gradient to be calculated for each road slab using a Dynamo script. The road surface is constructed using a mesh topology (i.e. it has a theoretical zero thickness), the script extracts all the mesh points of the slabs using the a Revit API built into Dynamo as a 3 dimensional list of points (i.e. 2 nested lists). By looking at the X and Y coordinates it is possible to calculate the longest and shortest distances from the model datum for each slab. As the slabs are rectangular this generates the start and end point for the longest distance across the slab, in this case the diagonal. A screenshot of the dynamo script can be seen in Appendix 2.

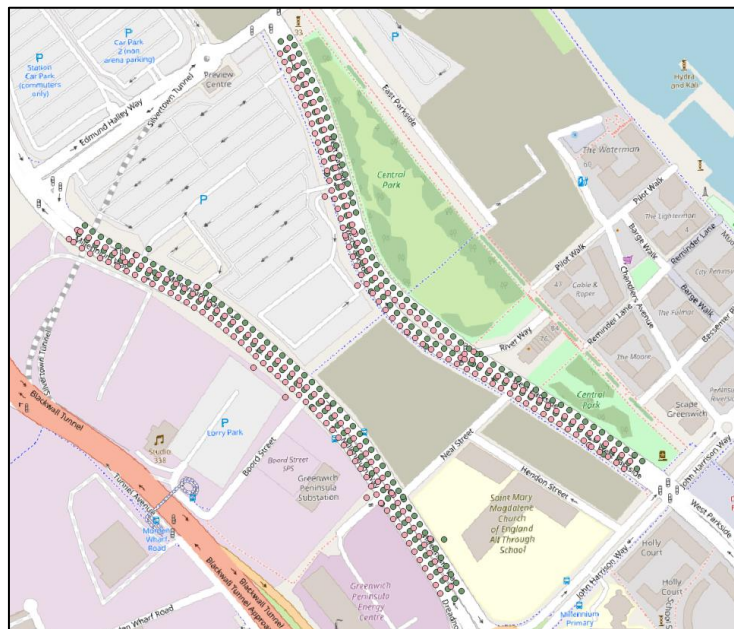
As depicted in Figure 48, the diagonal provides a point on the back edge and point on the front edge, now knowing these two points, the script then calculates the elevation difference which can be shown as a percentage of the horizontal length also known as gradient.



**Figure 48 - Diagram showing the diagonal points of each slab**

It should be noted that ideally to calculate the gradient of each slab you would want to find the average elevation of the front and rear of the slab and calculate the elevation difference between them as a percentage of the longitudinal distance of the slab. This is also true for the crossfall of the road (horizontal gradient) which would be calculated using the average elevation of edges.

As this activity was conducted for proof-of-concept purposes, with the primary focus being the linking of data between BIM and GIS, the diagonal gradient was deemed acceptable for use within this project, especially as it produced a consistent result for all the slabs (seen in Figure 48). However, further work could be conducted to achieve a better definition of the road gradient, as mentioned above, truly in terms of longitudinal and crossfall (Figure 49).



**Figure 49 - The plotted points used to calculate the gradient for each slab**

### 6.2.3.2 Dynamo - BIM parameters to CSV

To standardise the process for extraction, all geometry attributes calculated were reassigned as parameters into the model using a module within Dynamo. For this to work the parameter must have already been created which is done using the standard Revit tools. This then outlines a clear pipeline for structuring the data within the BIM model to prepare it for linked data. Dynamo was used again to extract the metadata from the model for selected objects, this data is attached to element IDs.

Selecting the objects is done in the 3D-view in the Revit software window; the model is filtered based on object type; in this example “topology” was used to filter for the road surfaces only. The higher the granularity of classification of the model in terms of “families” or object types within Revit, the more flexibility a user has for filtering and selecting objects.

All the mesh points and parameters can be separately queried using the standard Dynamo modules for selected objects. A specific parameter can be retrieved from the module using the element ID and a string variable that specifies the parameter name. If multiple elements (e.g., road slabs) have been selected, this outputs a list of values. A similar methodology is used to extract all the mesh points for the slab using only element ID as the input.

A Python module within Dynamo is used to then filter and structure the data to enable a smooth export of the data into a CSV. This Python module is shown below in Figure 50, to export the parameters and mesh points against the element ID, the information needs to be structured into a single 2D array. Additionally, the Revit Python API is used to filter only the boundary points for each of the slabs, removing any internal points. Once this data has been structured, the new array is exported to CSV using the standard Dynamo module.

```
from scipy.spatial import ConvexHull
import numpy as np
import csv
from road_classes import Slab, PavementLayer
import config, access_graphdb

def read_csv_pavement(pathtocsv):
    # Initiate pavement layer dict
    pavement_layer_dict = {}

    # Open and read csv file
    with open(pathtocsv) as csvfile:
        csvreader = csv.reader(csvfile, delimiter=',')
        for row in csvreader:
            slab_id = row[0].split("Element ID : ")[1]

            layer_id_1 = f"slab id: {slab_id} 1"
            layer_designation_1 = row[1].split("Layer Designation 1 : ")[1]
            layer_material_1 = row[2].split("Layer Material 1 : ")[1]
            layer_thickness_1 = row[3].split("Layer Thickness 1 : ")[1]

            layer_1 = PavementLayer(slab_id, layer_id_1, layer_designation_1, layer_material_1, layer_thickness_1)
            pavement_layer_dict[layer_id_1] = layer_1

            layer_id_2 = f"slab id: {slab_id} 2"
            layer_designation_2 = row[4].split("Layer Designation 2 : ")[1]
            layer_material_2 = row[5].split("Layer Material 2 : ")[1]
            layer_thickness_2 = row[6].split("Layer Thickness 2 : ")[1]
```

**Figure 50 - Python module in Dynamo for data filtering and structuring**

## 6.2.4 Stage 2 – CSV to linked data

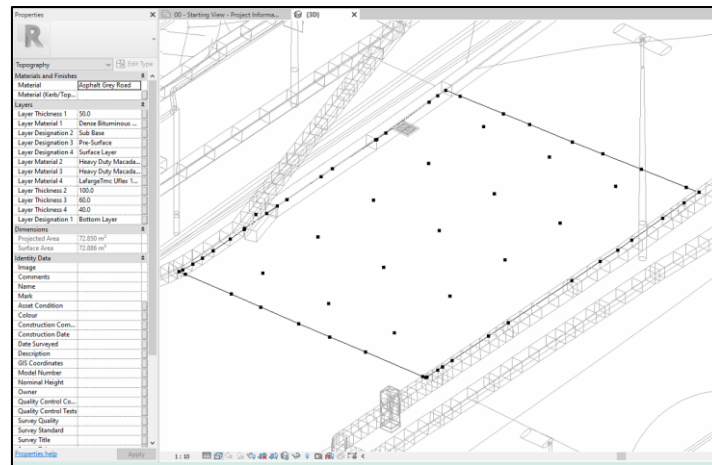
### 6.2.4.1 Python Module

A Python module developed on the project was used to read in CSV data and generate a SPARQL query to POST the information into the database. Two POST requests are performed for each of the slab elements and the data is iterated through. The first is for the geometry (i.e., the points detailing the perimeter of the slab). When building the query for the geometry of the slab, there are too many data points in some cases which causes the string to exceed the character limit for a query and outputs an error. To resolve this issue, before the data is built into the query, it is simplified to shorten the string and keep the data as lightweight as possible. This was achieved using a convex Hull method outlined in more detail below. More detail on this stage is provided in the BIM to linked data section.

### 6.2.4.2 Simplification of the Mesh – Convex Hull

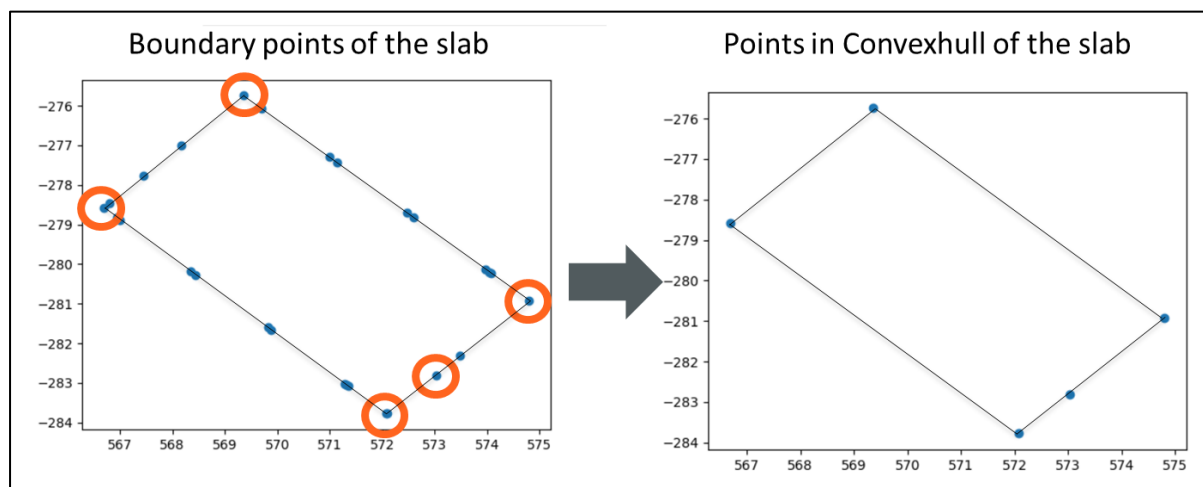
A convex hull algorithm is used to simplify the mesh which contains a high number of points to define the road surface, as seen in Figure 51. Only the boundary points are extracted from Revit into the CSV, however there are still a high number of points (40+) along the perimeter.

Convex Hull, or Convex Envelope, calculates the shortest convex or enclosure that encapsulates all the points. A convex is defined as a shape in which a theoretical line can be drawn between any two points inside the shape and the line remains inside the shape also.



**Figure 51 - Typical number of mesh points for a road slab**

When this algorithm is applied to a perimeter of points for a rectangle this typically find the 4 vertices only. Figure 52 shows the dramatic benefits of using this method on the slabs in the BIM model, simplifying ~30 points down to only 5.



**Figure 52 - Real example from the model of the effects from applying a Convex Hull algorithm**

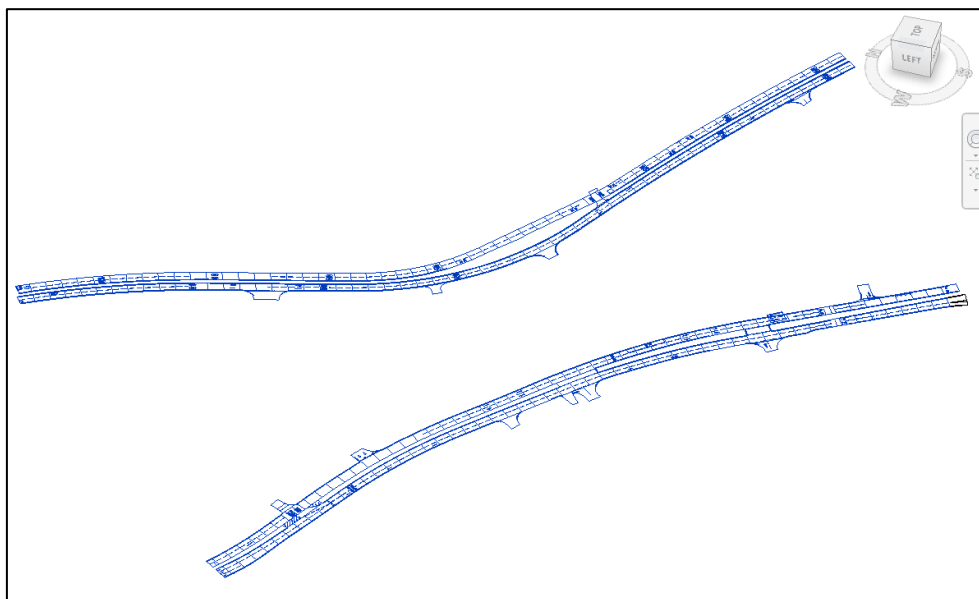
The algorithm used is part of the Python Scipy package and typically imported using:

```
from scipy.spatial import ConvexHull, convex_hull_plot_2d
```

The module computes the algorithm using the Qhull library which is used in QuickHull algorithm developed by Barber and Dobkin and first published in 1995. An iterative process that adds a point one at a time to an intermediate hull before removing all interior points.

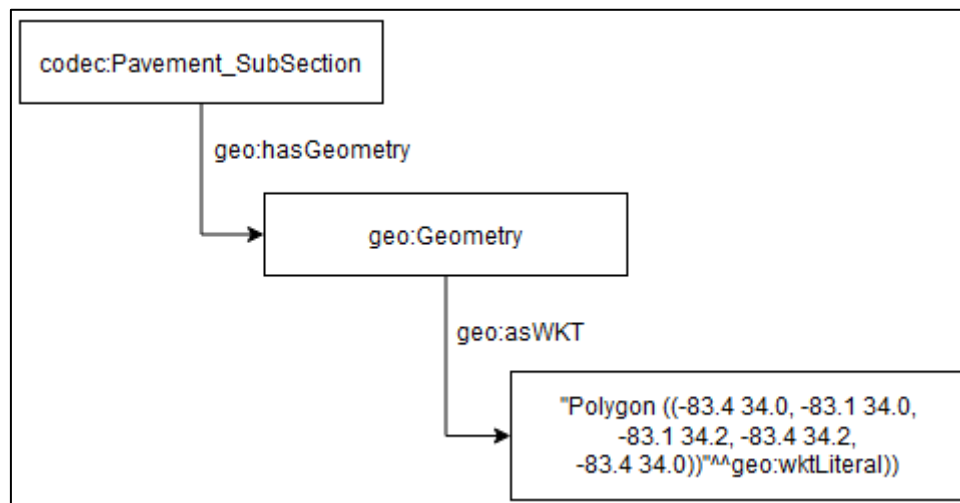
### 6.2.4.3 Importance of road slabs and the breaking up of larger roads

Figure 53 shows the road section divided into 10m sections. This proved crucial to developing this workflow. Chainage is the measure of longitudinal distance along a road and is measured in increments of 10m. This was also the case for the slabs and it provided enough granularity to assign and extract relevant road information from the model. If the road was defined as one complete model section the entire length of the road would have the same parameters assigned to its entire length, more complex geometry (potentially inhibiting the building of a SPARQL query) and make the linear referencing more difficult if it likely included overlapping network sections.



**Figure 53 - Wireframe of BIM model road surface section**

The first challenge in the linear referencing method was to store this pavement geometry in the CoDEC linked data repository. The European Road OTL uses the official GeoSPARQL ontology from the Open Geospatial Consortium to represent geospatial location and relationships to objects (assets). Within the GeoSPARQL standard the property *geo:hasGeometry* is used to link a feature with a geometry that represents its spatial extent. This geometry class is defined as *geo:Geometry* and can contain multiple spatial properties. In this pilot project we use the *geo:asWKT* property, which is defined to link a geometry with its WKT serialization, to store the convex hull coordinate information from the pavement subsection as a polygon (Figure 54).



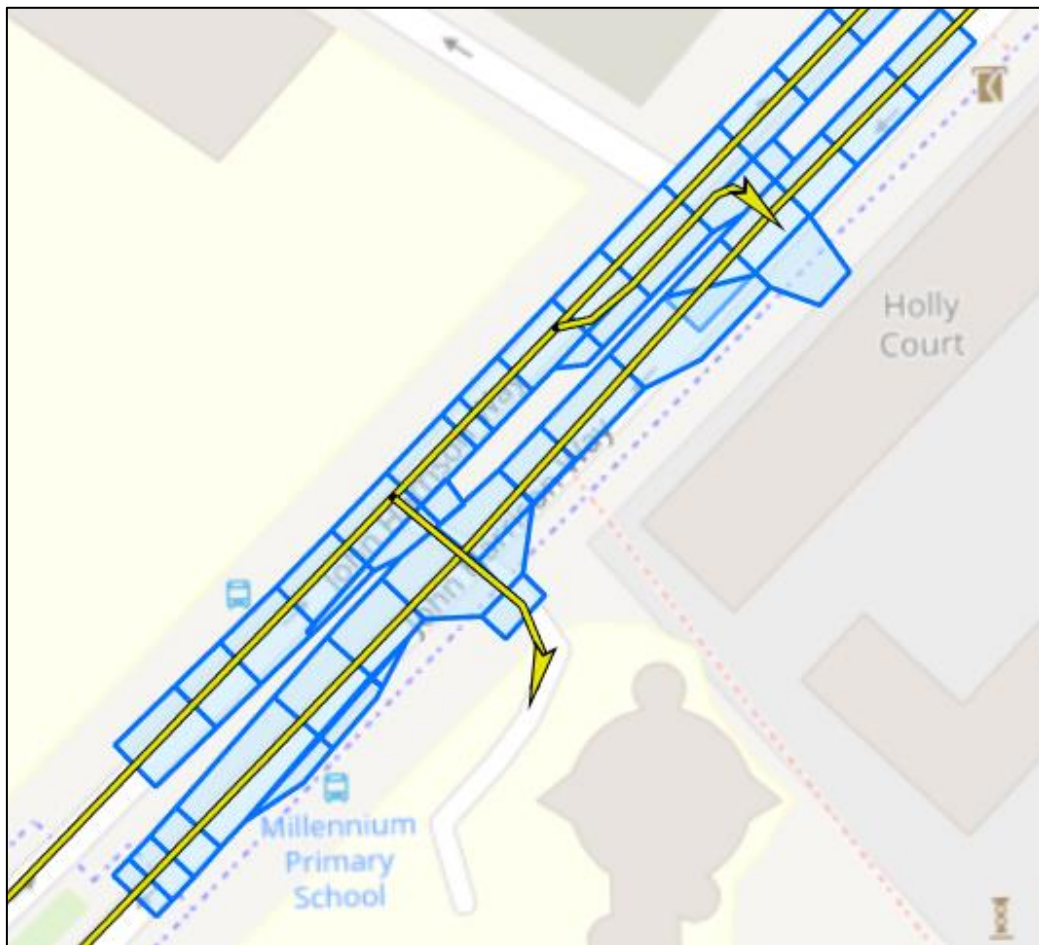
**Figure 54 – Pavement Geometry in CoDEC Ontology**

The OGC standard definition requires that the coordinates of the exterior linear ring of a polygon are defined in a counterclockwise direction. This is achieved by sorting the individual points within the convex hull around the pavement subsection using a python script. The next step is to insert all points in the geo:wktLiteral string and using all information in the slab to build a SPARQL query. Using the CoDEC API, each slab is uploaded to the repository through a POST request, containing the query as text. The query uses the INSERT DATA operation to insert all triples that store information of one pavement subsection to the database, using the CoDEC data dictionary and European Road OTL to define the classes and properties. In addition, all triples containing information about the pavement layers are inserted as well.

### 6.2.5 From linked database to GIS

The next challenge is to retrieve the information about the slabs from the repository and store them in a GIS-readable format. Again, we use a SPARQL query through the CoDEC API, but now we use a GET request, to store the information as a JSON within a python script. Asset management in GIS is usually done with commercial software package ArcGIS, which is the industry standard. That is why we use the python package arcpy to convert the JSON to a feature class in a geodatabase. This is done by creating an empty polygon dataset and loading in the individual slabs from the JSON one by one, while using an arcpy function that converts a WKT geometry to a Geometry object readably by ArcGIS. It is now possible to view the pavement slabs on a map and visually compare the data with the road network (Figure 55).



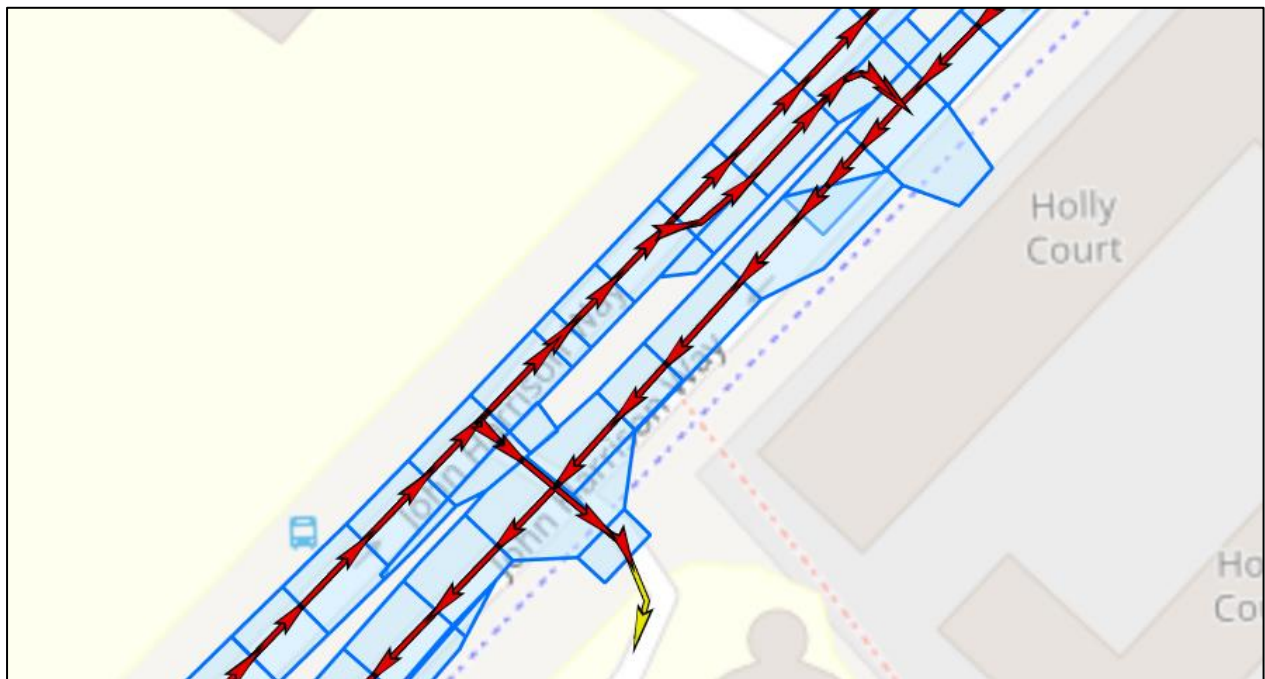


**Figure 55 – Pavement Slabs and Road Network**

#### **6.2.5.1 Linear referencing within the GIS environment**

The biggest question regarding the linear referencing method is how to assign parameters from the detailed geometric representation in a 3D BIM model to the 2D line representation of the road network. The network model is simple by design because it is used to provide insights of the roads in a quick and clear overview. The BIM model is used to provide a more realistic representation of the roads. Converting these complex geometries to simple lines will always result in losing data. In this pilot project, we have chosen to determine which slabs are part of the road network by intersecting them with the network lines. As a result, the slabs that do not cross the line, will be excluded from the analysis. The placement of the lines in the network is important because it can determine if data from slab A or from slab B will be assigned to this network segment (Figure 56). After the analysis, each intersection between network line and slab polygon will be stored in a new line dataset that contains information from both data sources.



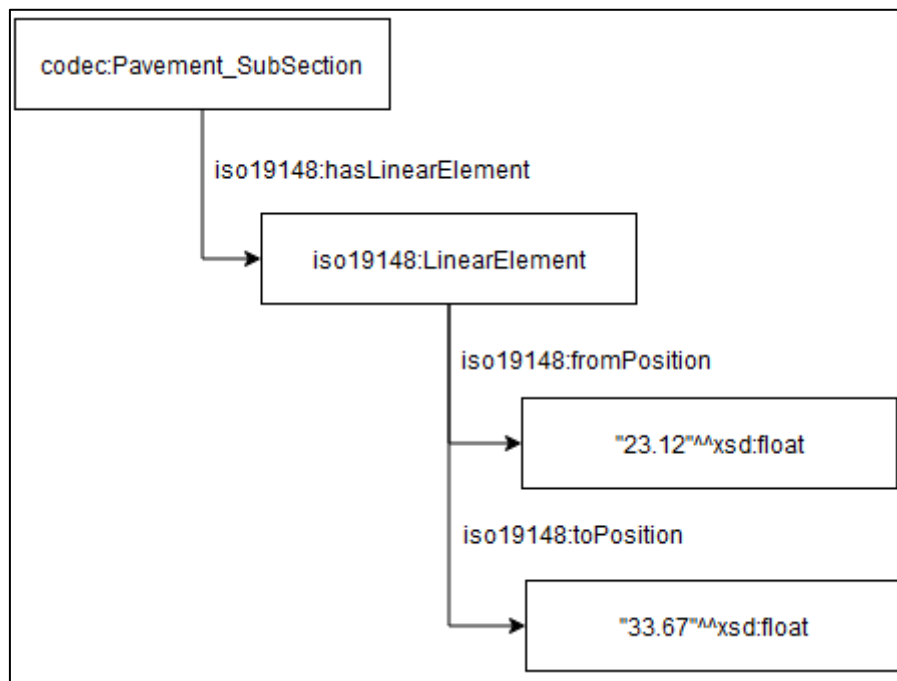


**Figure 56 – Intersections between Slabs and Network lines**

#### **6.2.6 From GIS to Linked data**

The positions of the slabs are stored as linked data using the ISO 19148 Linear Referencing ontology, used in the European Road OTL. This ontology provides means for locating objects (assets) along elements of a network, alignments, or other linear elements. In our case the linear element is an individual road object within the road network (Figure 57). For each object, the start and end position on the network is determined, by measuring the start and end distance relative to the start of the entire polyline, using the `arcpy.measureOnLine` geometry method. These measurements are stored as IEEE 32-bit floating-point (`xsd:float`) values.

Finally, these linear elements are uploaded back into the CoDEC repository through a POST request using the CoDEC API. We have now successfully enriched the road network model with information from the BIM model using linked data technology and international standards.



**Figure 57 – Linear Referencing in CoDEC Ontology**

### 6.2.7 Lessons learnt/ challenges faced

The developed methodology provides a proof of concept to demonstrate the conversion of BIM to linked data and from linked data to GIS. There are several technical improvement areas that would provide benefit should this be developed further:

1. Single stage upload and query process:

Removing the CSV file from the workflow to avoid a middle format which opens the process up to errors. Ideally a single script that could query the model and perform the calculations currently performed in Dynamo before uploading the results to GraphDB. Revit supports an external C sharp and C++ API that could be used to facilitate this.

2. Platform agnostic – how does this work without dynamo?

Building on point 1, currently Revit and Dynamo are key contributors to the workflow and any parameter types can be created with values assigned using this software. However, ideally this process should be platform agnostic and applied regardless of the choice of software to all models developed. An open modelling format is IFC which could be used to standardise the process and make it cross platform specific.

3. More accurate calculations of geometry data:

As mentioned in the Stage 1 section, the calculation of the gradient data could be improved, or an alternative geometric attribute identified that could be extracted to truly show the benefits of this process.

#### 4. The linear referencing method in GIS:

The challenging part in linear referencing is to correctly identify the position of the pavement polygon on the network line. To achieve this both datasets need to be perfectly aligned, which is rarely the case. The results of the analysis will be more accurate if the pavement sections are cut up into smaller segments.

### 6.3 Outcomes of Pilot Project 3

#### 6.3.1 *What Pilot Project 3 has achieved?*

Pilot Project 3 has:

- Successfully achieved the objective of accurately linking BIM object data to GIS network objects
- Developed a method for accurately mapping location of polygonal objects in BIM to a linear alignment in GIS
- Developed methods for calculating geometric properties of BIM objects and making them available for linking to GIS
- Demonstrated the use of the CoDEC ontology for linking data between BIM and GIS

#### 6.3.2 *Implications for BIM model design:*

The project has thrown up some implications for the design of BIM models to facilitate such a linking approach in future. In brief, these are:

- Pavement objects should be modelled in small segments

It is impossible to say what the optimum size of such segments should be, since there are many factors influencing the decision – for example, the granularity of the data available to be attached to each segment, the road layout (curvature, length between junctions, complexity...), and maybe even constraints on model size.

There may be an argument to say that the pavement can be modelled as a single continuous object at first, and then segmented when required for a linking process such as we have demonstrated. However, the end result is largely the same.

- The road alignment should be included as an object in BIM models

Including the alignment as an 'object' (perhaps as connected curvilinear elements) should facilitate the referencing of BIM data to GIS data by effectively including the linear referencing system within the BIM model.

- Ensure useful information is parametrized and easily accessible

In the BIM models we had available for our pilot project, 3D geometry data was not directly available as parameters attached to objects – we had to develop methods to calculate/approximate this information. If the model had been designed and

developed in such a way that useful parameters were pre-calculated and assigned to objects, a significant amount of effort would have been saved.

Note that standardisation around BIM (and Asset Information in general) will be a significant enabler for all of the above. We expect that IFC Road will deal with these issues as it evolves.

## 6.4 Recommendations

Although Pilot Project 3 has achieved its stated objectives, to make the method ready for operational use at scale, and applicable to more use cases, we recommend investigating:

- Integration of “near real-time” data within the same process
- Application of the PP3 methodology to point assets: point assets are significantly less complex to deal with than polygonal assets in the context of this project. However, many roadway assets are effectively point assets (signs, gulleys, etc.), and will be included in BIM models. There is no good reason not to include point assets in the BIM->GIS linking process, and it is worth investigating whether there are particular difficulties with doing so, and to formalise a methodology.
- Develop ways to automate as much of the process as possible: due to the nature of PP3, we did not seek to develop a 100% “operationally ready” methodology with a fully-automated workflow. However, in order to apply such methodologies operationally, and at scale, the process would need to be automated as far as is possible (while simultaneously being more integrated in the relevant applications).
- Investigate data connection in the opposite direction (GIS -> BIM): PP3 had a specific remit of making data from BIM available in GIS applications. We therefore approached the problem from the viewpoint of data flowing from BIM to GIS, which generated specific problems. For a full realisation of the principles of linked data, the problem should also be investigated from the GIS->BIM perspective – this is also a critical step to understanding how to build effective digital twins.
- Investigate the implications of lateral pavement delineation (lanes, XSPs...)

## 7 Further outputs associated with this report

CoDEC has developed a number of relevant outputs that form “physical deliverables”, namely the source code of the software scripts, API, and add-ins, that are attached to this deliverable, in a folder named Repository.

Guidance is provided in Appendix 1 on the content of the Repository and how to use it, this includes:

- Accessing and using the CoDEC Ontology
- Use of GraphDB
- The code for the CoDEC API
- Accessing and using Bexel Manager
- The Add-ins developed for Pilot Projects 1, 2 and 3

## 8 Conclusions

The CoDEC project aims to provide a better understanding of how BIM principles could be applied to manage assets during the operational phase. The project has developed tools and procedures to connect asset management and BIM platforms using linked data / semantic web techniques to manage asset information. With the increasing number of ICT systems and, potentially heterogeneous, data sources, the use of semantic web techniques is increasingly relevant to accommodate the complexity of current and future data.

In this WP3 report we have demonstrated the application of the principles of integration between these platforms, using a linked data structure, based on the data dictionaries developed in WP1 and WP2. The three pilot projects have demonstrated different parts of the ontology (different types of structure), as well as different visualization techniques, within BIM and GIS environments.

The pilot projects have allowed the BIM environment to be both the interface with the user and also the repository of information for decision-making, since it can integrate any kind of data (e.g., sensors, inspections, risk, media), with semantic relationships between concepts. From a technical point of view, we have had to address two main challenges:

- How to structure and organise the data, building on the data dictionaries developed in WP1 and WP2.
- How to integrate data in a BIM environment, in an accessible, scalable and independent way (allowing interoperability with any BIM environment).

For the first challenge, we have developed the CoDEC ontology building on the ontology of the Interlink project, to ensure alignment with the Road OTL and enabling the development of CoDEC ontology instances from Road OTL implementations.

For the second challenge, we have developed the CoDEC API to create an abstraction layer for access (reading and writing) to data described by the CoDEC ontology. This solution created: Technological independence (any suitable system can consume the services provided by the CoDEC API); Reduced complexity (the use of the API services does not require knowledge of linked data and/or SPARQL, nor of the way the ontology is organised); Easy scalability and extension of services (new services can be added to the API, without the need to change the applications that use the API); Easy scalability and extension of the ontology (the ontology can be modified without impact on the services - developers “just” need to ensure that the services behave as intended, using the modified version of the ontology); and easy testing and validation (each service can be tested independently).

To demonstrate the use of the technical environment of the CoDEC project we have developed visualization tools in a BIM environment (for PP1 and PP2) via the production of add-ins to the Bexel Manager tool. These visualization add-ins use the CoDEC technical architecture, but are specific to the Bexel Manager. However, this was for the purpose of demonstration - to implement PP1 and PP2 in other BIM environments it would only be necessary to create the visualization component, as the service (CoDEC API) and data (CoDEC ontology) layers are independent of the BIM environment. On the other hand, since both add-

ins use the same technical environment (Bexel), all the functionality provided in PP1 and PP2 can be used on the same object. Hence it is possible to analyse sensor, risk and condition data in a tunnel or a bridge. The user “just” needs to install both add-ins and populate the ontology with the required information.

For pilot project 3, we have used a GIS environment as the visualisation tool. In this case, the BIM model is used as a data source for the ontology, i.e., there are services in the CoDEC API that allow updating the CoDEC ontology with data extracted from the BIM. The visualisation tool (in this case the GIS) uses the data stored in the CoDEC ontology to extend the visualisation capabilities present in the normal GIS environment for road asset management.

Hence all the Pilot Projects have demonstrated an integrated visualisation environment (in BIM or GIS environment), which utilises complex data stored in a linked data environment. With the proposed solution, it is possible to extend the services in the CoDEC API, making it possible to use any complex query or inference mechanism in the linked data environment. There are several benefits delivered through this:

- It supports advanced inspection methods, enabling analysis of infrastructure inspection data and data from monitoring systems (sensors) within BIM
- It supports centralised information models, with interoperability between multiple data sources, enabling data collection and processing at a centralised point
- It integrates complex data within advanced 3D visualisation tools to allow users to effectively analyse infrastructure status/condition from several viewpoints

The pilot projects have also helped to understand the limitations of current systems and identify the need for developments that could help the future exploitation of the CoDEC approach:

- *Level of detail within BIM models:* To simplify future processes, it is recommended that BIM model designers develop elements with the appropriate level of detail for visualisation - i.e., that visualisation needs are considered when developing BIM models. This simplifies the discretisation of the visualisation components, which, for example, was a critical aspect in the implementation of pilot project 1.
- *Normalisation and standardisation of conventions and nomenclature.* The mapping between the BIM elements and the elements present in the ontology is a critical aspect in the development of the integration. The use of ifcOWL enables the representation of the BIM model elements in the linked data environment. However, the tools for creating the representation of the BIM model in ifcOWL generate models with high detail, making the ontology complex and difficult to process. It is recommended that BIM solution manufacturers provide advanced filtering mechanisms for generating ifcOWL from BIM models. In the pilot projects, the instances of ifcOWL were created manually, which is a major limitation and should be automated as much as possible.
- *Automation.* Whilst the current solution is adequate, it requires effort in data instantiation and synchronization with distinct data sources that limits a fully automated method. Automating all steps in the process would increase the ability to exploit the results of the CoDEC project - allowing a real-time approach.



- *Linked data visualization on BIM environment.* Linked data is visualized in the BIM environment by importing dynamic data (such as sensor data and risk and condition data) via the CoDEC API. This approach does not conform with the principle of separation of concerns, with additional properties being created in the BIM model to store dynamic data. To use visualization features in the BIM environment while ensuring the separation of BIM and linked data, a new, more dynamic, solution should be researched and developed.

## 9 References

W3C Recommendation on SSN; <https://www.w3.org/TR/vocab-ssn/>, 2017 (accessed 9 July 2021)

Studer, R., Benjamins, V. R., & Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data & knowledge engineering*, 25(1-2), 161-197.

Stephan, G. S., Pascal, H. S., & Andreas, A. S. (2007). Knowledge representation and ontologies (pp. 51-105). Springer Berlin Heidelberg.

Pan, J. Z. (2009). Resource description framework. In *Handbook on ontologies* (pp. 71-90). Springer, Berlin, Heidelberg.

## Appendix 1 – Additional Deliverables

### CoDEC ontology Guidance

The CoDEC ontology has been used in every Pilot Project as a way to store knowledge about the structures, structural elements, sensors, sensor data and risk and condition data. This section guides the reader on how to open the ontology, importing data and making sure the knowledge is available for API (or other applications) consumption. Please, ensure that the following files and software are available before continuing:

- **CoDEC Ontology** -> The CoDEC ontology is available in repository folder “CoDEC Ontology” (“codec-project.owl”) and in the project’s website (<https://www.codec-project.eu/>). CoDEC ontology was developed to describe and store knowledge related to European highways industry. The ontology is an extension from Interlink project’s EurOTL. The new concepts and relationships described in the ontology are based on the Data Dictionary created in WP1 and WP2 from CoDEC. The ontology was developed to accommodate knowledge necessary for the pilot projects. Some of the concepts added for this reason include:
  - Pavement Sections and Layers
  - Structural Elements (Tunnel and Bridge Elements)
  - Data properties related to structure and structural elements representation
  - Risk and Condition Data (SSN Ontology extension)
- **Imported ontologies** -> As explained before, CoDEC ontology is an extension of EurOTL and utilizes concepts from Semantic Sensor Ontology (SSN). The imported ontologies can be downloaded in their respective websites (<https://www.roadotl.eu/static/eurotl-ontologies/index.html> and <https://www.w3.org/ns/ssn/>).
- **Protégé** -> Protégé is a standard software in ontology development. Stanford’s Protégé is open-source and free and can be downloaded in <https://protege.stanford.edu/>. This software will be used to open the ontologies and import the necessary knowledge.

### Opening CoDEC Ontology

Requirements: Protégé, CoDEC Ontology and Imported Ontologies

1. Start Protégé.
2. Open CoDEC Ontology (“File” -> “Open...”).
3. Protégé will alert for missing imports (see Figure 58). These alerts appear since ontology IRI may not lead to the ontology file itself (e.g., <http://www.codec-project.eu/def> IRI is not available on the web, nor does it lead to the owl/rdf file). To resolve these alerts just point to the correct imported ontology files that were downloaded before.



Figure 58 - Missing Imports

4. Once the missing imports are resolved, the CoDEC ontology is opened (Figure 59), and the import process can begin.

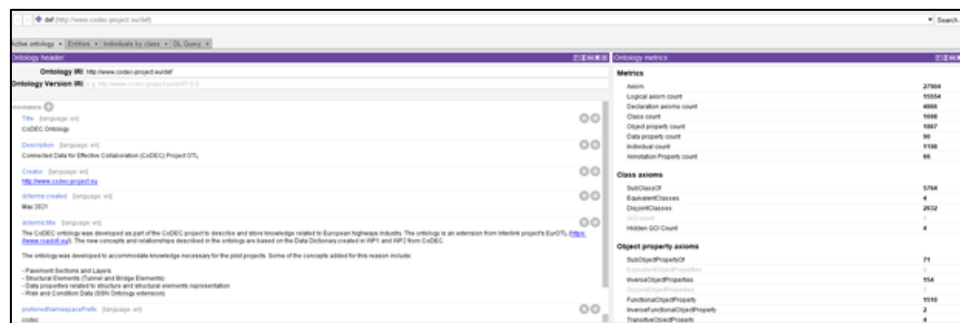


Figure 59 - CoDEC Ontology

## CoDEC Ontology Instances (Importing instance data)

This section describes the import process used for adding instance knowledge to the CoDEC Ontology. Before starting the guide make sure that CoDEC Ontology is opened in Protégé and the following files are available:

- **Instance Data** -> An excel spreadsheet with the knowledge that will be imported to the ontology. While the structure of this document can be quite flexible (the set of rules used in the importation is defined by the user), an example can be found in Pilot Project 2 repository (testDataPP22.xlsx).
- **Rules Definition** -> A json file containing the transformation rules necessary to import the instance data into the CoDEC ontology. **Not mandatory** since these rules can be defined during the importation interface. The rules (defined in Manchester OWL Syntax) can be saved to be reutilized. An example of such files can also be found in Pilot Project 1 repository (sensorImportRules.json) and Pilot Project 2 repository (PP2ImportRules.json).

1. To import instance knowledge to Protégé, Cellfie add-in will be utilized (available in the standard Protégé installation). To open the add-in, go to “Tools” and select “Create Axioms from Excel workbook”.

2. The systems prompt the user to select an excel file. Select and open the Excel file that contains the instance data that will be imported to the ontology.
3. Once the Cellfie is opened with the workbook (Figure 60), the transformation rules can be defined or imported (if a json file with transformations rules was previously available).
4. When creating or editing an existing rule, a new window will appear. The rule should be defined using Manchester OWL Syntax (<https://www.w3.org/TR/owl2-manchester-syntax/>). In each rule definition, select the column which contains the instances (individuals), the type (Class) of such individuals and any relationships or attributes that it might have (see Figure 61).

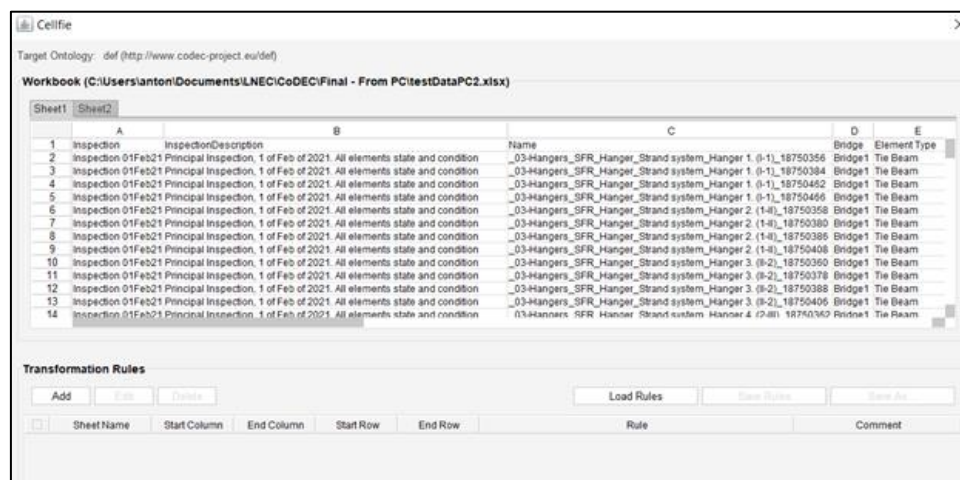


Figure 60 - Cellfie Interface

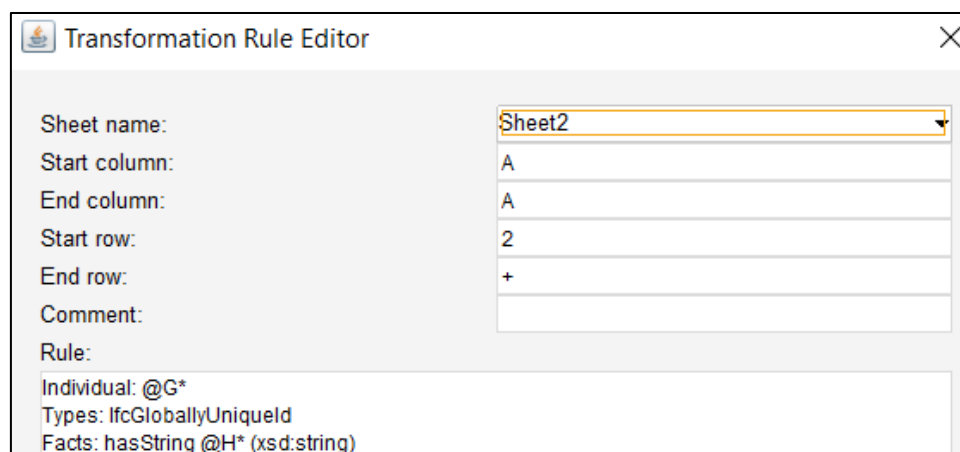
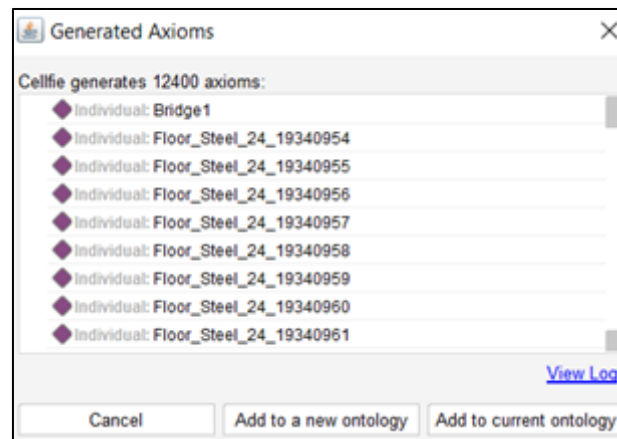


Figure 61 - Transformation Rule example

5. When all the necessary transformation rules are defined select "Generate axioms". This will create the triples with the instance knowledge that will be added to the

ontology. Two options will appear, “Add to a new ontology” or “Add to current ontology” (Figure 62).

6. Try to maintain the CoDEC ontology separated from the individual-full ontology. “Add to a new ontology” is recommended. Once saved, this new ontology (knowledge base containing information about the instances) can now be used inside Protégé or other semantic ready software (such as GraphDB) to be analyzed, queried, and explored.



**Figure 62 - Axiom generation**

The steps described in this section were utilized to create the knowledge bases (ontologies with individuals) for Pilot Project 1 and 2. Pilot Project 3 uses a different import process as explained in the respective PP3 section (Chapter 6).

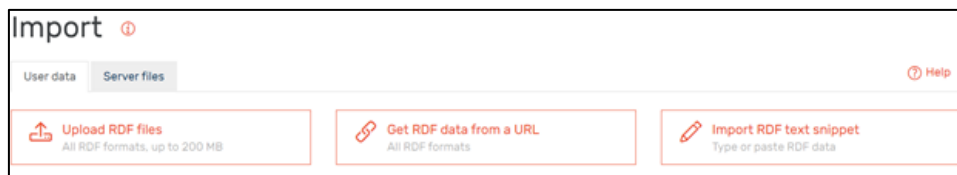
You can find instantiated ontologies in the Pilot Project repositories. The folder "Pilot Project 1" contains the file “sensorTC.owl” with knowledge about eleven sensors, with the respective observations and sensor data for August 2020. The folder “Pilot Project 2” contains the file “testPP2.owl” with information about two inspections, each one with Risk and Condition data about a set of structural elements from a Bridge.

## Importing knowledge to GraphDB

GraphDB is a semantic graph database (stores RDF triples). GraphDB has a free to use license (limit in the number of parallel queries) and can be installed locally or made available on a server. The software can be downloaded in <https://graphdb.ontotext.com/>. GraphDB will serve as the repository once the ontology is ready for consumption.

1. Start GraphDB.
2. Create a new repository (follow <https://graphdb.ontotext.com/documentation/free/quick-start-guide.html> for further instructions on GraphDB setup and repository creation).
3. After creating and connecting to the repository, select “Import” -> “RDF” (other import methods can be used).

4. In the Import interface (Figure 63), select “Upload RDF files”.



**Figure 63 - Import Interface**

5. Select and import the CoDEC Ontology, the Imported Ontologies and, if applicable, an ontology with individuals.
6. The ontology is now available to be explored (e.g. “Explore” -> “Class Hierarchy” or “Class Relationships”) and queried (using SPARQL, in the GraphDB interface or via API).

## CoDEC API

The folder “CoDEC API” contains the source code with all API services developed to support the execution of pilot projects. The CoDEC API is a REST API developed in .NET Core.

In order to compile and run the CoDEC, the following requirements apply:

- Visual Studio 2019 or newer
- ASP.NET Core 3.1 SDK
- Solution uses additional packages from NuGet.org that are configured within the solution and must be installed on the first use.

To set up a specific environment, the user must provide information to connect to the proper GraphDB instance and repository, which is done in file "appsettings.json", filling in the correct values for the variables under "GraphDB" node.

## PP1 Add-in

This section describes the detailed process of using the developed Bexel Manager Add-in, which allows the user to connect to a data source in order to import the processed sensor data into the BIM environment and get advanced visualization of BIM model elements. The folder “Pilot Project 1” contains the files for PP1 Add-in.



### **Content of required files:**

- “CoDEC\_PP1\_Add-in” folder contains all necessary files to demonstrate CoDEC Pilot project 1 use cases, including:
  - “CoDEC Add-in installation” folder contains installation files for Pilot Project 1 CoDEC Add-in.
  - “CoDEC Tunnel V01” folder contains complete Bexel Manger project with imported IFC models of the tunnel, including results of Pilot Project 1 use cases.
  - “Sensor data inputs” folder contains received sensor readings for one month period (XLSX file).
  - “User Manual” folder contains a user guide with detailed explanation of CoDEC Add-in usage (PDF file).

### **How to use Pilot Project 1 CoDEC Add-in?**

- Install BEXEL Manager software package.
- Install Pilot project 1 CoDEC Add-in.
- Follow detailed instructions from section.

### **How to get BEXEL Manager?**

Request for 30-day trial Bexel Manager is at the following link - Trial Request Form.

The user needs to provide contact data in the form. The trial license will be sent to the e-mail indicated. The key will allow you to activate BEXEL Manager software and test it in full function mode for 30 days.

Along with BEXEL Manager trial version, the user will get a password for entering BEXEL Manager User Area on the following link BEXEL Manager User Area, where there is access to a range of free information, including manuals, tutorials, add-ins, API Scripts, Demo / Samples, Databases, to get full support during free trial.

For all CoDEC Project participants, Bexel Manager is available during the entire project duration for testing and project validation purpose. In case you need Bexel Manager longer than 30 days, please contact Bexel support team at support@bexelmanager.com.

### **How to install BEXEL Manager CoDEC Add-in?**

Copy the folder CoDEC Add-in installation (located in “CoDEC\_PP1\_Add-in” folder) to the local computer, open it and run the install.bat file. All BEXEL Manager instances must be closed while the Add-in installation is in progress. After the installation of BEXEL Manager Add-in is finished, it will appear within BEXEL Manager environment under the Add-ins tab.

#### **A. Starting BEXEL Manager**

After successfully downloading the software and completing installation, a BEXEL Manager icon will occur on the desktop.

- B. To run the BEXEL Manager application, please double-click on the shortcut. The Welcome window opens, which enables the user an overview of recently opened projects and basic commands. **Opening the project using the Open command****

1. Click the Open Unmanaged Project command within the BEXEL Manager welcome window.
2. The Open window shows. Here we set a location of the CoDEC Tunnel V01 project that has been previously saved on the local computer (CoDEC\_PP1\_Add-in/CoDEC Tunnel V01). Choose the file CoDEC Tunnel V01.besln. Click the Open button.

The software interface opens with the BIM model preview.

**C. Starting CoDEC Add-in and loading Sensor data from the Web Service**

1. BEXEL Manager is run and the project version is opened. Go to Add-ins tab and click on the CoDEC – Sensor Data Demo button.
2. The Add-in window appears.
3. Click the Load Sensor Data from Service (JSON) button.
4. The user can see Sensor data (loaded from the Web Service) within CoDEC Add-in window. The Addin shows all sensor readings for all sensors by default.
5. To load sensor data into BIM model click the Generate Properties button.

**D. CoDEC Add-in functionalities for generating sensor data into BIM environment**

1. Load Sensor Data from Service (JSON) – Command for loading sensor data from Web Service.
2. Load Sensor Data (Excel) – Command for loading sensor data from the local XLSX file (folder Sensor data inputs, file Overzicht Data uit IRIS - BEV - aug 2020\_SENSORNEW.xlsx).
3. Timeline for choosing period of sensor data reading – Choose time period.
4. Section for Sensor type selection – Choose sensor/all sensors.
5. Date selection – Choose the date.
6. Time selection – Choose the time.
7. Export to JSON – Create JSON files (from XLSX format) for further use.
8. Generate Properties – Command for generating properties into BIM environment.

**E. Bexel Manager 3D Color Coded view**

1. The CoDEC Tunnel V01 project is opened.
2. Choose the 3D Color Coded View mode in the main 3D display window.
3. Make sure that the Custom Breakdowns option is set.

The wall panel elements are shown in different colors depending on the numerical thresholds defined within Custom Breakdown tool (in accordance with the generated sensor values).

## PP2 specific outputs

In the Pilot Project 2 repository folder three different files are found:

- testDataPP2.xlsx - An excel containing structured data related to Risk and Condition Data. The file was based on existing structural elements from the BIM model used in PP2, with dummy data concerning Risk and Condition Data being added to fictitious inspections. The file also contains the necessary data to connect the physical objects' representations to the GUIDs from the BIM model.
- PP2ImportRules.json - Manchester Syntax OWL rules used in Cellfie (Protégé Add-in) to import instanced data from testDataPP2.xlsx file to the CoDEC ontology.
- testPP2.owl - CoDEC ontology with the individuals utilized in Pilot Project 2.
- Add-in – source code, installation files and manuals for Pilot Project 2 Bexel Manager Add-in.

These files are inputs and outputs of the PP2 import phase. As explained before, with the CoDEC ontology loaded in Protégé (see CoDEC Ontology section in this chapter), Cellfie was used to open testDataPP2.xlsx file. Transformation rules were created for both Sheets in the workbook, with the first containing knowledge about the structural data, while the second contains information to instantiate ifcOWL relationships (see Figure 64). These rules are available in the PP2ImportRules.json file.

<input checked="" type="checkbox"/>	Sheet1	A	A	2	+	Individual: @D* Types: 'Bridge (CoDEC)' Facts: hasDirectPart @C*
<input checked="" type="checkbox"/>	Sheet2	A	A	2	+	Individual: @G* Types: IfcGloballyUniqueId Facts: hasString @H* (xsd:string)

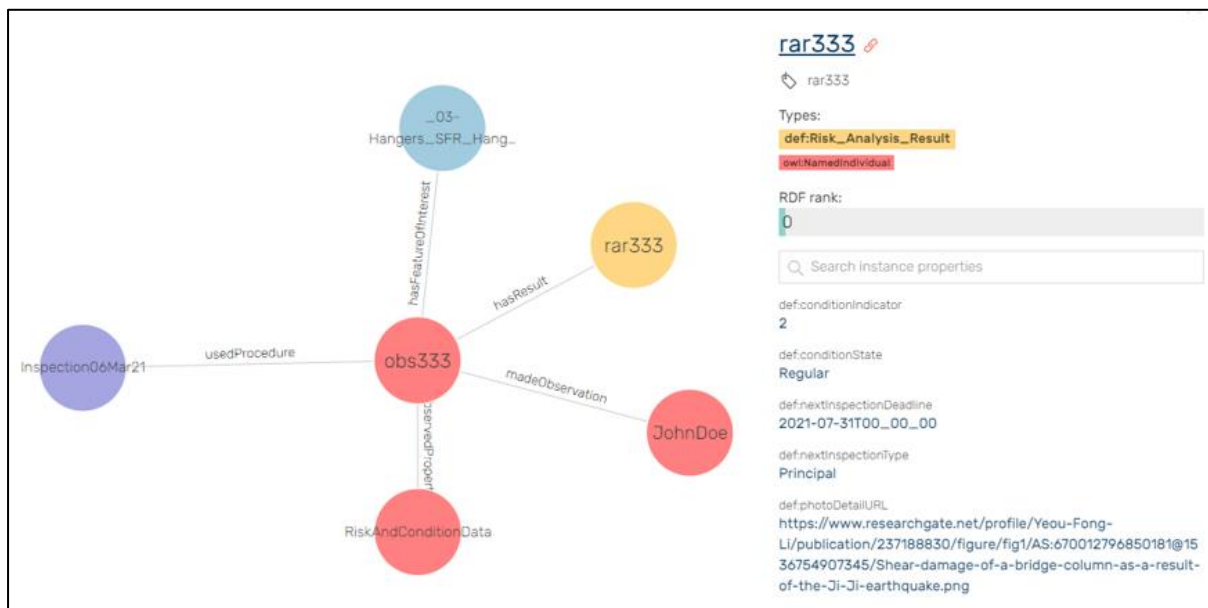
**Figure 64 – Transformation rules in Cellfie for Bridges and GUIDs**

Once the rules were developed, Cellfie creates the individual axioms with the respective data and object properties. The resulting product is a knowledge base with all the information structured according to the CoDEC's entities and relationships. The file testPP2.owl contains this instantiated ontology, with individuals such as the Inspection in Figure 65 below.



**Figure 65 – Inspection01Feb21 individual in Protégé**

In the end, this knowledge base was imported to GraphDB, following the steps describe in the "Importing knowledge to GraphDB" section. Once in GraphDB, the ontology can be explored and queried, either via the API or GraphDB interface. Figure 66 below shows a visual graph with information about an Observation and the result of this Observation in GraphDB.



**Figure 66 – GraphDB Interface: Observation and respective result**

## PP2 Add-in

This section describes the detailed process of using the developed Bexel Manager Add-in, which allows the user to connect to a data source in order to import the bridge inspection data into the BIM environment.

### How to use Pilot Project 2 CoDEC Add-in?

- Install BEXEL Manager software package.
- Install Pilot project 2 CoDEC Add-in.
- Follow detailed instructions from section.

### How to install BEXEL Manager CoDEC Add-in?

Copy the folder CoDECPP2\_Addin\_Install on the local computer, open it and run the Install.bat file. All BEXEL Manager instances must be closed while the Add-in installation is in progress. After the installation of BEXEL Manager Add-in is finished, it will appear within BEXEL Manager environment under the Add-ins tab.

#### **A. Starting BEXEL Manager**

After successfully downloading the software and completing installation, a BEXEL Manager icon will occur on the desktop.

1. To run the BEXEL Manager application, please double-click on the shortcut. The Welcome window opens, which enables the user an overview of recently opened projects and basic commands.

## B. Creating a new project using IFC files

1. Run the software. In the **Welcome** window click on the **New** command.
2. The **Add New Project** window opens. Here we define the **name of the project** and the **name of the project version**. For example, we can set *CoDEC* as project **name** (the **Project** section) and *CoDEC Bridge V1* as the **name** of the project version (the **Version** section). After defining the names and adding description if needed, click on the **Choose** button.
3. In the **Open** window choose **IFC** files and click on the **Open** command. Finish the process by clicking on the **OK** button within the **Add New Project** window.

## C. Starting CoDEC PP2 Add-in and loading Inspection data from the Web Service

1. BEXEL Manager is run and the project version is opened. Go to **Add-ins** tab and click on the **CoDEC Linked Data** button.
2. The **Add-in window** appears.
3. After click on the **Load** button, list of inspections loaded from the service appears within Add-in window.
4. Select one inspection from the list. The **Select Elements** button is available now. Click on it, the elements that are related to selected inspection are selected within Bexel Manager BIM environment (**3D view** in the main 3D display window).
5. Select one inspection. The **Load to BIM** button enables the user to load specific inspection properties from the Web service to BIM environment.

New properties that are generated within Bexel Manager BIM environment are located within Inspections properties group:

## D. Exchange data (Custom Break Down)

If the User wants to import Custom Break Down Structure (CBS) exchange files, with pre-defined rules based on a specific property, follow further steps:

Click on the Bexel Manager in the upper left corner. Choose the command **Exchange**. In the **Exchange data with another Bexel project** click on the **Import** button. Choose the exchange BXF file saved on the local computer, *CBS\_exchange.bxf*. The **Choose Selections for import window** opens, mark **Custom Breakdowns** option, click the **OK** button. Imported CBS appears within Custom Breakdowns section, Building explorer palette.

Note: The new properties should be loaded before creating or importing CBS.

## E. Bexel Manager 3D Colour Coded view

Custom Breakdown structure has been created (or imported as it has been explained above).

1. Choose the **3D Colour Coded View** mode in the main 3D display window.

2. Make sure that the **Custom Breakdowns** option is set.

The elements with specific inspection properties are shown in different colours depending on the values of imported properties, defined rules and colours within Custom Breakdown tool.

#### **F. View pictures**

To see the picture that relates to specific inspection, click on the Value of the property photoDetailURL. The picture will be opened in browser.

### **PP3 specific output**

Folder “Pilot project 3” contains the software components specifically developed to support pilot project 3, namely:

- Python API to connect with the linked data platform

#### **Python API**

Folder “Codec\_Python\_API” contains a set of Python scripts used to connect with the linked data environment. Services were developed to both read and write into the CoDEC ontology.

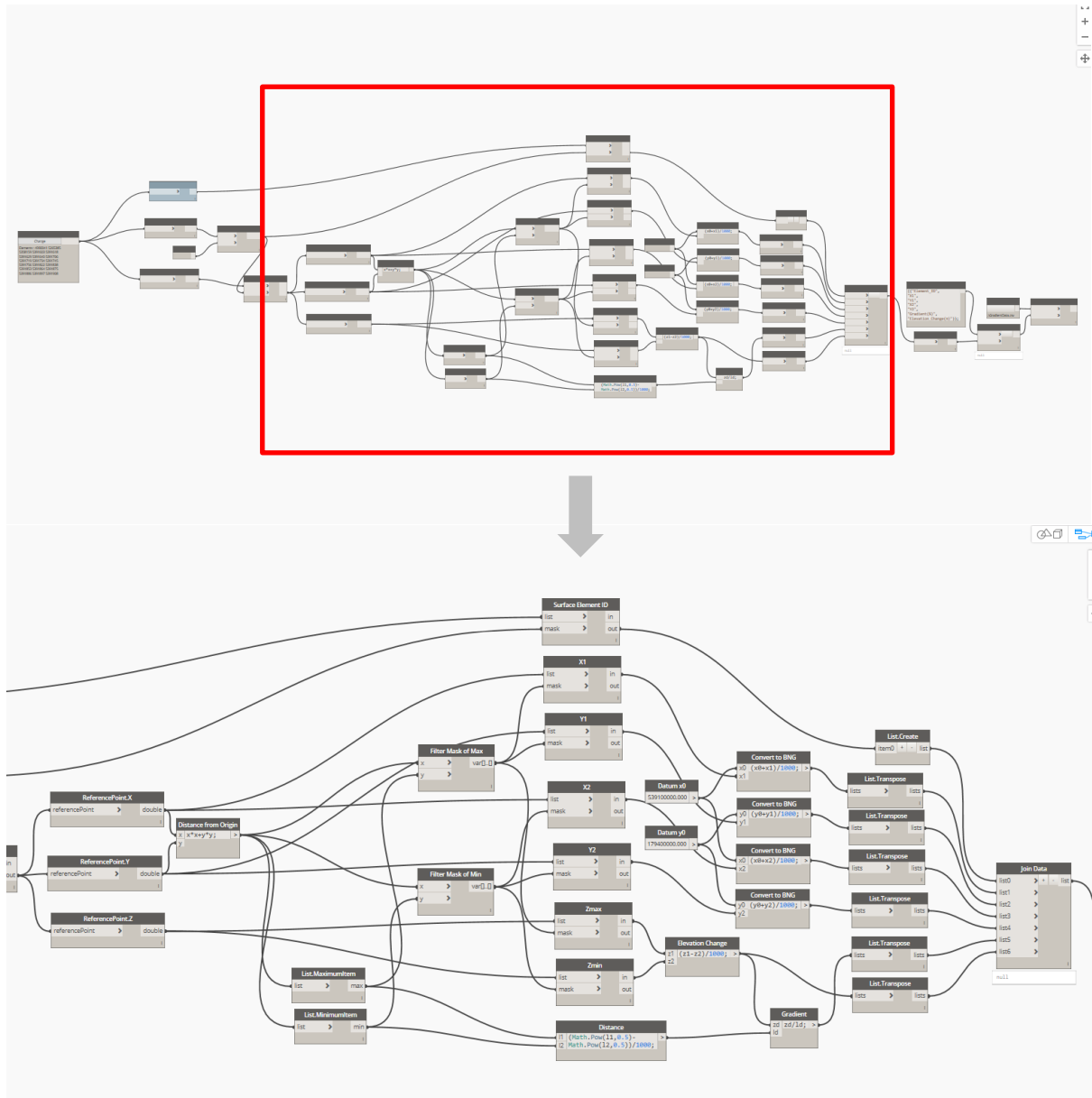
In order to run the scripts, Python 3 is required, and file *config.py* must be configured with the repository path and credentials.

The *access\_graphdb.py* script contains three functions, namely:

1. Load a saved query from the repository - It is possible to store queries in the Codec GraphDB repository. In the script the *saved\_query* function is used to access these stored queries. In this case the "Get Instances" query is read from the repository. This will return the query text that can be used in the *query\_graphdb* function.
2. Perform a GET request to retrieve information from the repository using a SPARQL query - The *query\_graphdb* function is used to perform a GET request to the repository. Next to the endpoint paths and credentials, this function also requires a SPARQL query as a string as input. One option is to use the *saved\_query* function to load a query that is already stored in the repository. It is also possible to enter a custom SPARQL query into the function. The function returns the results of the SPARQL query in JSON format.
3. Perform a POST request to upload information to the repository using a SPARQL query - The *insert\_graphdb* function is used to perform a POST request to the repository. In this function it is also required to enter a SPARQL query as string. This SPARQL query is used to insert new data to the repository. In Codec Pilot Project 3 we used a query with variables to upload the individual road slabs with attributes to the repository. There is a maximum limit of the length of the query string of 5000 characters. Usually when uploading individual instances, this limit will not be reached. However, in Pilot Project 3 we entered geographical data with many coordinate points as text, therefore we needed to simplify the polygons and round the coordinate points.



## Appendix 2 - Dynamo script



## Appendix 3 – Technical Tools Glossary

### **Protégé**

Stanford University's Protégé is a free, open-source platform that provides a suite of tools to construct domain models and knowledge-based applications with ontologies. More information can be found in <https://protege.stanford.edu/>.

### **Cellfie**

Cellfie is a Protégé plugin for creating OWL ontologies from spreadsheets (<https://github.com/protegeproject/cellfie-plugin>). Cellfie imports spreadsheet data into OWL ontologies by using MappingMasterDSL, a domain-specific language based on Manchester OWL Syntax. Once the transformations rules are defined, new axioms are generated and can be imported to the ontology.

### **GraphDB**

GraphDB is a semantic graph database (stores RDF triples). GraphDB has a free to use license and can be installed locally or made available on a server (cloud). The software can be downloaded in <https://graphdb.ontotext.com/>. Knowledge can be retrieved from GraphDB using SPARQL, a query language specific for RDF.

### **EurOTL Ontology (INTERLINK Project)**

INTERLINK, INformation management for European Roads using LINKed data is a previous CEDR project with the objective of defining, link, and manage infra-asset information for roads using Linked Data approach and Semantic. The main deliverable of this project is the European Road Object Type Library (EUROTL). The EurOTL framework covers highly reusable definitions such as provenance, quantities and units, temporal and spatial locations, transport networks, basic support for asset lifecycle and also main asset types and properties as needed for sharing asset lifecycle data. More information about the project can be found in <https://www.roadotl.eu/>.

### **Bexel Manager**

Bexel Manager is a BIM project management platform that provides a single solution and enables construction professionals to digitalize key workflows through advanced integrated and flexible system. BEXEL Manger fully supports BIM Open Interoperability standards and formats such as IFC and BCF file formats. Open API allows users to create fully customized BEXEL Manager tools or add-ins which can automatically execute various analyses and processes. More information about Bexel Manager can be found in: <https://bixelmanager.com/>

### **Revit and Dynamo**

Autodesk Revit is commonly used BIM modelling tool, allowing users to create digital assets for design and construction projects with emphasis on building construction. More information about Revit can be found in: <https://www.autodesk.com/products/revit>

Dynamo is an open-source visual programming language designed for Revit and comes as standard as a plugin to the main Revit program. It facilitates users to automate and iterate processes, access metadata within the model and import and extract information from Revit. Dynamo's website can be found here: <https://dynamobim.org/>

### **ArcGIS Pro**

ArcGIS Pro is a commercial GIS application developed by Esri. ArcGIS is the industry standard for working with desktop GIS. In the pilot project is used for data visualization, data management and spatial analysis. Information about this application can be found here: <https://www.esri.com/en-us/arcgis/products/arcgis-pro/>