



Project Acronym: **BIMERR**
 Project Full Title: **BIM-based holistic tools for Energy-driven Renovation of existing Residences**
 Grant Agreement: **820621**
 Project Duration: **45 months**

DELIVERABLE D6.5

Renovation Process Simulation Tool 2

Deliverable Status: **FINAL**
 File Name: **BIMERR_D6.5-v1.00.docx**
 Due Date: **30/06/2021 (M30)**
 Submission Date: **02/07/2021 (M31)**
 Task Leader: **BOC (T6.3)**

Dissemination level	
Public	X
Confidential, only for members of the Consortium (including the Commission Services)	



This project has received funding from the European Union's Horizon 2020 Research and innovation programme under Grant Agreement n°820621

The BIMERR project consortium is composed of:

FIT	Fraunhofer Gesellschaft Zur Foerderung Der Angewandten Forschung E.V.	Germany
CERTH	Ethniko Kentro Erevnas Kai Technologikis Anaptyxis	Greece
UPM	Universidad Politecnica De Madrid	Spain
UBITECH	Ubitech Limited	Cyprus
SUITE5	Suite5 Data Intelligence Solutions Limited	Cyprus
HYPERTECH	Hypertech (Chaipertek) Anonymos Viomichaniki Emporiki Etaireia Pliroforikis Kai Neon Technologion	Greece
MERIT	Merit Consulting House Sprl	Belgium
XYLEM	Xylem Science And Technology Management Gmbh	Austria
CONKAT	Anonymos Etaireia Kataskevon Technikon Ergon, Emporikon Viomichanikonkai Nautiliakon Epicheiriseon Kon'kat	Greece
BOC	Boc Asset Management Gmbh	Austria
BX	Budimex Sa	Poland
UOP	University Of Peloponnese	Greece
UEDIN	University of Edinburgh	United Kingdom
NT	Novitech As	Slovakia
UCL	University College London	United Kingdom
FER	Ferrovial Agroman S.A	Spain

Disclaimer

BIMERR project has received funding from the European Union's Horizon 2020 Research and innovation programme under Grant Agreement n°820621. The sole responsibility for the content of this publication lies with the authors. It does not necessarily reflect the opinion of the European Commission (EC). EC is not liable for any use that may be made of the information contained therein.

AUTHORS LIST

Leading Author (Editor)				
Surname		First Name	Beneficiary	Contact email
Falcioni		Damiano	BOC	damiano.falcioni@boc-eu.com
Co-authors (in alphabetic order)				
#	Surname	First Name	Beneficiary	Contact email
1	Chávez-Feria	Feria	UPM	serge.chavez.feria@upm.es
2	Demeter	Dominik	NT	demeter_dominik@novitech.sk
3	Kanóc	Csaba	NT	Kanoc@novitechgroup.sk
4	Lampathaki	Fenareti	SUITE5	fenareti@suite5.eu
5	Poveda-Villalón	María	UPM	mpoveda@fi.upm.es
6	Vergeti	Danae	UBITECH	vergetid@ubitech.eu
7	Woitsch	Robert	BOC	robert.woitsch@boc-eu.com

REVIEWERS LIST

List of Reviewers (in alphabetic order)				
#	Surname	First Name	Beneficiary	Contact email
1	Lampathaki	Fenareti	SUITE5	fenareti@suite5.eu
2	Bountouni	Nefeli	SUITE5	nefeli@suite5.eu
3	Giannakis	Giorgos	Hypertech	g.giannakis@hypertech.gr

REVISION CONTROL

Version	Author	Date	Status
0.1	BOC	09/04/2021	ToC
0.2	BOC	11/05/2021	Updated content from D6.4
0.3	BOC	15/05/2021	Added Verification component
0.4	BOC	17/05/2021	Added Service Description model
0.5	BOC	08/06/2021	Integrated UPM content
0.6	BOC	09/06/2021	Integrated SUITE5 content
0.7	BOC	10/06/2021	Integrated UBITECH and NT content
0.8	BOC	28/06/2021	Integrated first review comments
0.9	BOC	29/06/2021	Integrated second review comments
1.0	BOC	30/06/2021	Finalization for submission

TABLE OF CONTENTS

<i>List of Figures.....</i>	<i>8</i>
<i>List of Tables.....</i>	<i>12</i>
ACRONYMS.....	13
EXECUTIVE SUMMARY	14
1. Introduction.....	16
1.1 Objectives of the Deliverable	16
1.2 Introduction of Taxonomy and Methodology	18
1.2.1 Design Tool for Renovation Processes	19
1.2.2 Monitoring and Evaluation Tool for Renovation Processes	19
1.2.3 Innovation and Reflection Tools for Renovation Processes	20
1.2.4 Development Methodology	21
1.3 Updates to the first version of PWMA	22
2. Design Tools for Renovation Process	23
2.1 Renovation Process and Workflow Design Tool	23
2.2 Renovation Process KPI Design Tool	28
2.3 Models Services Marketplace.....	37
3. Monitoring and Evaluation Tools for Renovation Processes	39
3.1 Renovation Processes-Oriented KPI Dashboards.....	39
3.1.1 Models-based Monitoring Dashboards Demonstration	40
3.1.2 Models-based Monitoring Dashboards Architecture.....	43

3.2	Simulation Tools for Renovation Processes.....	49
3.2.1	Simulation Tool Demonstration Use Case.....	49
3.2.2	Simulation tool architecture	54
3.3	Formal Verification Tools for Renovation Processes.....	57
3.3.1	Formal Verification Tool Demonstration Use Case	57
3.3.2	Verification Tool Architecture	62
4.	<i>Reflection and Innovation Tools For Renovation Processes</i>	65
4.1	Process Mining of Renovation Process.....	65
4.1.1	Logs preparation for Celonis	66
4.1.2	Creation of Analysis Workspace.....	67
4.2	Collaborative Reflection of Renovation Process	70
4.2.1	Model Wiki Application	70
4.2.2	Model Wiki Sample Use Case	74
5.	<i>Integration with BIMERR Tools.....</i>	76
5.1	Integration with BIF.....	76
5.2	Integration with Ontology.....	77
5.2.1	KPI Ontology Description	78
5.2.2	Renovation Process Ontology Description	80
5.3	Integration with the Adaptive Workflow Management and Automation tool	81
5.4	Open Integration Framework.....	83
5.4.1	Microservice Definition Model Type	86

6. Catalogue of Tools for Renovation Processes	91
6.1 Design Tools	91
6.1.1 Community version of the Renovation process and KPIs design tool	91
6.1.2 Cloud-Based Renovation Process Design Tool	92
6.1.3 Standalone package of the Cloud Renovation Process Design Tool	92
6.2 Monitoring, Evaluation, Reflection, and Innovation Tools	93
6.3 Open Integration Framework OLIVE	94
7. Conclusion and Outlook.....	96
BIBLIOGRAPHY.....	97
Annex.....	98
BPMN Mapping to Petri Net.....	98
Properties to Computation Tree Logic Mapping	102

LIST OF FIGURES

Figure 1 - Ecosystem Overview	18
Figure 2 - Digital Twin Evolution (Tchana de Tchana et al., 2019).....	19
Figure 3 - Renovation Process Design Tool Community Version	24
Figure 4 - Renovation Process Cloud Modelling Environment.....	25
Figure 5 - Renovation Process Cloud Modelling Environment Main Interface	26
Figure 6 - Renovation Process Cloud Modelling Environment Design Interface	27
Figure 7 - KPI model.....	29
Figure 8 - Goals attributes	29
Figure 9 - KPI Attributes.....	30
Figure 10 - Trustability Attributes	31
Figure 11 - Data Calculation Model	32
Figure 12 - Metric Attributes	34
Figure 13 - Data Items Attributes.....	35
Figure 14 - Renovation Process KPIs Design Tool Community Version.....	36
Figure 15 - Renovation Process Marketplace	37
Figure 16 - Renovation Process Marketplace Architecture.....	38
Figure 17 - Renovation KPI Cockpit Use Case	39
Figure 18 - Backward-looking Monitoring and Forward-Looking Simulation of KPI- Scaffold Costs	41

Figure 19 - Simulation Output for KPIs.....	42
Figure 20 - Facade Renovation Status KPI Dashboard.....	43
Figure 21 - KPIs Dashboard architecture	44
Figure 22 - KPI Dashboard Chart Widget	45
Figure 23 - KPI Dashboard Table Widget	46
Figure 24 - KPI Dashboard Image Widget	46
Figure 25 - KPI Dashboard Tree Widget	47
Figure 26 - KPI Dashboard Widgets Combined.	47
Figure 27 - KPI Trustability Details.	48
Figure 28 - Renovation Process Simulation Inputs.....	49
Figure 29 - Renovation Process Simulation Input C_START_EVENT	50
Figure 30- Renovation Process Simulation Input C_TASK.....	51
Figure 31 - Renovation Process Simulation Input Calculation	52
Figure 32 - Renovation Process Simulation General Results.....	52
Figure 33 - Renovation Process Simulation Detailed Results.....	54
Figure 34 - Renovation Process Simulation Engine Architecture	55
Figure 35 - Formal Verification Input.....	57
Figure 36 - Deadlock Analysis	58
Figure 37 - Unboundedness Analysis	59
Figure 38 - Reachability Analysis.....	60

Figure 39 - Path Existence Analysis	61
Figure 40 - Formal Verification Architecture	62
Figure 41 - Celonis Logs Preparation.....	66
Figure 42 - Celonis Analysis Workspace for BIMERR.....	68
Figure 43 - Excel Output	69
Figure 44 - Model Wiki Use Case Scenario	71
Figure 45 - Model Wiki Architecture	72
Figure 46 - Facade Renovation Process to Wiki	74
Figure 47 - Wiki Pages Generated for the Facade Renovation Process.....	75
Figure 48 - Comments Imported for the Building Scaffold Task of the Facade Renovation Process.....	75
Figure 49 - Example of KPI Ontology Population	80
Figure 50 - Example of Renovation Process Ontology Population	81
Figure 51 - Services Configuration UI in Workflow.....	82
Figure 52 - Olive High Level Overview	84
Figure 53 - Olive Microservice Controller Architecture	86
Figure 54 - Microservice Definition Model Sample	87
Figure 55 - Microservice Definition Model, Input and Output Attributes.....	88
Figure 56 - Microservice Definition Model, Start and Call Attributes	89
Figure 57 - Microservice Definition Model, Publishing feature	90

Figure 58 - ADOxx Olive Homepage	94
--	----

LIST OF TABLES

Table 1 - Renovation Process Simulation General Results Details	53
---	----

ACRONYMS

Acronym	Meaning
API	Application Programming Interface
BIF	BIMERR Interoperability Framework
BIMERR	BIM-based holistic tools for Energy-driven Renovation of existing Residences
BPMN	Business Process Model Notation
FaaS	Function as a Service
KPI	Key Performance Indicator
MIME	Multipurpose Internet Mail Extensions
OSGi	Open Service Gateway initiative
PWMA	Process & Workflow Modelling & Automation
DSL	Domain Specific Language
PNML	Petri Net Modelling Language
CTL	Computation Tree Logic
EBNF	Extended Backus-Naur form
PQL	Publisher Query Language
SOAP	Simple Object Access Protocol
REST	Representational State Transfer
ABL	ADOxx Binary Language

EXECUTIVE SUMMARY

This document describes the final set of renovation process management tools, which we consider as an ecosystem of applications, Software as a Service offerings, microservices, as well as 3rd party applications, providing renovation specific features.

The provided renovation process management environment has two types of flexibility to enable configuration and adaptation. First, we use the meta-modelling platform ADOxx that enables the configuration of Process modelling notation, Key Performance Indicator (KPI) modelling notation and Data modelling notation by providing a full-fledged process model repository. ADOxx uses conceptual meta-models to define the modelling language, hence the modelling language can be adapted to the needs of BIMERR and aligned with the BIMERR ontology to use the same semantics. This semantic alignment enables the seamless use of data that come from other BIMERR applications. The ADOxx platform can be downloaded for academic use for free at adoxx.org and the corresponding BIMERR specific configurations are provided for download in the so-called development space of the developer community on the dedicated adoxx.org webpage.

To provide features, services, and tools for the ecosystem around the process management platform, we used the Microservice framework Olive. In particular, using Olive, we provide:

- (i) Features like (a) the knowledge-based simulation and formal verification of renovation processes, (b) the dashboard visualisation of renovation process status, (c) the co-creative reflection of the renovation process using XWIKI and generating pages from models and feedback comments from pages into the models.
- (ii) Connectors to third party tools like (a) to export process models for execution to the BIMERR Adaptive Workflow Management and Automation tool, (b) to import data from the BIMERR Interoperability framework (BIF) for displaying the status of the renovation process, (c) to interact with a Process Mining tool that analyses the process execution after the renovation process has been completed in order to create lessons learned for the next renovation project.

The functional capabilities have been defined in the corresponding D6.3 “Adaptive Renovation Process & Workflow Models 2”. Therefore, the deliverable at hand explains the technical concepts, the tool functionality of the requested features and provides the different applications for download at www.adoxx.org.

1. INTRODUCTION

1.1 OBJECTIVES OF THE DELIVERABLE

This deliverable provides the final set of Features for Renovation Process Modelling.

This deliverable corresponds with Deliverable D6.3 “Adaptive Renovation Process & Workflow Models 2” and provides the technological basis to manage the renovation process. Therefore, this document focuses on the tools, infrastructures and technical frameworks that are provided to enable renovation process management.

Process management is often performed by a standardized tool, mainly providing design features for process notations such as BPMN (Business Process Modelling Notation). Although for some cases those standard drawing tools may be sufficient, we observe challenges when aiming to interpret the models.

In this case the process models require to be stored with the corresponding semantic description (in our case we use conceptual meta-models), hence simple drawing tools are not sufficient. Full-fledged modelling tools typically provide a model repository with the capability to parametrize the models and each individual object inside the model. This enables key features of process management such as simulations, formal verification and model transformation but on the other side require sophisticated repository technology.

Model-driven tools can follow one of two complementary approaches. The first one is the standardized tool approach, aiming to implement a standard tool according to a standard modelling notation with standard features. The second approach is the provisioning of configurable and flexible tools that can provide both standard modelling notation and features, as well as personalized and configurable modelling approaches and personalized tool features. We provide the latter, as it enables flexibility in both: (a) the modelling languages as well as (b) the modelling features. On the other hand, we can always downgrade our tool by instantiating a particular standard.

This deliverable introduces the meta-models and tools for renovation process simulation created around the platform ADOxx, which is openly available for academic use in the

world-wide community ADOxx.org¹. Results of BIMERR, in form of the corresponding prototypes introduced in this deliverable are therefore available for open use and partly open source in this community. The meta-modelling approach enables the configuration of the modelling language and thus allowing a personalised configuration of the platform modelling language. In our case we adapted the modelling language BPMN to also include KPIs relevant concepts and semantically enriched the modelling languages for alignment with the BIMERR ontologies and data models as introduced in the BIMERR Deliverable D4.3 and followed in BIF.

Besides the flexibility of the meta-model, we introduce the microservice Framework Olive² that enables the flexible configuration and personalization of the functional capabilities of the application. The application consists of a set of cloud offerings, in combination with tools and microservices, hence we follow the idea of an “ecosystem” that has the process management application in the center and integrates and uses several microservices and additional tools to personalize the functional capabilities.

The functionalities introduced in this deliverable in particular focus on the simulation and formal verification of renovation processes, on the monitor of their progress respect to the simulated cases and on their improvements analyzing the execution logs and the feedbacks from stakeholders. Additionally, Olive enables the integration with the BIMERR Adaptive Workflow Management and Automation tool described in BIMERR deliverable D6.7 and the integration with BIF described in BIMERR deliverable D4.9.

This deliverable describes the final set of ADOxx meta-modelling configuration, tools and microservices composing the PWMA, to provide an ecosystem for renovation process management.

¹ <https://adoxx.org/live/web/bimerr/overview>

² <https://www.adoxx.org/live/olive>

1.2 INTRODUCTION OF TAXONOMY AND METHODOLOGY

The application ecosystem consists of the main application and the supporting microservices framework. The main application in our setup is the design component realised on the meta-modelling platform ADOxx providing all necessary modelling features. The microservice framework Olive includes a set of identified features to support the renovation process management. Those features are either implemented as microservices that provide the requested feature or as microservices that interact with a corresponding third-party tool.

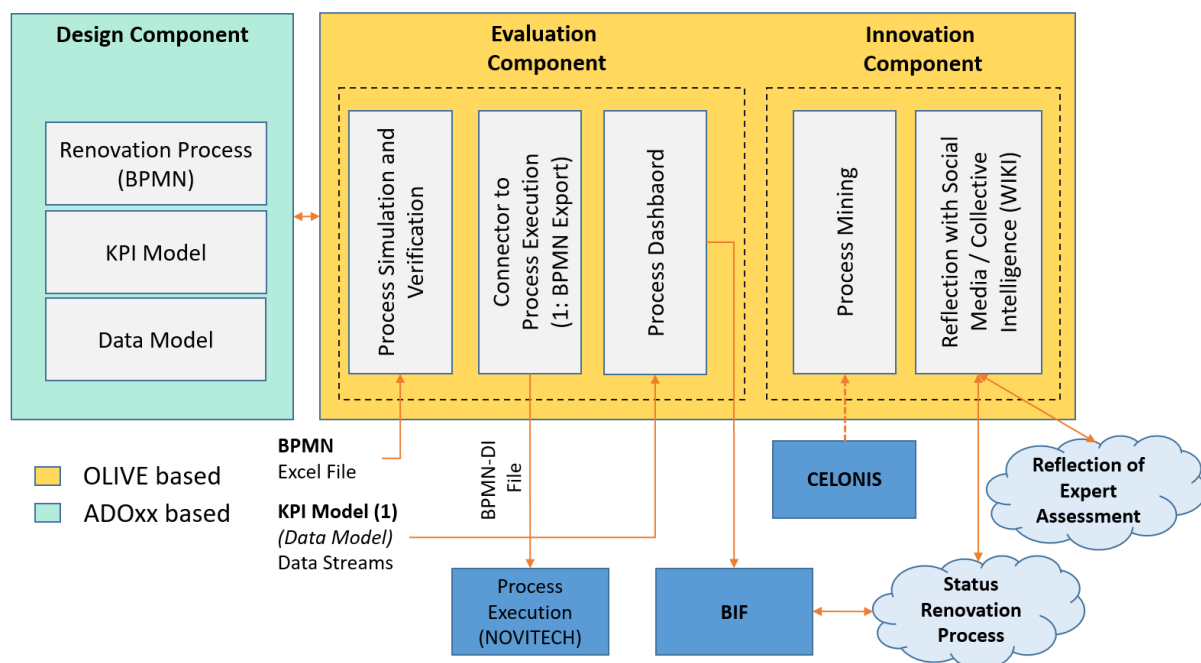


Figure 1 - Ecosystem Overview

Figure 1 introduces the PWMA ecosystem that is provided in the context of this deliverable through a running demonstrator as well as the corresponding ADOxx configuration files and Olive based Microservices for BIMERR as downloadable packages on ADOxx.org.

The ecosystem proposed in Figure 1 reflects the classification of digital twins proposed by (Tchana de Tchana et al., 2019), where three evolution phases are identified to reach a digital twin: (a) Digital Model, (b) Digital Shadow and (c) Digital Twin.

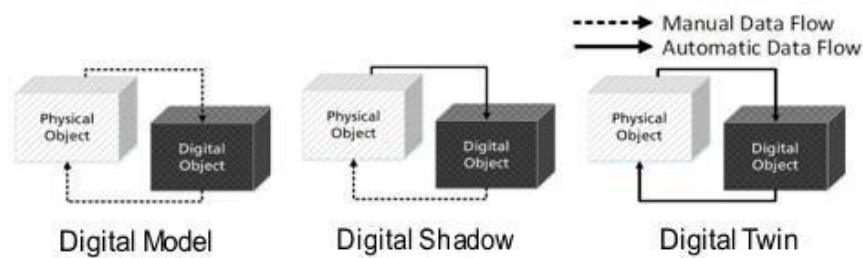


Figure 2 - Digital Twin Evolution (Tchana de Tchana et al., 2019)

The Digital Model phase is supported by the design components of the ecosystem where the domain knowledge is collected using appropriate model types. In the Digital Shadow there is a connection with the real world and the renovation process is simulated, executed, and monitored. Finally, in the Digital Twin, the automatic adaptation of the renovation processes is supported with data mining and reflection components. In the following subsections, each components category is introduced.

1.2.1 Design Tool for Renovation Processes

The design tool is based on ADOxx.org which can be configured using a modelling library configuration – in the so-called ABL file format – that uses the pre-defined functional capability of the modelling repository, the access and model management, the graphical drawing, the analysis, some simulation algorithms as well as the transformation algorithm that enables the conversion of the models into other formats. This transformation engine is important as it enables the graph-rewriting of a model and hence the transformation into another syntax to allow interaction with other tools.

The three boxes of the design component (a) Renovation process, (b) KPI Model and (c) Data Model correspond to different configuration files – ABL files – that configure the modelling environment accordingly, to enable process modelling, KPI modelling or data modelling depending on which configuration file is in use.

1.2.2 Monitoring and Evaluation Tool for Renovation Processes

This tool set is provided in the form of microservices that interpret the KPI and Data Models and collect data from BIMERR Interoperability Framework – BIF from WP4. Thanks to the semantic alignment between the metamodel and the BIMERR ontology described

in section 5.2, the data that are received from the BIF can be related to the concepts on the models. This knowledge-based enrichment enables the use of semantic information and domain specific concepts when reading the BIF data.

This alignment with BIMERR concepts enables to connect the renovation process models with (a) a process execution that reports each stage of the process, (b) a forward-looking simulation that assesses how the process is likely to be in the near future and (c) a formal verification tool that checks the structural correctness of the process. The forward-looking simulation is novel and the mechanisms in use are based on discrete event simulation that simulates each BPMN token and can read the properties for each process element per token. Such detailed configuration possibilities for the simulations enable a so-called knowledge-based simulation, where knowledge is currently extracted in form of an Excel sheet that configures the simulation. Additionally, the formal verification is using state-of-the-art techniques to transform the process in a formal model first and then apply mathematical models to verify logical properties in order to identify structural problems in the renovation process models.

The semantic alignment between the concept models in our design component and the BIMERR ontology allows also to connect with the BIMERR Adaptive Workflow Management and Automation tool, responsible for the execution of the renovation process, in order to provide the workflow to execute and monitor the execution status in a dashboard with renovation specifics KPIs calculated using also data extracted from BIF.

1.2.3 Innovation and Reflection Tools for Renovation Processes

The final phase of the renovation process management is related to learning and reflection. We consider reinforcement learning by using process mining to check if the process execution runs as planned. When decisions are being made, the status of the process as well as the status of the simulations are stored, hence the reinforcement learning cycle can consider the lessons learned generated from the process log files and introduce them in the form of updated models and simulation setting for the next renovation project.

As the renovation process is a highly manual task in its execution, we also propose the complementary use of a collaboration platform in form of Wiki. The open-source project xWiki was used to demonstrate the alignment of wiki pages with the models. Models are used to generate wiki pages, which can be used to document a certain process stage,

elaborate decisions or co-creatively improve the model. The wiki pages are seen as complementary input from users to better document a process status, contribute to a decision or reflect on the decision retrospectively.

1.2.4 Development Methodology

The prototypes have been developed using a rapid prototyping approach that is combined with a design-thinking approach in two iterations. Firstly, the design-thinking is a top-down approach where ideas on the functional capabilities of the Process & Workflow Modelling & Automation (PWMA) are assumed and then developed as a proof of concept. The idea has been elaborated with the end users and all other stakeholders in BIMERR deliverable D6.1 and the functional capabilities has been elaborated with the end users in BIMERR deliverable D6.3. Secondly, the actual software development is performed following a rapid prototyping approach, where one prototype is developed after the other, and each prototype has a typical size of 5-10 person days of implementation. This leads to a series of rapid prototypes which result into one consolidated prototype when the functional capabilities that have been originally foreseen are available. Each intermediate rapid prototype has been presented to the end users and the stakeholders to allow agile changes. The final prototype with its full functional capabilities is approached in two iterations, the first from month 10-18 and the second from month 19-30. The first iteration focused on the initial set of functional capabilities to present a proof-of-concept of the PWMA, whereas the second iteration focused on the integration of other BIMERR and 3rd party services and the adaptation of the prototype while being used by the end users. Overall, we followed the combination of iterative design-thinking approaches and proof-of-concept development in combination with rapid prototypes to achieve the proof-of-concept as appropriate for the PWMA development.

1.3 UPDATES TO THE FIRST VERSION OF PWMA

This document extends the D6.4 “Renovation Process Simulation Tool 1” with the last improvements in the modelling related components of the Process & Workflow Modelling & Automation (PWMA) toolkit as in the following:

- The formal verification feature that enriches the simulation of the renovation processes has been introduced.
- The KPI model has been extended to introduce reliability indicators used to specify how reliable the value of a specific KPI can be based on specific factors.
- A model service marketplace has been introduced as an initial approach to simplify the fruition of the model related services.
- The dashboard of the monitoring and evaluation tools for renovation processes has been updated to align with the reliability indicators and improved mapping between the Business Process Model Notation (BPMN) and petri-net that have been described and used in the simulation component.
- The model wiki component of the innovation tools for renovation processes has been updated with a new and simplified user interface and the Celonis data mining tool has been validated with additional data.
- PWMA has been integrated with other BIMERR tools. The integration comprises the last changes in the ontology, in BIF, in the workflow engine connections and the new microservice definition model for the Olive framework.

2. DESIGN TOOLS FOR RENOVATION PROCESS

The Renovation Process design tool is the starting point for all the tools described in this deliverable. It is based on the ADOxx³ meta-modelling platform that is used to create the renovation process and workflow design tools as well as the KPIs design tool.

In the following sections, initially the demonstrations of the design environment for the renovation process and the renovation workflow and then the design environment for the Renovation KPIs and Goals are reported.

2.1 RENOVATION PROCESS AND WORKFLOW DESIGN TOOL

The business process design tool is an application built with ADOxx. ADOxx is a meta-modelling platform that allows to define your own meta-model and automatically generates the modelling environment for you accordingly. The meta-model in the business process design tool is based on the BPMN2.0⁴ standard schema. This allows to have a modelling environment that is compliant with the BPMN2.0 standard, allowing to abstract the renovation process at many levels. In the D6.3 (BIMERR Consortium,2020) the concept of Template models has been introduced and all the created models have been described in detail. The template model is a renovation process abstract enough to be valid for all the renovation use cases, including all the practices to be considered. This template process has been designed using the BPMN2.0 meta-model. At a more detailed level, the Renovation process instance represents the instantiated templated model relative to a specific use case. Decisions specific to the use case have been taken depending for example on the type of the facade to renovate or on the ventilation system in use, so the model has been kept as BPMN. At a lower abstraction level, the renovation process workflows have been modelled with enough details to be executable by the BIMERR workflow engine. Appropriate constraints have been applied to create a model compliant with the BIMERR workflow engine, such as the necessity of converging

³ www.adoxx.org

⁴ <https://www.bpmn.org/>

gateways after every choice. Again, in this case, the BPMN2.0 meta-model has been used in order to be compliant not only with the BIMERR Workflow Engine but also with the majority of workflow execution engines on the market.

The Renovation Process Design tool has been provided in two forms, (1) a community version library for ADOxx and (2) a project specific cloud version of ADOxx modeler.

The community version library allows to create an ADOxx modelling environment as a windows desktop application that everyone can freely setup and redistribute. The renovation process design tool in this case is based on ADOxx v1.5 and allows to model the renovation process template, instance and workflow using the BPMN2.0 standard. Additional features are in this case provided by the ADOxx Community in terms of add-ons that the user can install from the ADOxx community portal www.adoxx.org.

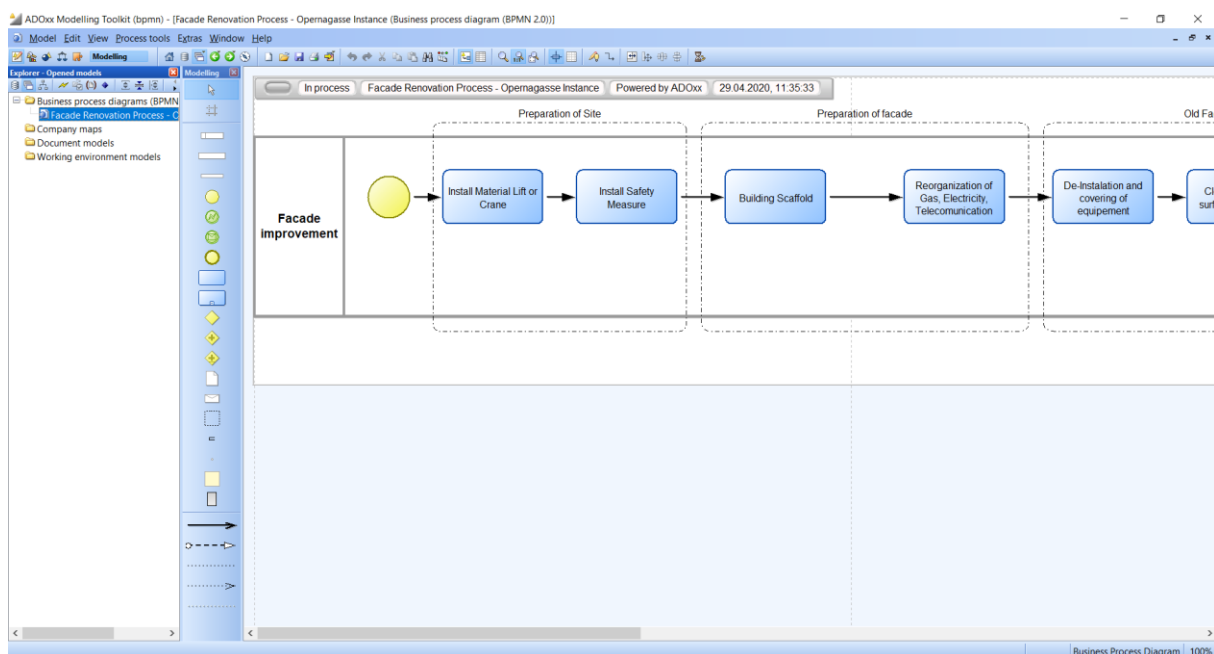


Figure 3 - Renovation Process Design Tool Community Version

The cloud version of the Renovation Process Design tool is a BIMERR customized version of the cloud based ADOxx and is available under the endpoint https://bimerr.boc-group.eu/ADONISNP10_0/auth.view (Figure 4). The integration with the BIMERR Identity provider Keycloak is currently ongoing so a separate login interface is currently available. Credentials can be freely requested on faq@adoxx.org and as soon as the integration with

the BIMERR Identity provider is completed the modelling environment could be accessed with the same credentials of the other BIMERR tools.

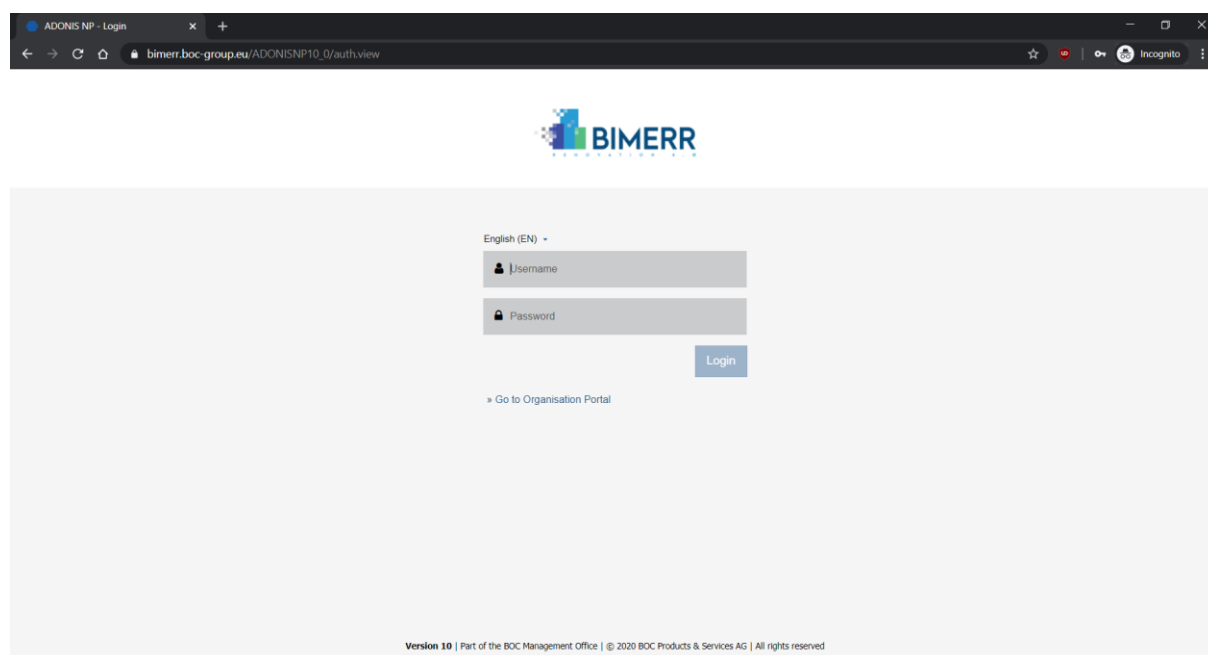


Figure 4 - Renovation Process Cloud Modelling Environment

The cloud version of the Renovation Process Design tool (Figure 5) is a full business process management suite with possibilities not only to create models but also to manage their release cycle and check their correctness. In this demonstration we will focus on the design features of the renovation process templates, instances and workflow models in the BPMN2.0 standard model-type.



Figure 5 - Renovation Process Cloud Modelling Environment Main Interface

In the “Design & Document” section of the modelling environment is possible to explore all the existing models in a folder tree view, visualise them and create new ones using the BPMN2.0 modelling canvas (Figure 6). Here the interface allows to click and create all objects supported by the BPMN2.0 standard, with some facilitating features like next objects and connectors suggestions or automatic alignments.

The tool provides export features in the BPMN2.0 standard format as well as generation of images and reports with details of the model objects. This and other features are also available in a REST interface to enable integration with other components of the BIMERR platform and with the Adaptive Workflow Management and Automation tool.

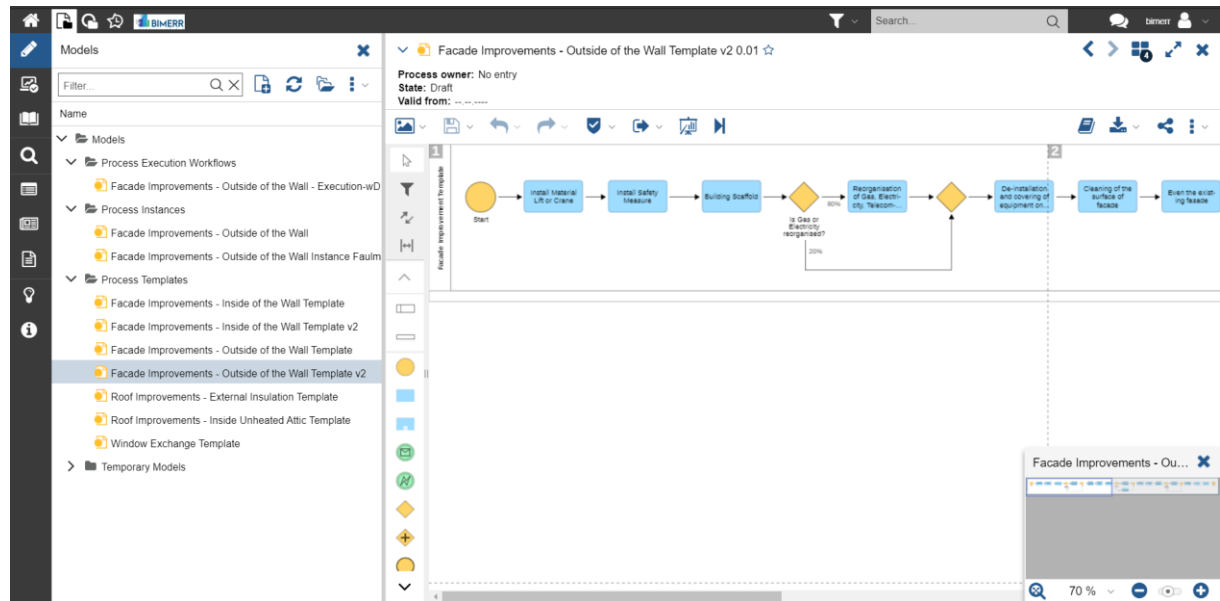


Figure 6 - Renovation Process Cloud Modelling Environment Design Interface

2.2 RENOVATION PROCESS KPI DESIGN TOOL

The Renovation process KPI design tool is an application built with ADOxx. Similar to the renovation process design tool, a community version of the platform is available for the renovation process KPI design tool.

In the renovation process KPI design tool the meta-model is based on concepts of the balanced scorecard (Kaplan, Robert S., and Norton, 1992), extended with a data model-type that allow to specify how the KPIs are retrieved and calculated. In the D6.3 (BIMERR Consortium,2020) the concept of KPIs for the scaffold cost of the renovation facade scenario has been introduced. The first defined meta-model is the “Cause and Effect” model-type. It allows to define KPIs and Goals with their relations and group them in specific perspectives. In particular:

- Perspectives: group similar KPIs, like grouping all “Costs” indicators or all “Time”.
- Goals and sub-goals: describe the objective to be achieved.
- KPIs: describe measurable data sets that assess in combination with the indicator context – plan value, real value, thresholds, type of thresholds and meta data about the indicator –, if the corresponding goal can be achieved or not.

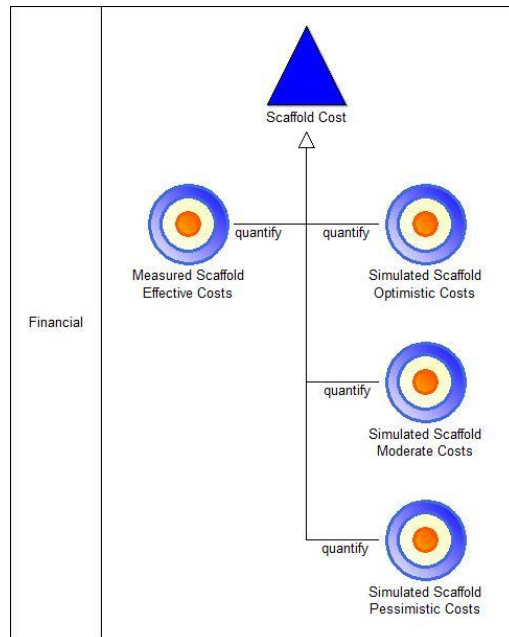
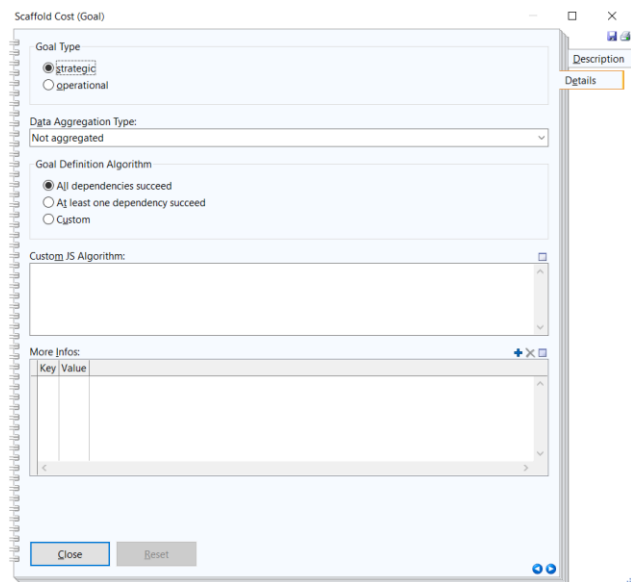


Figure 7 - KPI model

Every object in the model has a common set of attributes like a name and a description of the object, plus a set of specific attributes that characterize it.

In the case of Goals and Sub-Goals, the specific attributes (Figure 8) refer to the type of the goal that can be Strategic or Operational and on the aggregation type of the data that represent how often this goal is evaluated. Referring to the details of the evaluation of the Goal, it is possible to specify the procedure used during its evaluation. The goal, therefore, can succeed if all its dependencies succeed or at least one succeeds.



The dialog box titled 'Scaffold Cost (Goal)' contains the following fields and options:

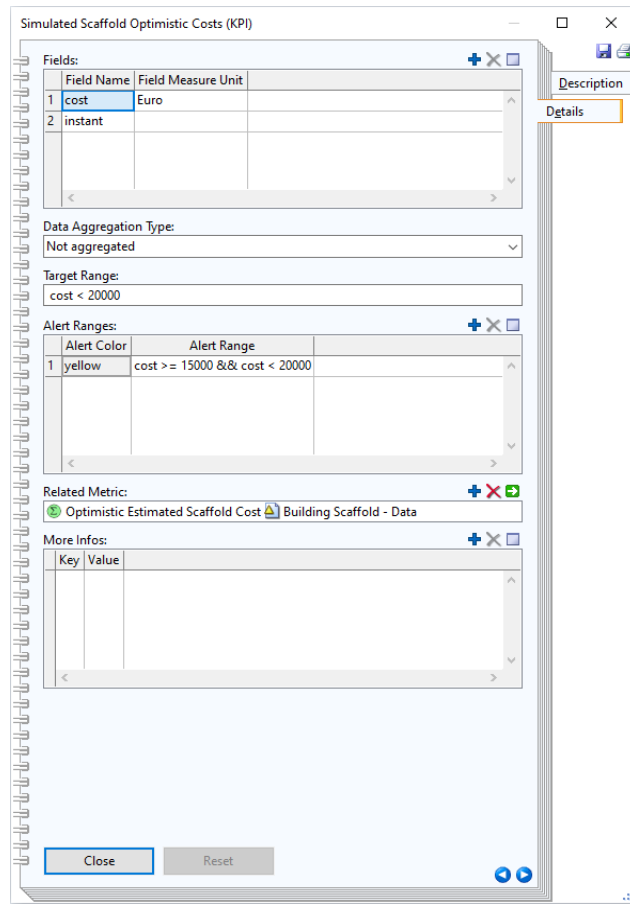
- Goal Type:** Radio buttons for 'Strategic' (selected) and 'Operational'.
- Data Aggregation Type:** A dropdown menu currently showing 'Not aggregated'.
- Goal Definition Algorithm:** Radio buttons for 'All dependencies succeed' (selected), 'At least one dependency succeed', and 'Custom'.
- Custom JS Algorithm:** A text area for entering a custom algorithm.
- More Infos:** A section with a '+ X' icon and a table with columns 'Key' and 'Value'.
- Buttons:** 'Close' and 'Reset' buttons at the bottom.

Figure 8 - Goals attributes

Additionally, if none of these reflect the goal behaviour, it is possible to provide the actual algorithm in JavaScript format needed to evaluate the Goal. In this context the connection flows between the goal and its relevant KPIs and sub-goals that determine the goal dependencies that are relevant.

In the case of KPI objects, the specific attributes are relevant to the fields of data available in the KPI and information on the target and alert ranges of the KPI value (Figure 9).

The Fields represent what kind of metrics are available in these KPIs. Usually there is a value field (cost in this case) that contains the value of the KPI and an instant time field that contains at what time the KPI value has been calculated, but this is not always applicable, so the user can provide as many fields as he needs. Every field can also have a specific measurement unit.



	Field Name	Field Measure Unit
1	cost	Euro
2	instant	

Data Aggregation Type: Not aggregated

Target Range: cost < 20000

	Alert Color	Alert Range
1	yellow	cost >= 15000 && cost < 20000

Related Metric: Optimistic Estimated Scaffold Cost Building Scaffold - Data

	Key	Value
--	-----	-------

Figure 9 - KPI Attributes

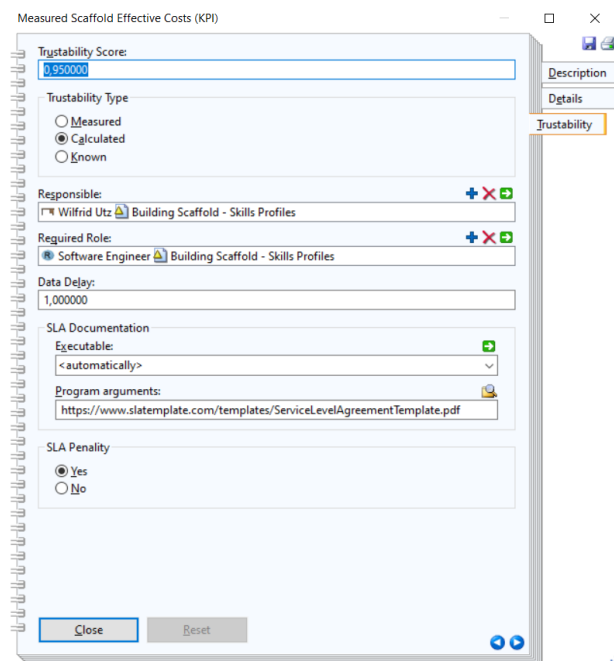
Also in this case, it is possible to specify the aggregation type of the data representing how often the KPI is calculated. KPIs, to be meaningful, must be associated with thresholding that is represented as target and alert ranges. The target range must contain a formula (as a JavaScript expression) that uses the field name defined above, and that specifies the value range that is the target of our KPI (e.g., Cost < 20000). Using the same approach, it is possible to define one or many alert ranges that allow to specify when the KPI is approaching a risky value on the border of the target range. Multiple alert ranges are possible here, for example, if the cost exceeds 15.000 then it should produce a yellow/moderate alert and if the cost becomes greater that 19.000 it should produce a red/important alert, because the value is approaching the top border of the target range.

Ranges allow to represent information on the threshold of a value (e.g., 20000) combined with information of expected directions of the values (so if the threshold is an upper or lower bound), allowing to represent cases of discrete values as well.

Each KPI has also some associated attributes related to its reliability. This set of attributes is related to:

- The type of the data used by this KPI.
- The experience of the responsible that provide the KPI data.
- The delay of the KPI data.
- The presence of penalties applied by Service Level Agreement (SLA) to this KPI.

The data associated can be of three types: Measured, Calculated and Known. If the data is measured, it means that it is provided by BIF or measuring device. In this case the data has the major level of reliability. Calculated data is instead provided by an algorithm while known data is based on the expertise of the provider. In all the three cases the experience of the responsible for the specific data has a key impact on the reliability. Every KPI have associated for this reason an attribute allowing to specify the actual responsible for the specific data and the required role that this responsible must cover.



The screenshot shows a software window titled "Measured Scaffold Effective Costs (KPI)". It contains several input fields and sections:

- Trustability Score:** A text input field containing "0.950000".
- Trustability Type:** A section with three radio buttons: "Measured", "Calculated" (which is selected), and "Known".
- Responsible:** A dropdown menu showing "Wilfrid Utz" and "Building Scaffold - Skills Profiles".
- Required Role:** A dropdown menu showing "Software Engineer" and "Building Scaffold - Skills Profiles".
- Data Delay:** A text input field containing "1,000000".
- SLA Documentation:** A section with "Executable:" set to "<automatically>" and "Program arguments:" set to "https://www.slatemplate.com/templates/ServiceLevelAgreementTemplate.pdf".
- SLA Penalty:** A section with two radio buttons: "Yes" (which is selected) and "No".

At the bottom of the window are "Close" and "Reset" buttons. On the right side, there is a vertical toolbar with icons for "Description", "Details", and "Trustability".

Figure 10 - Trustability Attributes

In case the data of the KPI is not provided in real-time, the delay attribute can be used indicating the time units of delay. At the end if there is an SLA associated to this KPI, it can be referenced and enriched with the presence of a penalty. All these attributes contribute to the calculation of an overall trustability indicator.

At the end, it is important to associate the KPI to a metric to be correctly calculated. The metric is defined in the data calculation model that specifies how the metric value must be retrieved or calculated.

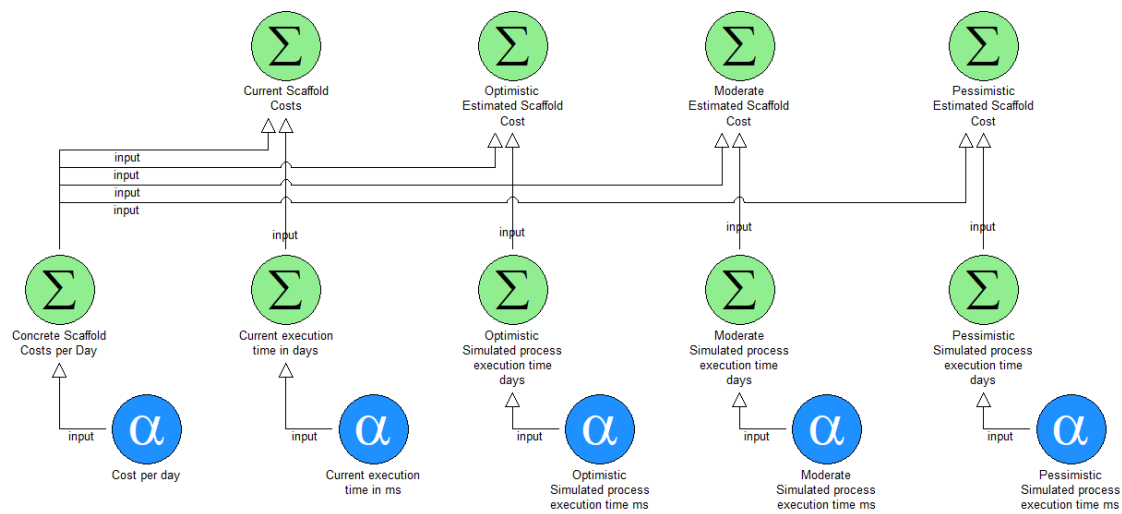


Figure 11 - Data Calculation Model

The data calculation model is composed of Metrics and Data Items including dependencies between them. Metrics (represented as green circles) represent data in a specific format and contain information on how the value of these data must be calculated using as inputs sub-metrics and data access indicators. The Data Items (represented as blue circles) on the other side can describe how a data value is retrieved from an external system, as for example a remote service like the BIF, a middleware, a database or even an Excel sheet. In this context, the data access indicator is strongly dependent on the Olive microservice framework that is responsible to provide the features to access external systems.

Additionally, in the data calculation model, every object has a common set of attributes like a name and a description of the object, plus a set of specific attributes that characterize it.

For the Metric objects, such attributes refer to the way the metric is calculated based on its dependencies. The Input Object Aliases attributes refer to this and allow to specify for every dependency an alias name to use in the calculation formula. Alias can be any name but must not contain spaces or start with a number.

The Fields represent what kind of data are available in this metric and as described in the KPI attributes it usually contains two field, one about the data value of the metric and one

about the instant time of the data. Every field can optionally have a specific measurement unit (Euro in the case of the cost field) but must specify the formula used to calculate the specific field of the metric. The function can be described as a JavaScript expression and the defined aliases can be accessed and used in the formula. Fields of dependent objects can be accessed using the dot operator. For example, if we defined an alias for a sub-metric and the sub-metric includes a field “cost”, this can be reached in the formula by writing the alias name followed by a dot followed by the field name (e.g., “a.cost”). Every field defined must contain a calculation formula. If there is no need for a formula, like in the case of the instant time field, this can be taken directly from the dependency (e.g., specifying “b.instant”). Considering as a sample the metric “Current Scaffold Cost”, that should be calculated multiplying the scaffold cost per day with the current execution time, we can define this behaviour by creating first the aliases for the sub-metrics “Concrete Scaffold Costs per Day” and “Current execution time in days”, named respectively “a” and “b”. Then the “cost” field of the current scaffold can be calculated using the function “a.cost * b.executionTime”, where “a.cost” refers to the “cost” field of the “Concrete Scaffold Cost per Day” sub-metric, while “b.executionTime” refers to the “executionTime” field of the “Current execution time in days” sub-metric. About the “instant” field we used the formula “b.instant” meaning that we use here the same value of the “instant” field of the sub-metric “Current Execution time in days”.

In case of exotic metrics functions, it is possible to provide your own algorithm in JavaScript format that will calculate the metric fields using the “Custom JS Algorithm” attribute.

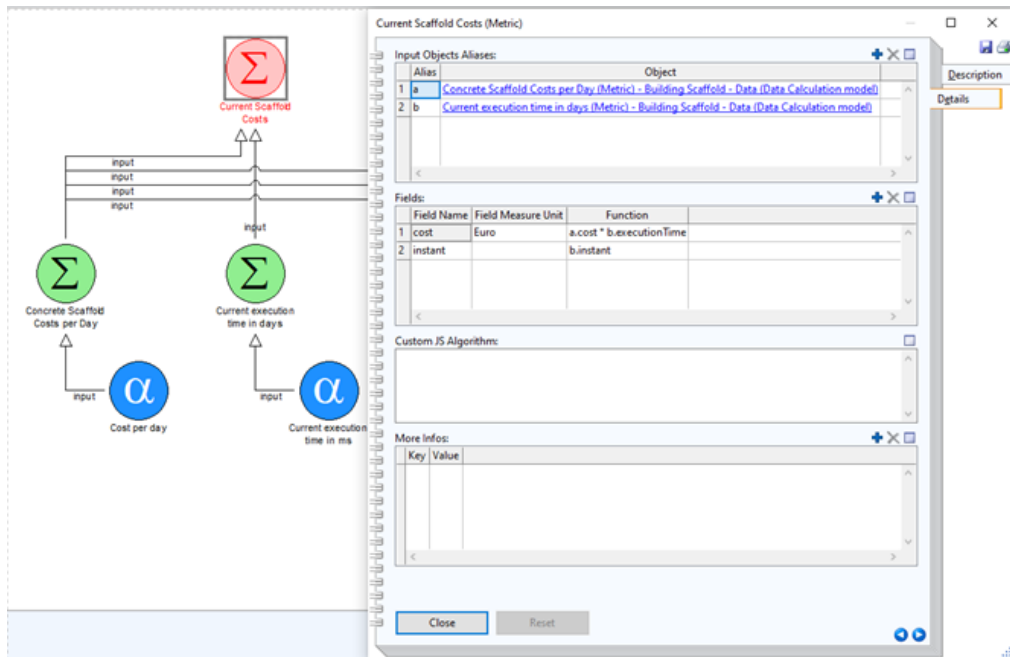


Figure 12 - Metric Attributes

The Data Items can represent how the data values are collected from external systems through the microservice framework Olive⁵. For this purpose, the microservice must return the data in a specific format to be recognized. The data must be a JSON object that contains a "columns" array with the list of returned field names (that must match the one defined in the fields attributes) and a "data" array of JSON objects each one containing a key for every field defined in the "columns" with the appropriate value in string format.

An example of a valid JSON that the Olive microservice must return is the following:

```
{
  "columns" : ["cost", "instant"],
  "data" : [{
    "cost": "2000",
    "instant": '2020-01-30T11:40:22'
  }, {
    "cost" : "1900",
    "instant": "2020-01-29T10:40:50"
  }, {

```

⁵ <https://www.adoxx.org/live/olive>

```

    "cost" : "1800",
    "instant": "2020-01-28T09:40:15"
  }
}

```

Also in this case, it is important to define the data fields that the service returns with optional information on the measure unit for the specific value in the field.

As soon as the microservice in Olive is ready, it is important to refer to it, using its unique id and providing the operation name and its required inputs in terms of input id with the appropriate value. This will be the same input that the microservice expects as a JSON format but explicitly defined key by key.

In the case of the Data Items that contain the results of the optimistic simulation (“Optimistic Simulation process execution time ms”) the microservice operation used is the “getSimulationResults” providing as input the parameter “simulationType” with value “o” that in the context of the service means “give me the result of the optimistic simulation”. The returned JSON contain the fields “executionTime” as milliseconds value and the “instant” time of the simulation.

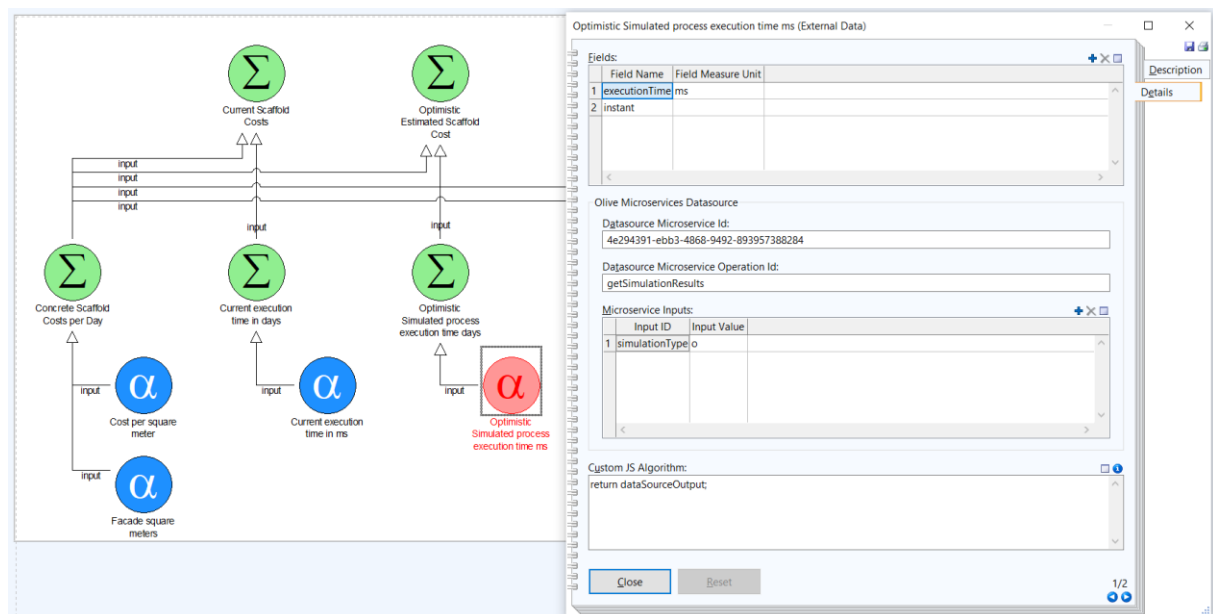


Figure 13 - Data Items Attributes

The community version library allows to create an ADOxx modelling environment as a windows desktop application that everyone can freely setup and redistribute. The renovation process KPI design tool in this case is based on ADOxx v1.5 and allows to model the renovation process KPIs using the previously described meta-model in terms of KPIs and Data access models. Additional features are in this case provided by the ADOxx Community in terms of add-ons that the user can install from the ADOxx community portal www.adoxx.org.

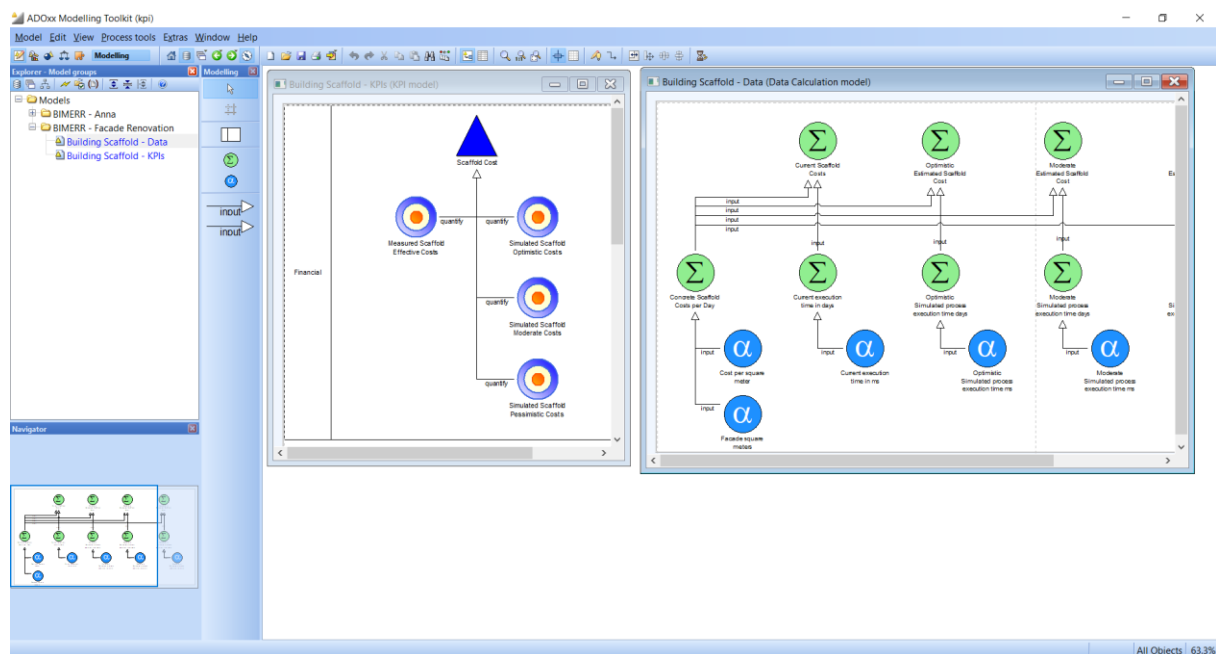


Figure 14 - Renovation Process KPIs Design Tool Community Version

2.3 MODELS SERVICES MARKETPLACE

To simplify the usage of the models and their specific features to the final user, we introduced a sort of marketplace for renovation processes, where the users can see all the defined renovation processes avoiding the complexity of the modelling environment and, for each model, use the available PWMA features.

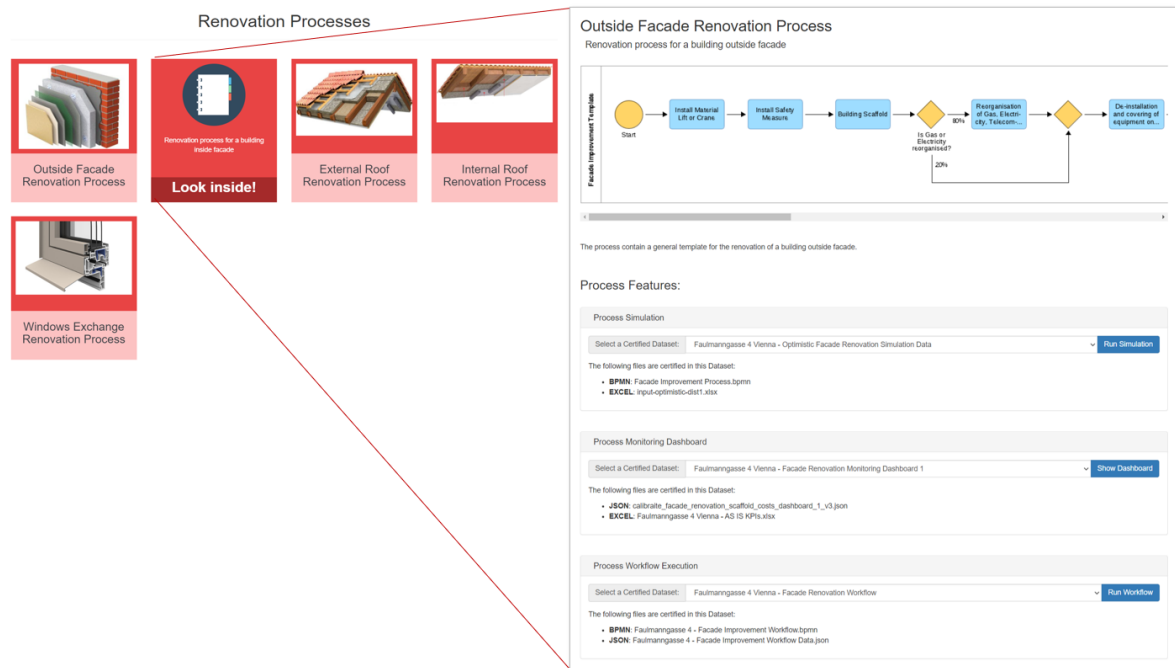


Figure 15 - Renovation Process Marketplace

The marketplace retrieves all the renovation processes directly from the modelling environment and based on the available dataset, it allows to (a) simulate, (b) visualise the KPIs and (c) execute the workflow of a specific instance of the renovation process.

As soon as a renovation process is selected, the list of all the available features is presented. The user must select, for each feature, the dataset he/she wants to use for the specific instance of the renovation process. In the case of the Simulation feature, the simulation engine will be pre-loaded with the selected dataset, and it can be executed clicking the “Run Simulation” button. The process monitor feature instead, after clicking the “Show Dashboard” button will open the visualisation of a configured dashboard for

the specified running process, while the workflow execution will upload the workflow of the selected instance to the “i3d” engine for execution.

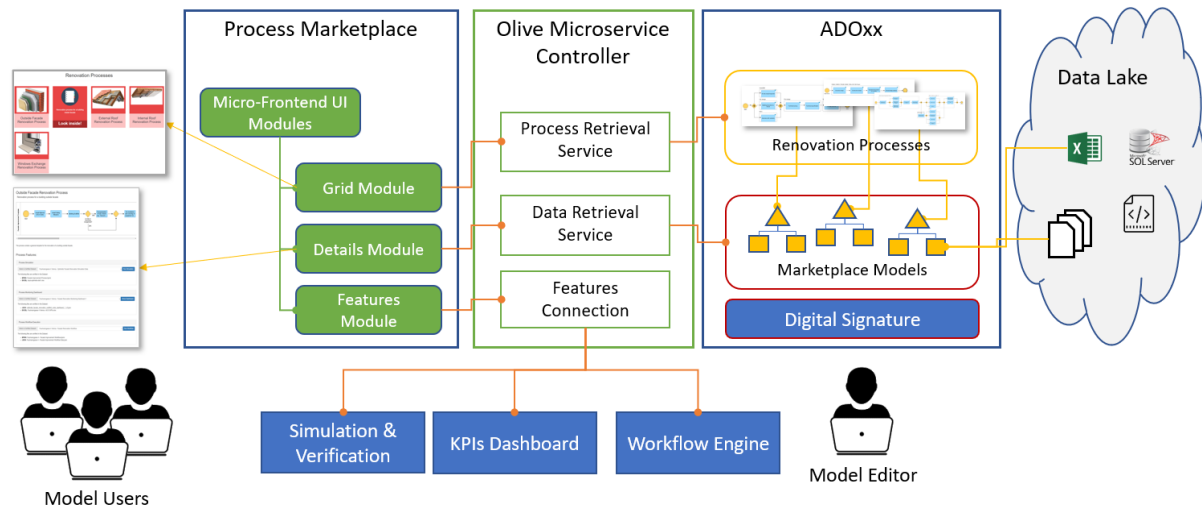


Figure 16 - Renovation Process Marketplace Architecture

The access to all the features is mediated by services created using the Olive Microservice Framework explained in chapter 5.4 while the User Interface (UI) is composed of reusable micro-frontend modules.

To configure valid data for a specific feature over a renovation process, a specific feature-based marketplace model type is required. This model type contains the semantic associations between a feature and a dataset, also providing methods to certify the dataset’s correctness through a digital signature approach. Digitally signing a model allows to check that the model has not been changed by someone who does not have competence on the selected domain and guarantees that the provided dataset is valid for the specific feature. The marketplace model is not intended to be demonstrated in the context of BIMERR project but will be finalized in future projects.

3. MONITORING AND EVALUATION TOOLS FOR RENOVATION PROCESSES

The monitoring and evaluation tools for the renovation process are used to support the user in the evaluation of the status of the process and on the future behaviour.

In the following sections the demonstration of the monitoring cockpits first and renovation process simulation and verification later, are reported.

3.1 RENOVATION PROCESSES-ORIENTED KPI DASHBOARDS

The KPI dashboard visualises in a combined view both the backward-looking monitoring and the forward-looking simulation results. The dashboard interface is based on configurable widgets where the KPIs can be visualised in different formats according to the widget features. The KPIs definitions are taken directly from the renovation process KPIs design tool that can export them in a format recognized by the dashboard. The dashboard uses the KPIs information reported in the model to automatically evaluate them, calculating the relative metrics and retrieving the data from the data sources (BIF and simulations results) using the appropriate Olive microservice.

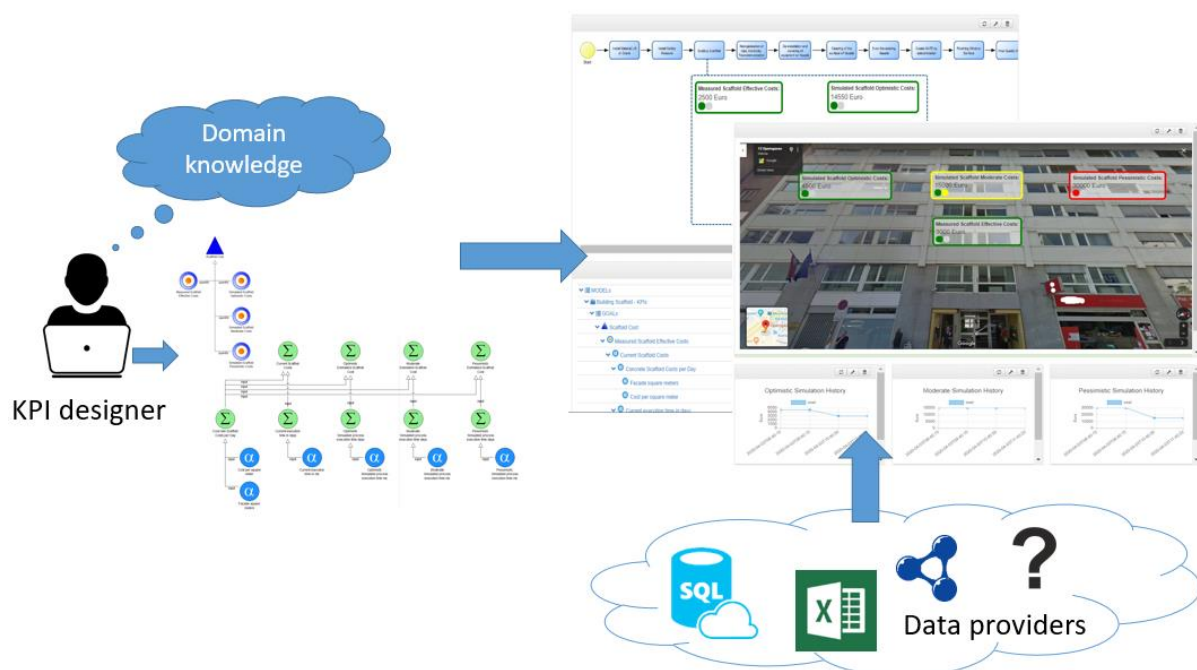


Figure 17 - Renovation KPI Cockpit Use Case

3.1.1 Models-based Monitoring Dashboards Demonstration

The KPI dashboard is strongly based on the underlying KPI data model. A detailed description of the models created for the BIMERR demonstration is available in the BIMERR deliverable D6.3. A sample model focused on the scaffold costs, contains the definition of the following KPIs including their relative metrics and data sources:

- **Measured Scaffold Effective Costs:** a backward looking KPI that monitors the scaffold current costs on a scheduled basis.
- **Simulated Scaffold Optimistic Costs:** a forward looking KPI that shows the results of the last simulation performed with optimistic risks evaluation on the scaffold estimated costs.
- **Simulated Scaffold Moderate Costs:** a forward looking KPI that shows the results of the last simulation performed with moderate risks evaluation on the scaffold estimated costs.
- **Simulated Scaffold Pessimistic Costs:** a forward looking KPI that shows the results of the last simulation performed with pessimistic risks evaluation on the scaffold estimated costs.

Three dashboards are currently available that support a product view of the KPIs and two type of process dependent view.

The first dashboard uses four widgets to visualise the KPIs: one image map widget and three chart widgets. The image map visualises the building facade to renovate using an image from the construction site and an overlay with the values of the previously described KPIs using a color code that immediately reflects the KPI status (green for KPIs with value in the target range, yellow for KPIs with value in the alert range and red for KPIs with value outside the target range). The three chart widgets are used to display the historical trend of the three simulation results, by visualising in a Cartesian chart the estimated scaffold cost for every performed simulation over time, allowing to analyze the alignments of the simulation inputs with real past data.

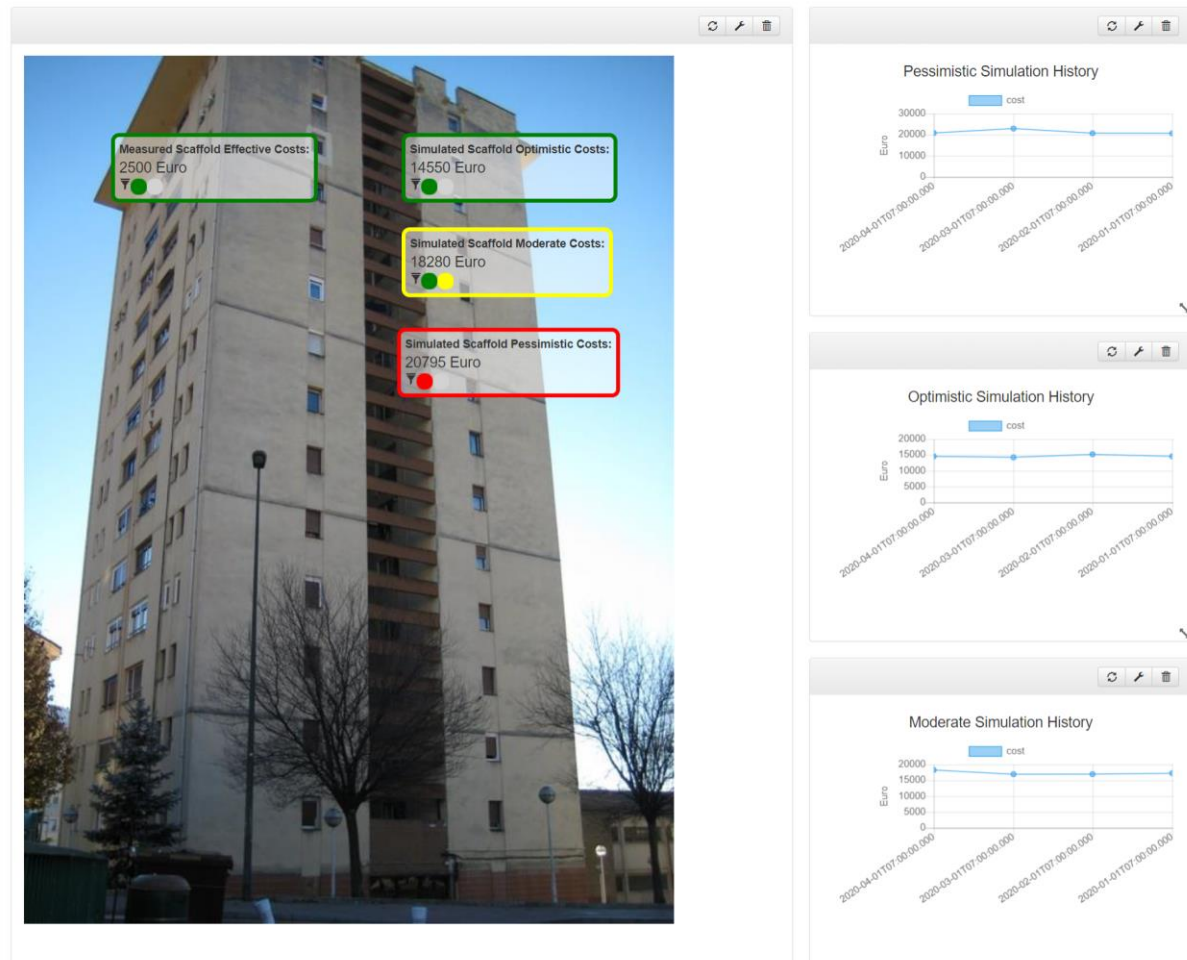


Figure 18 - Backward-looking Monitoring and Forward-Looking Simulation of KPI-Scaffold Costs

The second dashboard demonstrates the Process-Oriented Context of the dashboard with the capability to link KPIs to different phases of the process. Each time-slot of a process can be linked to the actual as well as to the simulated KPIs.

The process-oriented representation also allows to drill the KPI down either in the process-oriented view, or using the model-tree, which represents the KPIs as they are modelled in the KPI model. This helps to understand the cause of a failing KPI checking how its dependencies behave, thus finding the root of the problem.

Hence the process-oriented representation is on the one hand an alternative visualisation of the dashboard and secondly allows the introduction of the linkage of process phases to a concrete KPI.

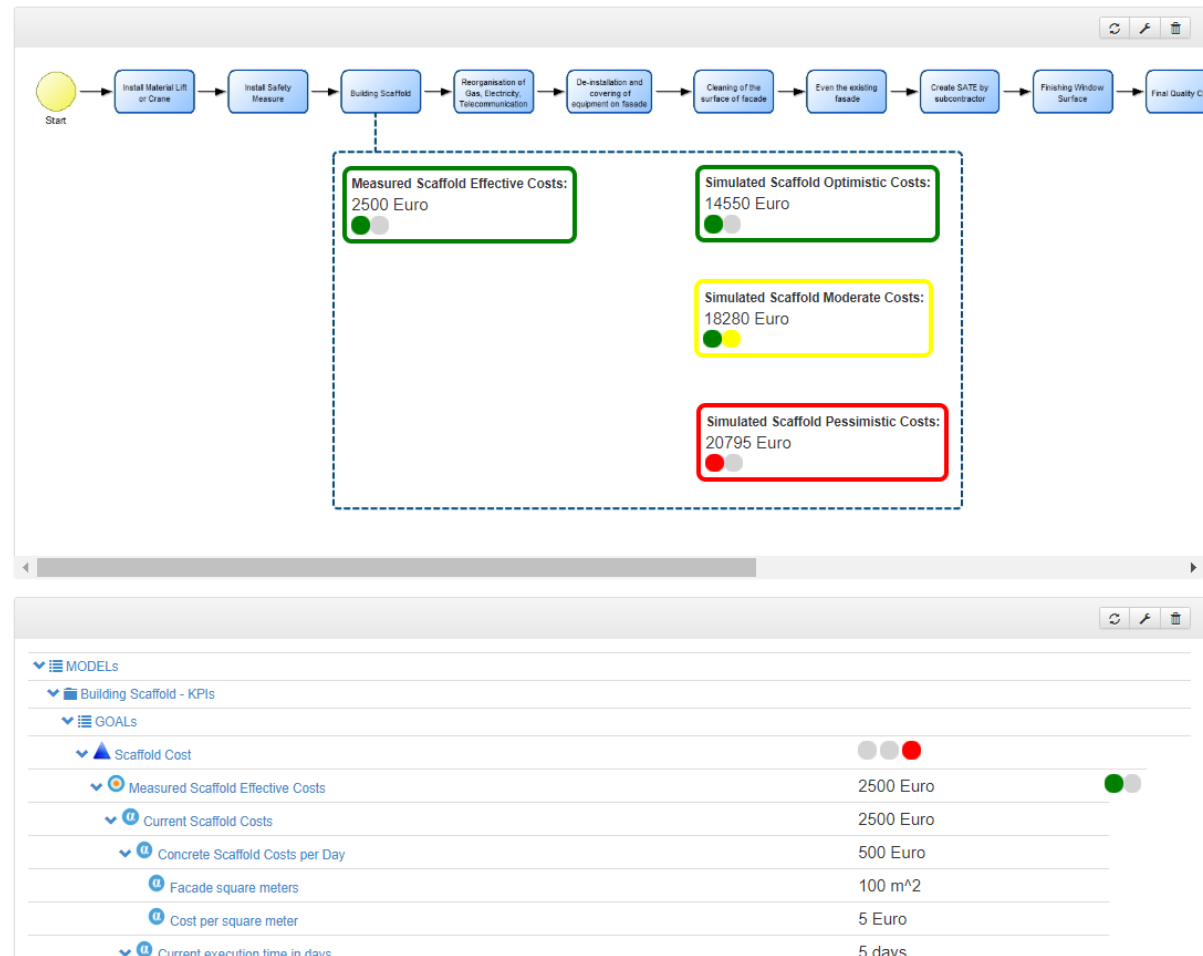


Figure 19 - Simulation Output for KPIs

The intention is that a dashboard can be created where KPIs are linked to different phases of the process that can be simulated and hence dependencies between phases can be considered on an aggregated view. In case a complex process is described by several construction sites running in parallel, and each process of the individual construction sites uses simulated KPIs, the aggregated complex process emerges from the simulated using the underlying processes dependencies.

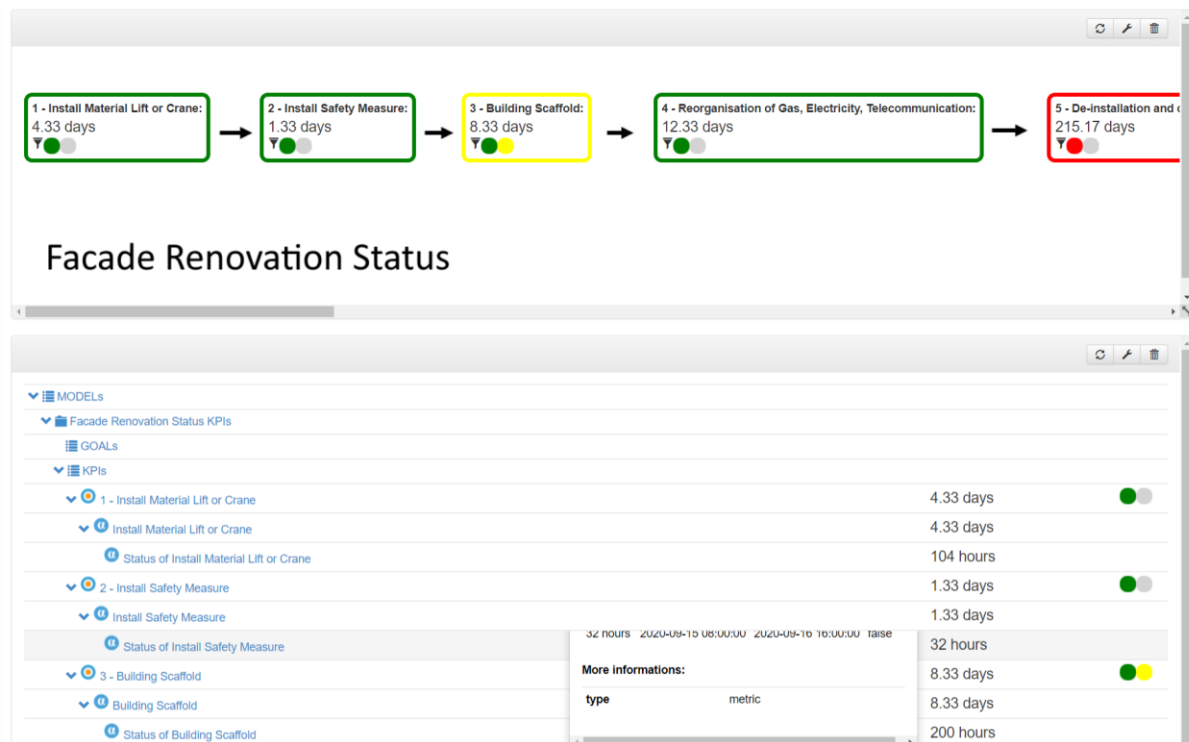


Figure 20 - Facade Renovation Status KPI Dashboard

A second process view has been also introduced to focus on the status of the process as returned by the execution engine. This is different from the first process view because it allows to visually check whether the activities are aligned with their scheduled times or not.

Such complex scenarios require complex modelling and knowledge externalization in the design phase, and hence may only be appropriate for the specific dashboard. A simple simulation of one renovation process for the simulation of one KPI will be introduced in the next section.

3.1.2 Models-based Monitoring Dashboards Architecture

The KPI dashboard is a component that performs a union between models and data, resulting in a customizable web dashboard. The models contain information about how to retrieve and combine data to create metrics, and how to use such metrics to evaluate KPIs and goals. The data are external and obtained through specific microservices, created with the Olive framework, able to connect with different types of data sources. In

the end, the results are displayed using a widget-based interface that is able to display the KPIs and metrics in different formats depending on the domain and the user experience.

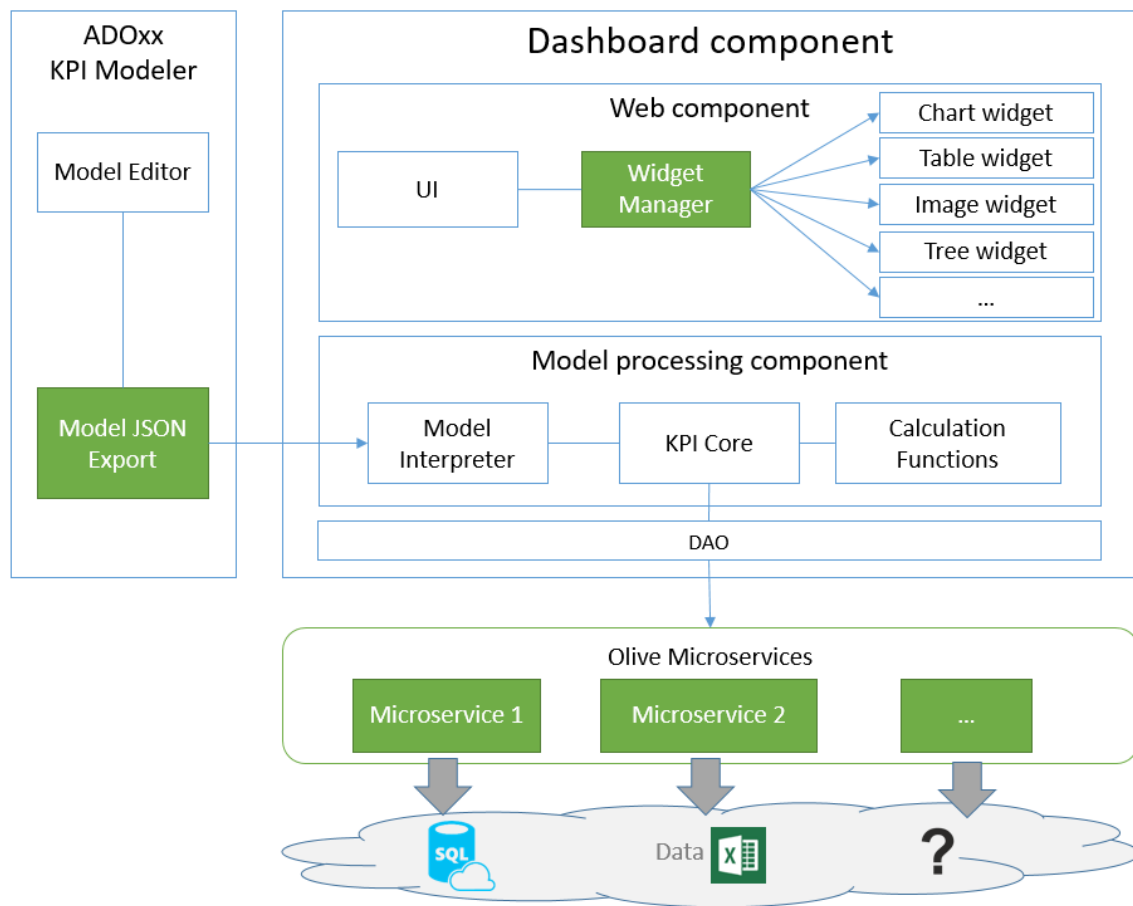


Figure 21 - KPIs Dashboard architecture

The dashboard accepts KPIs models provided in a specific JSON format. The Renovation process KPIs design tool provides a feature to export the KPIs and data model in a specific JSON format. This model is processed by the model interpreter component that has the responsibility to create an internal representation of the model collecting all the dependencies, the calculation methods, and the data sources for all the KPIs, goals and metrics in the model.

Such model information is processed by the KPI core component that is responsible to call the microservices described in the model to retrieve the data, apply the calculation functions as described in the model and make the resulting values available through the

user interface. The interaction of the KPI core with the different microservices is helped by a Data Access Object (DAO) component that provide also caching features and an optimized microservice communication. The functions are evaluated in a component that interprets and validates them to avoid security issues.

When the evaluated KPIs and Goals are available, they are provided to the UI component upon widget request. The widget manager component is responsible to manage the interface for the creation and configuration of the different widgets giving them also access to the calculated data.

The dashboard provides out of the box the four most used widgets, but extensions are available through a plug-in based mechanism:

- **Chart widget:** can visualise a KPI value in a Cartesian chart. The user can configure the type of chart to use, choosing between horizontal bar chart, vertical bar chart, line chart, curve chart and radar chart and after that he can specify which KPI field is visible for every axis and an optional threshold line. For example the “Simulated Scaffold Optimistic Cost” KPI containing the data fields “cost” and “instant” can be visualised in a line chart selecting for the X axis the “instant” field and for the Y axis the “cost” field.

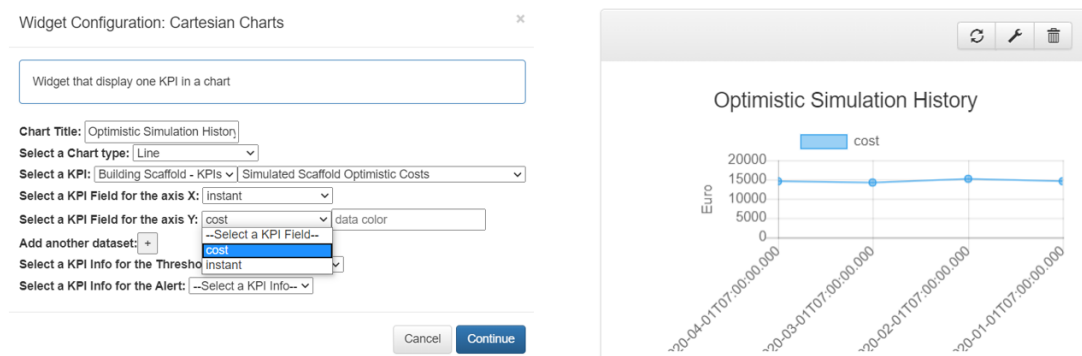


Figure 22 - KPI Dashboard Chart Widget

- **Table widget:** this widget allows to visualise all the details of a configured KPI in a table format. All the values will be visualised, along with evaluation of the KPI status with a green, yellow, red indicator.

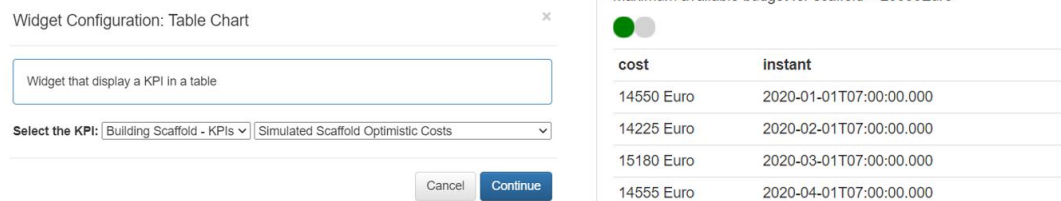


Figure 23 - KPI Dashboard Table Widget

- **Image widget:** This widget allows to overlay one or more KPI details over an image. The KPIs are displayed with a color code that reflects the KPI status and details are visualised when hovering with the mouse over the indicator.

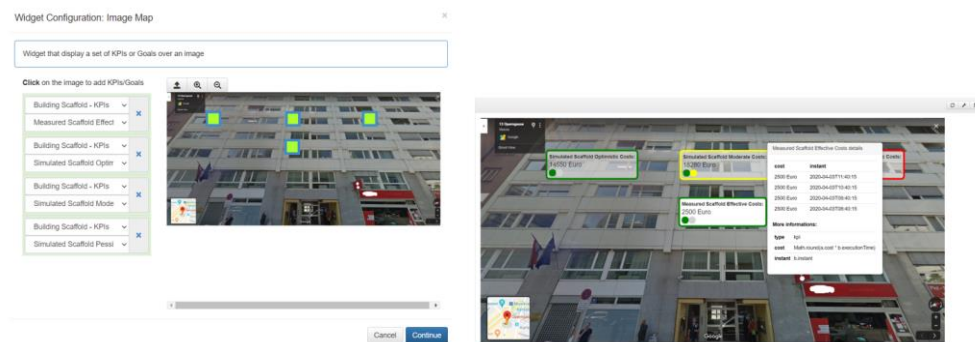


Figure 24 - KPI Dashboard Image Widget

- **Tree widget:** This widget allows to visualise all KPIs in a collapsible tree view organised hierarchically or linearly on their dependencies. In this way it is possible to identify the root cause of a problematic KPI or goal simplifying the behaviour analysis.

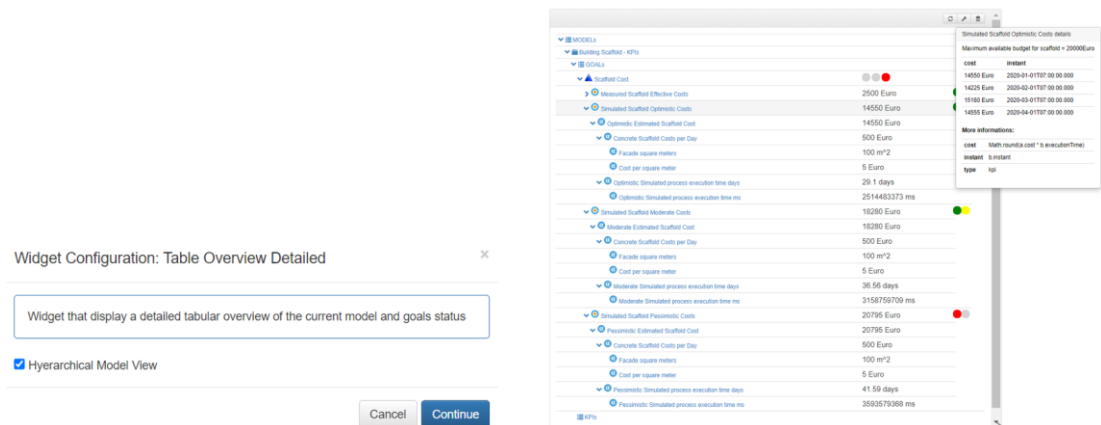


Figure 25 - KPI Dashboard Tree Widget

At the end, the UI component is responsible for the rendering of the configured widgets in a docked layout that the user can resize and move the widgets around the page with a drag and drop mechanism.

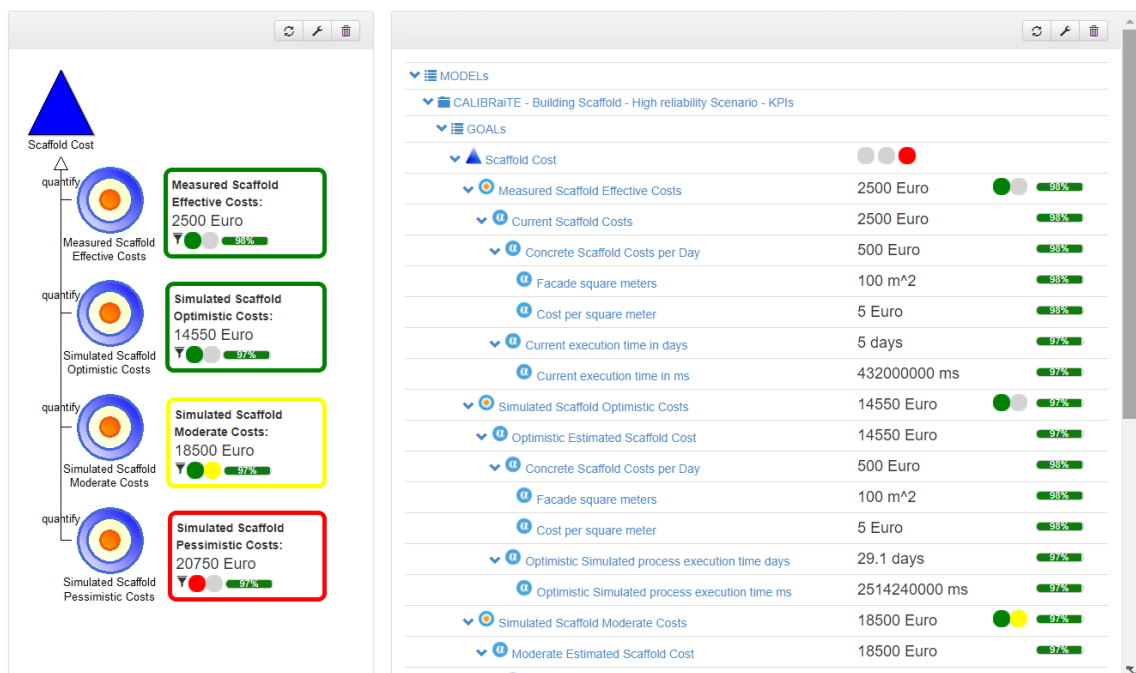


Figure 26 - KPI Dashboard Widgets Combined.

All the widgets allow also to visualise in a similar way all the trustability related indicators. The overall trustability score is represented as a percentage bar of green, yellow, and red color in the cases of high, medium and low the trustability score respectively. Clicking over this score a popup will appear to visualise further details.

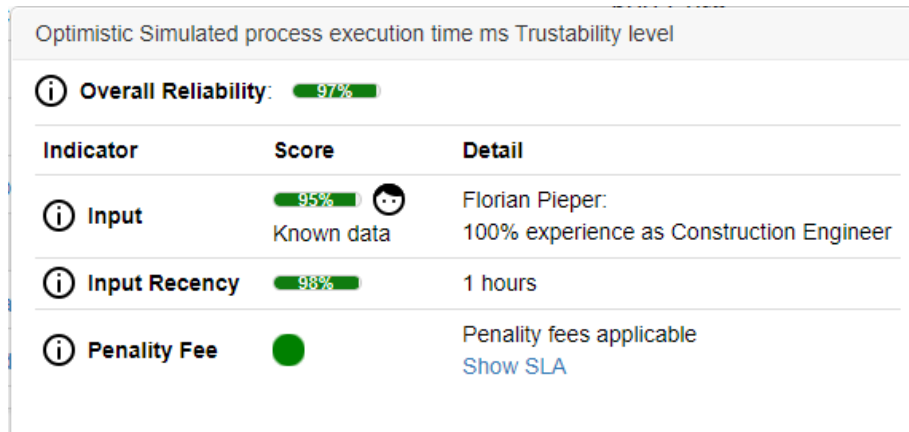


Figure 27 - KPI Trustability Details.

It is possible to distinguish the three main indicators affecting the trustability score: (1) the type of the KPI input data, that also contains information about the user expertise, (2) the input data recency showing also the details of the data delay and (3) the SLA Penalty presence with details on the referenced SLA.

3.2 SIMULATION TOOLS FOR RENOVATION PROCESSES

This section demonstrates the knowledge-based simulation for the renovation process first and then provides a view on the architecture of the tool.

3.2.1 *Simulation Tool Demonstration Use Case*

The simulation tool requires as input, first, the renovation process workflow in standard BPMN format. The process to be simulated can be exported in BPMN format directly from the renovation process and workflow design tool. Additionally, an Excel sheet containing times, costs and decision probabilities is required.

Please select the file containing the model to simulate and press the Simulate button. Supported file format is BPMN.



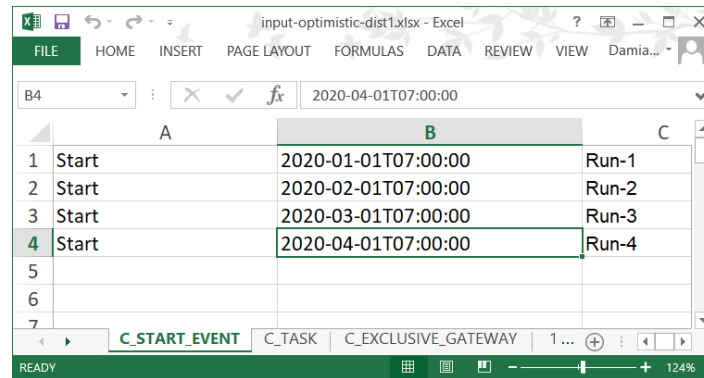
Figure 28 - Renovation Process Simulation Inputs

The Excel input file must be created according to the details described in BIMERR deliverable D6.3.

In particular, the Excel must contain the following 3 sheets in the proposed order:

1. C_START_EVENT
2. C_TASK
3. C_EXCLUSIVE_GATEWAY

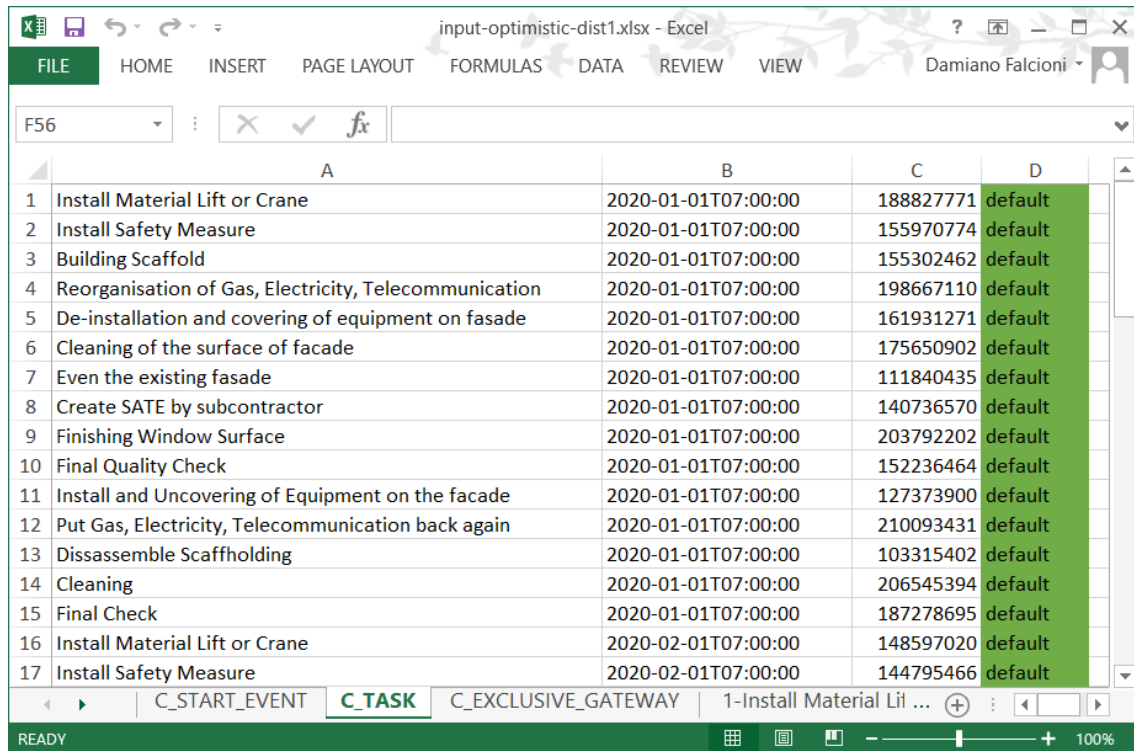
The C_START_EVENT sheet provides information on the number of simulations to run with their starting time and unique identifier. The first column of this sheet must contain the name of the starting event as reported in the BPMN model. The second column must contain the starting time of this specific event while the third column includes a unique id to associate it to the simulation run and to serve later as a reference in the simulation results.



	A	B	C
1	Start	2020-01-01T07:00:00	Run-1
2	Start	2020-02-01T07:00:00	Run-2
3	Start	2020-03-01T07:00:00	Run-3
4	Start	2020-04-01T07:00:00	Run-4
5			
6			
7			

Figure 29 - Renovation Process Simulation Input C_START_EVENT

The C_TASK sheet contains the execution time of every activity in our renovation process. The first column must contain the activity name as reported in the BPMN model. The second column contains an optional starting time that can be used to provide an additional waiting time before the activity starts. By default, the activity starts as soon as the previous one is terminated. The third column represents the execution time expressed in milliseconds. This value can be provided directly but the great advantage of using Excel as input source is that you can calculate this value combining different factors together. Additional sheets are used for this scope. The fourth column can contain the unique id of the simulation run to be used or the defaults value if the activity is valid for every simulation run.



	A	B	C	D
1	Install Material Lift or Crane	2020-01-01T07:00:00	188827771	default
2	Install Safety Measure	2020-01-01T07:00:00	155970774	default
3	Building Scaffold	2020-01-01T07:00:00	155302462	default
4	Reorganisation of Gas, Electricity, Telecommunication	2020-01-01T07:00:00	198667110	default
5	De-installation and covering of equipment on fasade	2020-01-01T07:00:00	161931271	default
6	Cleaning of the surface of facade	2020-01-01T07:00:00	175650902	default
7	Even the existing facade	2020-01-01T07:00:00	111840435	default
8	Create SATE by subcontractor	2020-01-01T07:00:00	140736570	default
9	Finishing Window Surface	2020-01-01T07:00:00	203792202	default
10	Final Quality Check	2020-01-01T07:00:00	152236464	default
11	Install and Uncovering of Equipment on the facade	2020-01-01T07:00:00	127373900	default
12	Put Gas, Electricity, Telecommunication back again	2020-01-01T07:00:00	210093431	default
13	Dissassemble Scaffolding	2020-01-01T07:00:00	103315402	default
14	Cleaning	2020-01-01T07:00:00	206545394	default
15	Final Check	2020-01-01T07:00:00	187278695	default
16	Install Material Lift or Crane	2020-02-01T07:00:00	148597020	default
17	Install Safety Measure	2020-02-01T07:00:00	144795466	default

Figure 30- Renovation Process Simulation Input C_TASK

In case of choices the C_EXCLUSIVE_GATEWAY sheet must be filled. This allows to specify for every choice in the BPMN process, the probability to use during the simulation. The first column in this case must contain the exclusive gateway name as reported in the BPMN process; the second column contains an optional waiting time to postpone the choice execution and the third column provide information about the probability value of all its outgoing sequence flows.

As introduced previously, additional sheets can be present to define the execution time for every task, combining different indicators and risk factors. Different approaches can be used to combine the risk factors as described in D6.3 (BIMERR Consortium,2020). An example is the weighted combination of the normal distribution of five different factors: (1) the normally estimated average task time, (2) the probability of the delay due to payment problems, (3) the delay introduced by bad weather forecast, (4) the delay introduced by sub-contractor's problems, and (5) the delay introduced by unexpected events.

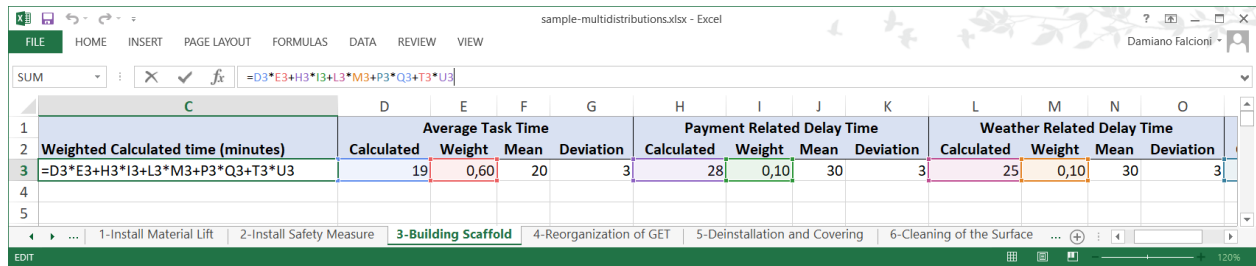


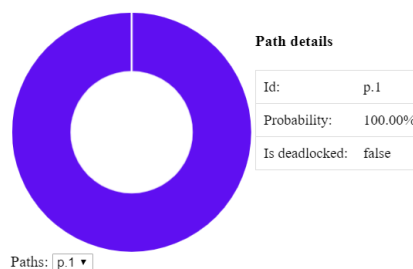
Figure 31 - Renovation Process Simulation Input Calculation

As soon as both inputs are ready the simulation can start. Once completed, the results are visualised in two different forms: an overview of times and cost with path probabilities and generic information of the process, and a detailed view in the form of an execution log that can be used also to perform a comparison with real workflow execution.

General results

Measure	Details
Average Cost:	0.00
Max Cost:	0.0 Trace: t.1
Min Cost:	0.0 Trace: t.1
Total Costs:	0.00
Average Executions Time:	00:024:20:31:23
Max Executions Time:	00:036:13:25:59 Trace: t.1
Min Executions Time:	00:033:20:56:20 Trace: t.1
Total Executions Time:	00:138:20:18:29
Total Runs:	4
Total Traces:	1
Total Paths:	1

Paths Infos



Paths: p.1

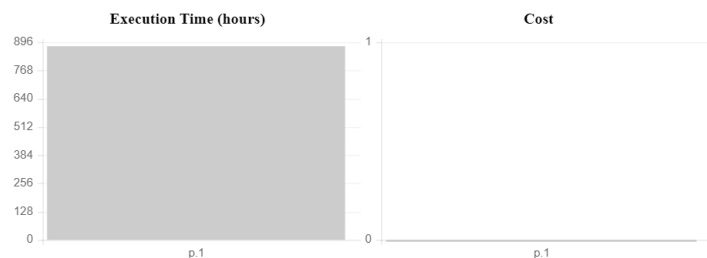


Figure 32 - Renovation Process Simulation General Results

The general results contain the following data:

Name	Measure	Details
Average Cost	Average cost during all the simulation runs	-

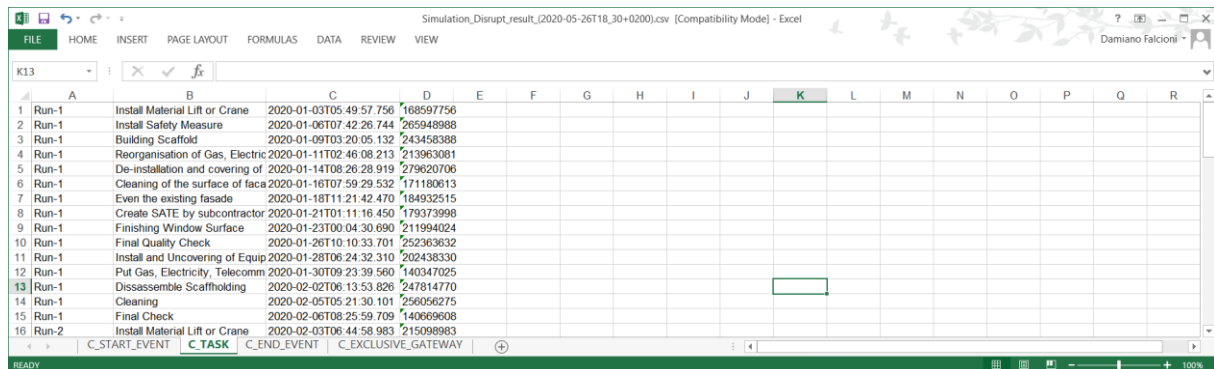
Name	Measure	Details
Max Cost	Max cost during all of the simulation runs	Trace name that contains this cost
Min Cost	Min cost during all of the simulation runs	Trace name that contains this cost
Total Costs	Sum of all costs during all the simulation runs	-
Average Executions Time	total execution time / total simulation runs number	-
Max Executions Time	Max execution time during all of the simulation runs	Trace name that contains it
Min Executions Time	Min execution time during all of the simulation runs	Trace name that contains it
Total Executions Time	Sum of all execution times during all of the simulation runs	-
Total Runs	Number of simulations runs	-
Total Traces	Number of Petri Net traces passed through each simulation run	-
Total Paths	Number of Petri Net places passed through each simulation run	-

Table 1 - Renovation Process Simulation General Results Details

The detailed results are in form of Excel sheet following the same structure with the input data, but also including details on the simulated starting and execution times.

The C_TASK sheet contains in the first column the id of the run involved, in the second column the name of the task performed and in the third column the simulated starting time with the actual execution time in the fourth column.

Additionally, a C_END_EVENT sheet is present that contains the ending time for every started simulation.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Run-1	Install Material Lift or Crane	2020-01-03T05:49:57.756	168597756														
2	Run-1	Install Safety Measure	2020-01-06T07:42:26.744	265948988														
3	Run-1	Building Scaffold	2020-01-09T03:20:05.132	243458388														
4	Run-1	Reorganisation of Gas, Electric	2020-01-11T02:48:08.213	213963081														
5	Run-1	De-installation and covering of	2020-01-14T08:26:28.919	279820706														
6	Run-1	Cleaning of the surface of facade	2020-01-16T07:59:29.532	171180613														
7	Run-1	Even the existing facade	2020-01-18T11:21:42.470	184932515														
8	Run-1	Create SATE by subcontractor	2020-01-21T01:11:16.450	179373998														
9	Run-1	Finishing Window Surface	2020-01-23T00:04:30.690	211994024														
10	Run-1	Final Quality Check	2020-01-26T10:10:33.701	252363632														
11	Run-1	Install and Uncovering of Equip	2020-01-28T06:24:32.310	202438330														
12	Run-1	Put Gas, Electricity, Telecomm	2020-01-30T09:23:39.560	140347025														
13	Run-1	Dissassemble Scaffolding	2020-02-02T06:13:53.826	247814770														
14	Run-1	Cleaning	2020-02-05T05:21:30.101	256056275														
15	Run-1	Final Check	2020-02-06T08:25:59.709	140699608														
16	Run-2	Install Material Lift or Crane	2020-02-03T06:44:58.983	215098983														

Figure 33 - Renovation Process Simulation Detailed Results

3.2.2 Simulation tool architecture

The BIMERR Renovation Process Simulation provides a fast and extendible service able to simulate renovation process executions. The service uses the Petri Net logics to simulate processes and workflow provided in BPMN2.0 formats and is flexible enough to support the simulation of other kind of models through the definition of their appropriate mapping rules to Petri Net. The service is provided as REST API with a graphical HTML client that shows the results in a user-friendly way.

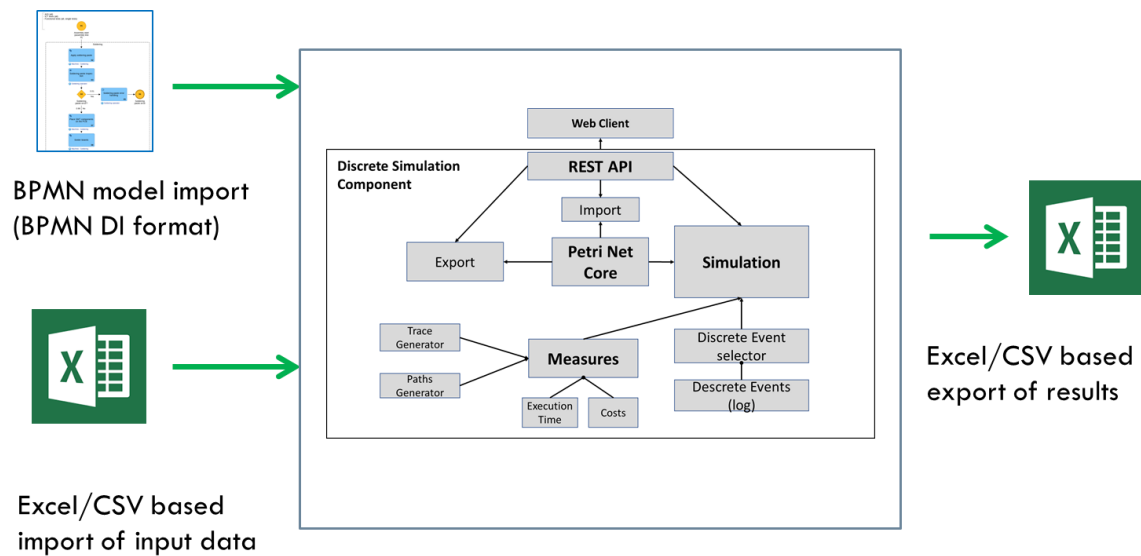


Figure 34 - Renovation Process Simulation Engine Architecture

Following are the descriptions of the main components of the simulation tool:

The ***petri net core module*** is the component that contains the main logic of a petri net and manages its semantics. The simulation service uses this component to evaluate at each step which transition can be enabled.

The ***import module*** is an easy to extend component that automatically recognizes the format of the provided model and converts it into the internal petri net structure. It manages separately the logic of document's parsing and object's mapping to reuse the same mapping logic for multiple file formats (like in the case of BPMN and ADOxx BPMN). This is also responsible to associate the input from the Excel sheet to the right BPMN object. More details on the work done in BIMERR in the context of mapping the BPMN into a petri-net is available in the Annex section **BPMN Mapping to Petri Net**.

The ***export module*** is for diagnostic purposes only. It gives the possibility to export the internal petri net structure in Petri Net Modeling Language (PNML) standard format, to be visualised in any supported editor.

The ***simulation measures module*** is an easy to extend component that allows the definition of listeners for the simulation event. Each listener produces a measure or a result from a single simulation, like a trace, a path, the waiting times, or the execution

costs. The resulting indexes can then be collected in a special container to calculate some final indexes (like average values).

The ***discrete event selector module*** is the component that performs the selection of the transition to be executed among the available ones. The module provides a base mechanism that performs a fair choice between parallel transitions and a user defined probabilistic choice between concurrent transitions. The base mechanism has been also extended to support dynamic probability evaluation using a scripting system.

The ***simulation module*** is the component that manages all the simulations, invoking the functionalities of the simulation measures module and the transition choice. It is also responsible for the generation of the simulation output in a structured format.

The ***REST API*** is the module responsible to expose the simulation component features to the external world. It is responsible for processing the simulation parameters input and generating the simulation results in the service specific output format.

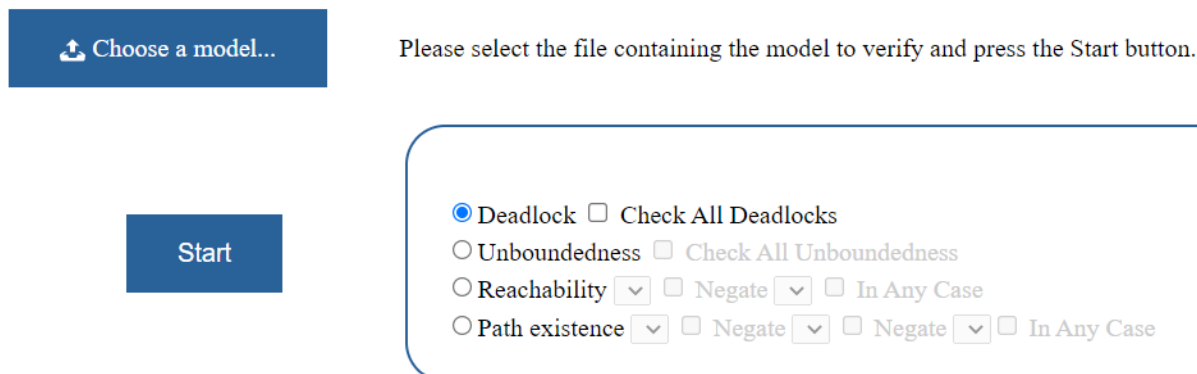
The ***Web Client*** is a simple web interface that allows the user to simulate a BPMN model and visualise the simulation results in terms of charts and tables, communicating directly with the REST API.

3.3 FORMAL VERIFICATION TOOLS FOR RENOVATION PROCESSES

This section demonstrates the formal verification of the renovation process first and then provides a view on the architecture of the tool.

3.3.1 Formal Verification Tool Demonstration Use Case

The simulation tool requires as input first the renovation process workflow in standard BPMN format. The process to be formally verified can be exported in BPMN format directly from the renovation process and workflow design tool. As soon as the model is selected, it will be automatically loaded and then the user will be able to select the desired verification capability and activate it by clicking on the Start button.



Choose a model...

Please select the file containing the model to verify and press the Start button.

Start

- ☒ Deadlock ☐ Check All Deadlocks
- ☐ Unboundedness ☐ Check All Unboundedness
- ☐ Reachability ☐ Negate ☐ In Any Case
- ☐ Path existence ☐ Negate ☐ Negate ☐ In Any Case

Figure 35 - Formal Verification Input

The application allows to perform four kinds of formal verification: Deadlock analysis, Unboundedness analysis, Reachability analysis and Path existence detection. The results are reported as statements reporting the validity of the verified condition and in case of availability a list of counterexamples that prove the evaluation.

Deadlock analysis

The deadlock analysis allows to check if one or more deadlocks exist in the model, that prevent the correct termination of the process. If the “Check All Deadlock” option is selected it will identify all the paths that bring to a deadlock, otherwise only the first one occurring will be found. Checking all deadlocks at once may result in the model checking

state explosion problem⁶, especially if the analysis is performed on big workflow models, resulting in the saturation of all the machine memory resources. Finding only one deadlock at once can help to solve or minimize this problem.

Facade Improvement Process.bpmn

Please select the file containing the model to verify and press the Start button. Supported file formats are BPMN, ADOxx XML, ADO XML Light and PNML.

Verification Parameters

☒ Deadlock ☐ Check All Deadlocks
☐ Unboundedness ☐ Check All Unboundedness
☐ Reachability ☐ def_24517 ☐ Negate ☐ Cleaning ☐ In Any Case
☐ Path existence ☐ def_24517 ☐ Negate ☐ Cleaning ☐ In Any Case

Start

Select Model

Verification Results	
Status:	OK
Description:	The property "Deadlock absence" is TRUE!
Critical Activities:	<input checked="" type="checkbox"/> Show Full Counterexample

Figure 36 - Deadlock Analysis

The verification results are visualised as a sentence reflecting the semantics of the Computation Tree Logic (CTL) formula verified. In case of the deadlock analysis, the "Deadlock absence" property results will be TRUE or FALSE, respectively when the model is free from deadlock or not. When one or more deadlocks are detected, the analysis will also report the activities directly involved, along with the ability to see the full process path resulting in that activity, to help the resolution of the problem.

Unboundedness analysis

The unboundedness analysis allows to check if the process can get into an infinite loop from where it cannot be recovered. Also in this case, the check allows to verify that the process ends correctly in every possible situation. Like in the deadlock detection, also this analysis allows to search for all the unboundedness at once or, in case of big processes, to identify the unboundedness one by one to minimize the state explosion problem.

⁶ https://link.springer.com/chapter/10.1007/978-3-642-35746-6_1

Facade Improvement Process.bpmn

Please select the file containing the model to verify and press the Start button. Supported file formats are BPMN, ADOxx XML, ADO XML Light and PNML.

Start

Verification Parameters

☐ Deadlock ☐ Check All Deadlocks
☒ Unboundedness ☒ Check All Unboundedness
☐ Reachability ☐ Negate ☐ In Any Case
☐ Path existence ☐ Negate ☐ In Any Case

Select Model

Verification Results	
Status:	OK
Description:	The property "Unboundedness absence in all the model" is TRUE!
Critical Activities:	<input checked="" type="checkbox"/> Show Full Counterexample

Figure 37 - Unboundedness Analysis

The verification results will be TRUE or FALSE in case the CTL formula associated to the sentence "Unboundedness absence" is valid or not. When one or more livelocks are detected, the analysis will also report the activities directly involved with the possibility to see the full process path resulting in that activity, to help the resolution of the problem.

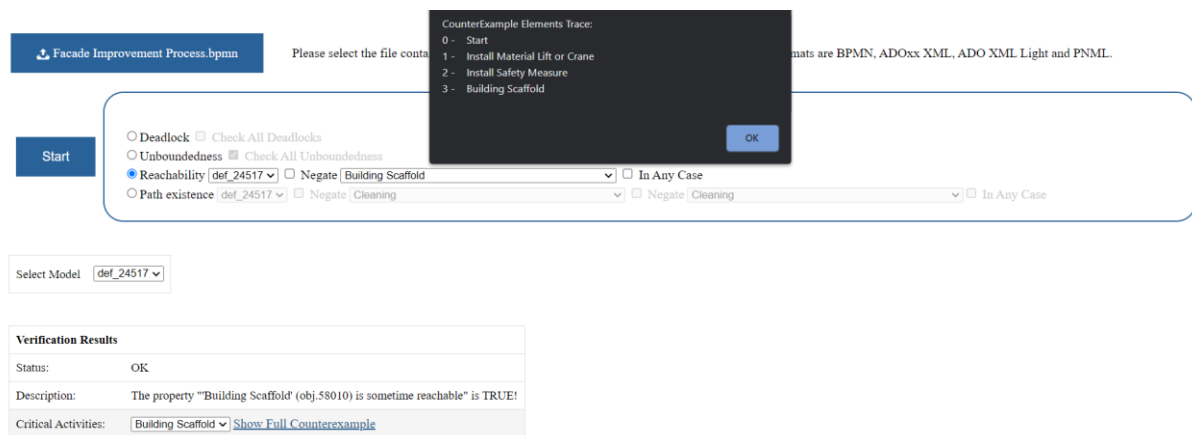
Reachability analysis

The reachability analysis allows to check if a specific activity is reachable or not at some point in the process. This analysis will not detect structural problems automatically like in the deadlock and unboundedness cases, but, depending on the specific process, it will allow to check if critical activities are correctly executed.

This analysis allows to indicate the specific activity to analyse, by selecting it from all the available processes, and on this activity perform the reachability or not reachability check (using the "Negate" checkbox) for every possible situation or for at least one (via the "In any case" checkbox). So, based on the combination of such parameters, the reachability analysis allows to effectively perform four kinds of checks:

- The activity is always reachable: any path in the process brings to that activity.
- The activity is sometimes reachable: at least one path in the process brings to that activity.
- The activity is always not reachable: any path in the process does not bring to that activity.

- The activity is sometimes not reachable: at least one path in the process does not bring to that activity.



Verification Results	
Status:	OK
Description:	The property "Building Scaffold" (obj.58010) is sometime reachable" is TRUE!
Critical Activities:	Building Scaffold Show Full Counterexample

Figure 38 - Reachability Analysis

The analysis returns TRUE or FALSE in case the specific property is verified or not. In case a counterexample is available it is visualised as the sequence of activities that prove the theory. In the example in Figure 38 the “Building Scaffold” activity is checked on whether it is sometimes reachable. This is true because, as the counterexample says, you can reach it following the sequence of activities “Install Material Lift or Crane” and “Install Safety Measures”.

Path existence analysis

This analysis is similar to the reachability because it does not automatically check a structural problem but allows the user to analyze the behaviour of the process on specific critical activities checking whether an execution path exists between the two of them.

This analysis allows to select the starting and ending activities of the path to analyse, selecting it from all the available processes, with the possibility to negate both (using the “Negate” checkbox). The negation allows to search for the existence of paths that do not start or end with the selected activity. Additionally, it is possible to check the path existence in every possible scenario (via the “In any case” checkbox) or in at least one.

So, based on the combination of such parameters, the path analysis allows to effectively perform eight different kinds of checks:

- Existence of at least one path from the start to the end activities.
- Existence of at least one path that does not start with the first selected activity but does end with the second selected activity.
- Existence of at least one path that does start with the first selected activity but does not end with the second selected activity.
- Existence of at least one path that does not start with the first selected activity and does not end with the second selected activity.
- Every path goes from the start to the end activities.
- Every path does not start with the first selected activity but does end with the second selected activity.
- Every path starts with the first selected activity but does not end with the second selected activity.
- Every path does not start with the first selected activity and does not end with the second selected activity.

Facade Improvement Process.bpmn

Please select the file containing the model to verify and press the Start button. Supported file formats are BPMN, ADOxx XML, ADO XML Light and PNML.

Start

Verification Parameters

☐ Deadlock ☐ Check All Deadlocks
☐ Unboundedness ☐ Check All Unboundedness
☐ Reachability ☐ def_24517 ☐ Negate ☐ Building Scaffold ☐ In Any Case
☒ Path existence ☐ def_24517 ☐ Negate ☐ Building Scaffold ☐ Negate ☐ Disassemble Scaffolding ☒ In Any Case

Select Model

def_24517

Verification Results	
Status:	OK
Description:	The property "For every path where happens 'Building Scaffold' (obj.58010), then happens 'Disassemble Scaffolding' (obj.58013)" is TRUE!
Critical Activities:	<input checked="" type="checkbox"/> Show Full Counterexample

Figure 39 - Path Existence Analysis

The analysis returns TRUE or FALSE depending on whether the specific property is verified or not. In case a counterexample is available it is visualised as the sequence of activities that prove the theory. In the example in Figure 39, it is checked whether every path that starts from the “Building Scaffold” activity is reaching the “Disassemble Scaffold” activity in order to guarantee that the scaffold will be always disassembled after its assembly.

3.3.2 Verification Tool Architecture

The renovation process verification tool is integrated in the simulation component and reuses some of its modules, in particular the ones related to the input processor and the petri net core. This is possible because the formal verification tool relies on the same model representation logic used in the simulation component, based on petri-net and obtained through an automatic mapping procedure from the BPMN renovation process. In this specific case the petri-net model instead of being executed is exported in the LOLA⁷ specific format for formal model checking specific properties expressed in CTL.

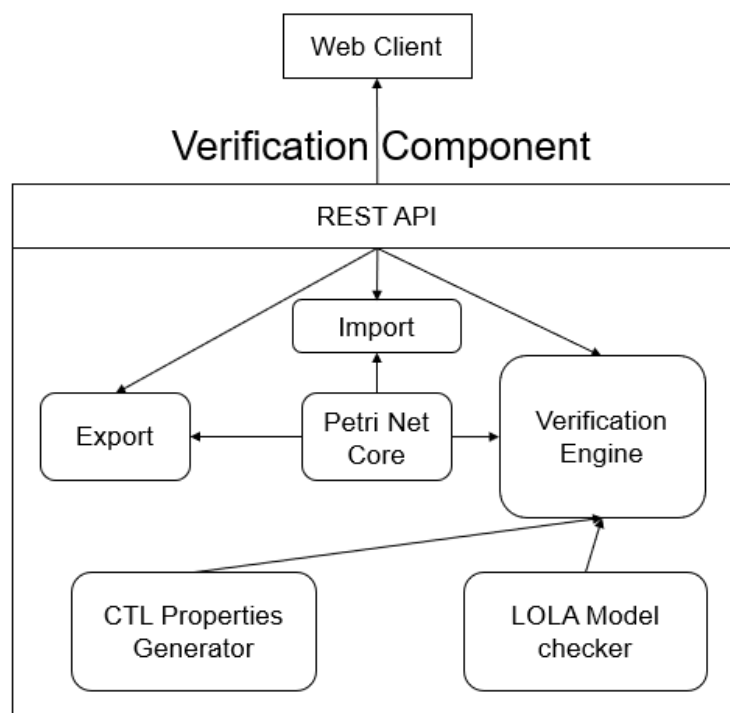


Figure 40 - Formal Verification Architecture

The descriptions of the main modules composing the verification component is provided in the following paragraphs.

The **petri net core module** is used to store an internal representation of the model in petri-net specific semantics. It is responsible to manage the model provided as input and

⁷ <https://theo.informatik.uni-rostock.de/theo-forschung/tools/lola/>

make them available to the needed modules, in particular to the export and verification engine modules.

The ***import module*** is an easy to extend component that automatically recognizes the format of the provided model and converts it in the internal petri net structure using state of the art techniques described in Annex, section BPMN Mapping to Petri Net. It manages separately the logic of document parsing and of object mapping to reuse the same mapping logic for multiple file format (like in the case of BPMN and ADOxx BPMN).

The ***export module*** contains the logic to translate the internal petri-net model in the specific file format Extended Backus-Naur form (EBNF) required by the LOLA model checker. Additionally, it is used for diagnostic purposes providing the capability to export the petri-net in PNML standard format in order to be visualised in any supported editor.

The ***verification engine module*** is responsible to communicate with the LOLA Model checker and initialise both the petri-net export in the specific format required by LOLA and the generation of the CTL property to verify the processing of the petri-net model. Additionally, the module will capture the LOLA output identifying counterexamples and properties verification results.

The ***LOLA Model Checker module*** contains the Windows, Linux and macOS executables of LOLA, an open source, state-of-the-art and multi award winning⁸ model checker for petri-net that allows to verify properties expressed in CTL.

The ***CTL Property Generator module*** allows to generate specific Computation Tree Logic properties in the format required by the LOLA model checker. The generation takes also into account elements from the petri-net model that may be required by the specific property (e.g., for the reachability analysis a specific model activity is required). More details on the work done in BIMERR in the context of mapping the property to verify into a CTL formula is available in the Annex section **Properties to Computation Tree Logic Mapping**.

⁸ <https://mcc.lip6.fr/>

The **REST API** is the module responsible to expose the verification component features to the external world. It is responsible for processing the verification parameters inputs and generating the verification results in the service specific output format.

The **Web Client** is a simple web interface that allows the user to select a BPMN model and the property to verify and visualises the verification results in terms of sentences, communicating directly with the REST API.

4. REFLECTION AND INNOVATION TOOLS FOR RENOVATION PROCESSES

In this chapter the set of components for workflow log analysis and renovation process collaboration are described.

4.1 PROCESS MINING OF RENOVATION PROCESS

Process mining is used to support the analysis and evaluation of business processes. Trends and patterns in the process data are interesting for the improvement of processes. Therefore, data mining algorithms are applied on the process data. Not only should the efficiency of processes be improved by process mining, but also the understanding, especially dependencies and interconnections. It might not only be necessary to improve specific tasks regarding their execution time, as sometimes a restructuring of the whole process is more reasonable. For mining the renovation processes, the free process mining platform Celonis⁹ was used. In the following subsection, a description of the preparations, the creation of an analysis workspace and the results are provided based on our outside facade renovation process sample.

Celonis is a process mining platform that allows to analyze log files and construct custom analytical dashboards. Its free version Celonis Snap¹⁰ can be used after registration to their portal and the whole platform is available as a cloud application. The process to provide log files to analyze and obtain back the results is now manual. Currently the log generated by the workflow engine requires some manual processing in order to be accepted as valid inputs for Celonis Snap.

⁹ <https://www.celonis.com/>

¹⁰ <https://www.celonis.com/solutions/celonis-snap>

4.1.1 Logs preparation for Celonis

Celonis requires one single CSV file as input. The results of the workflow must be combined in one Excel worksheet that, afterwards, must be converted to a CSV file with field separator “;” so that Celonis can parse the entries. The CSV parser of Celonis must be configured as follows:

- Input type: CSV
- Field separator: “;”
- Header row: “unchecked”
- Date format: ‘yyyy-MM-dd'T'hh:mm:ss’

In a second step Celonis requires to associate a specific semantic meaning to the CSV columns to interpret its data. This association must be performed as follows:

- Assign ‘Case ID’ to the first column, which holds the information.
- Assign ‘Activity Name’ to the second column.
- Assign ‘Timestamp’ to the third column.

After this assignment, the data is ready to be analyzed.

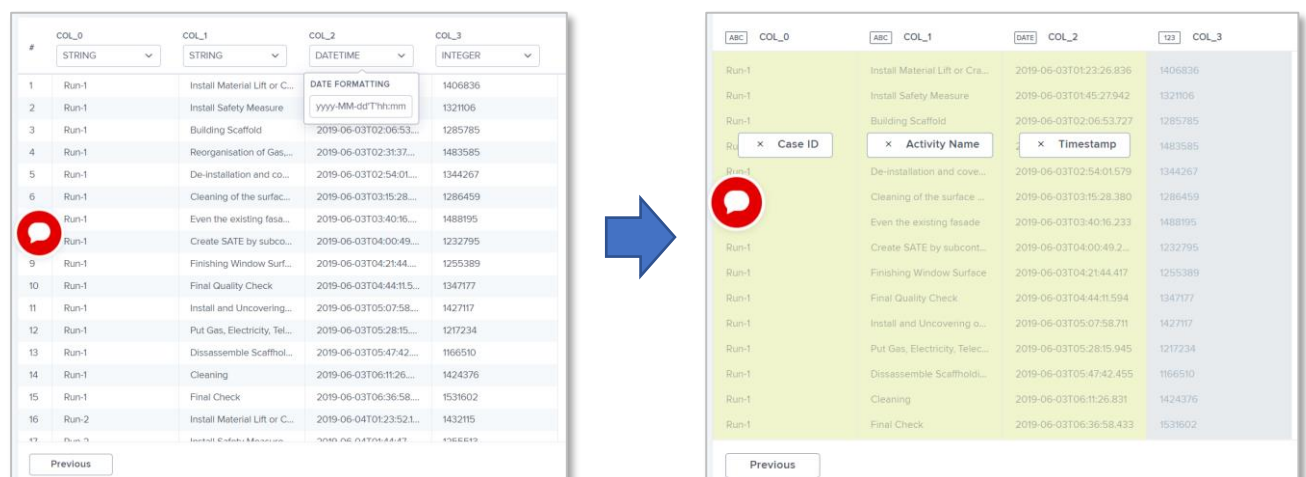


Figure 41 - Celonis Logs Preparation

4.1.2 Creation of Analysis Workspace

First, a new workspace for analysis must be created. This procedure can take place in the process analytics space where all the defined workspaces are available. Here the user can create a new workspace starting from the previously uploaded logs. A new analytics workspace allows by default to visualise the process obtained from the log analysis, including their variants and statistics. This workspace can be customized using configurable widgets and custom table analytics done using a proprietary SQL-like language named Publisher Query Language (PQL) used to combine and analyze tables resulting from the logs.

For the BIMERR use case in particular a workbench view with the following widgets has been created:

- **Process Explorer widget:** allows to visualise the workflow generated from the logs, including all the variants. It allows to compare it with the original workflow to identify incorrect actions.
- **Cases Variants:** A table widget that allows to visualise all the process variants listing the tasks of each one. This table must be constructed with the following columns:
 - **Case ID:** Containing the formula
`"_CEL_CSV_ACTIVITIES_CASES"."_CASE_KEY"`
 - **Variant:** Containing the formula
`VARIANT("_CEL_CSV_ACTIVITIES"."ACTIVITY_EN")`
- **Frequencies:** A table widget that allows to visualise the frequency of every choice. This is useful to identify if the data used for the process simulation were correct or need alignments. This table must be constructed with the following columns:
 - **Source Activity:** With the formula
`SOURCE("_CEL_CSV_ACTIVITIES"."ACTIVITY_EN")`
 - **Target Activity:** With the formula
`TARGET("_CEL_CSV_ACTIVITIES"."ACTIVITY_EN")`
 - **Frequency:** Containing
`COUNT(TARGET("_CEL_CSV_ACTIVITIES"."ACTIVITY_EN"))`

- **Execution Times:** A table widget that allows to visualise the average execution time of every task. This table must be constructed with the following columns:
 - **Activity:** Using the formula `SOURCE("_CEL_CSV_ACTIVITIES"."ACTIVITY_EN")`
 - **Average Execution Time:** Using the formula `AVG(DATEDIFF(ms, SOURCE("_CEL_CSV_ACTIVITIES"."EVENTTIME"), TARGET("_CEL_CSV_ACTIVITIES"."EVENTTIME")))`

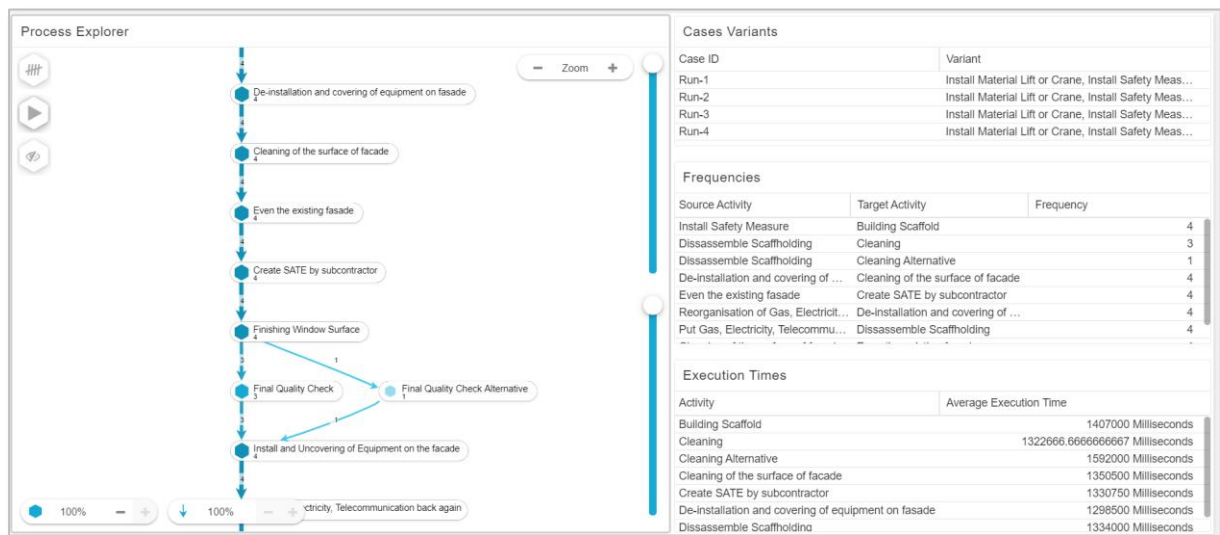


Figure 42 - Celonis Analysis Workspace for BIMERR

To enable the exports of the results, the created workspace needs to be configured. In the Edit mode the user can access the Analysis setting in the menu and turn on the checkboxes named “Allow excel and csv export of analysis components” and “Allow BPMN export of the process explorer”. This enables a right click menu entry on every widget that allows to generate a BPMN file for the process to be imported in the modelling environment and to be compared with the original workflow, and to generate CSV files for every one of the tree tables available in our workspace.

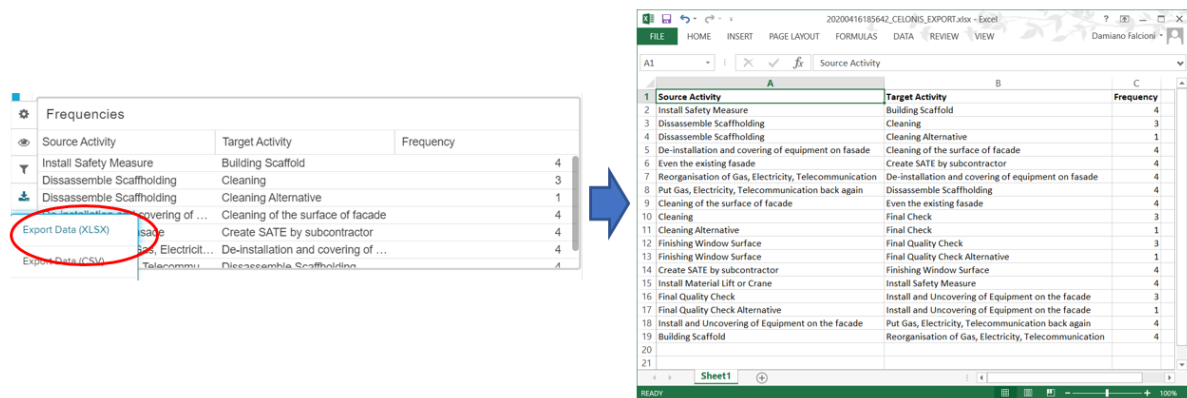


Figure 43 - Excel Output

4.2 COLLABORATIVE REFLECTION OF RENOVATION PROCESS

Beginning with people working in collaboration, the so-called collective intelligence can emerge. Due to interaction and competition, the group has an increased problem-solving capacity. The change of finding a solution within the group is much higher compared to a single person tackling the issues of interest. Especially, the principle of agreeing on reasonable approaches and avoiding critical ideas applies in collective intelligence. The final goal should be consensus decision making. Network effects between distributed data, knowledge, software applications, computing capabilities and experts are used to build up the collective intelligence. Feedback and continuous improvements and learnings are considered in real time. To structure the way of working and to put the ideas and comments on record, social media or other contribution systems might be used. For instance, Wikipedia¹¹ is one platform widely known for collective intelligence, as it allows easy exchange of knowledge, ideas, and thoughts.

As already mentioned, collective intelligence is beneficial for solving problems and finding improvements. For this reason, a model wiki based on XWiki¹² allows commenting models and retrieving comments.

4.2.1 Model Wiki Application

The Model Wiki web application allows to generate xWiki pages from any model in the ADOxx modelling environment and as soon as the pages are generated, it enables the import of any existing comments in the wiki back to the model.

The user must first have a model available in the ADOxx Modelling environment. As soon as the user creates or imports a model in ADOxx, the web application can retrieve it and through the “Model Export” user interface, the user can automatically generate a series of xWiki pages. An external reviewer can then look at the generated wiki pages and collaborate on the model, commenting in the relative wiki page. Through the “Import wiki

¹¹ https://en.wikipedia.org/wiki/Main_Page

¹² <https://www.xwiki.org/xwiki/bin/view/Main/WebHome>

comments” user interface the modeler can decide to automatically import existing comments in a specific attribute of the model or of the model objects.

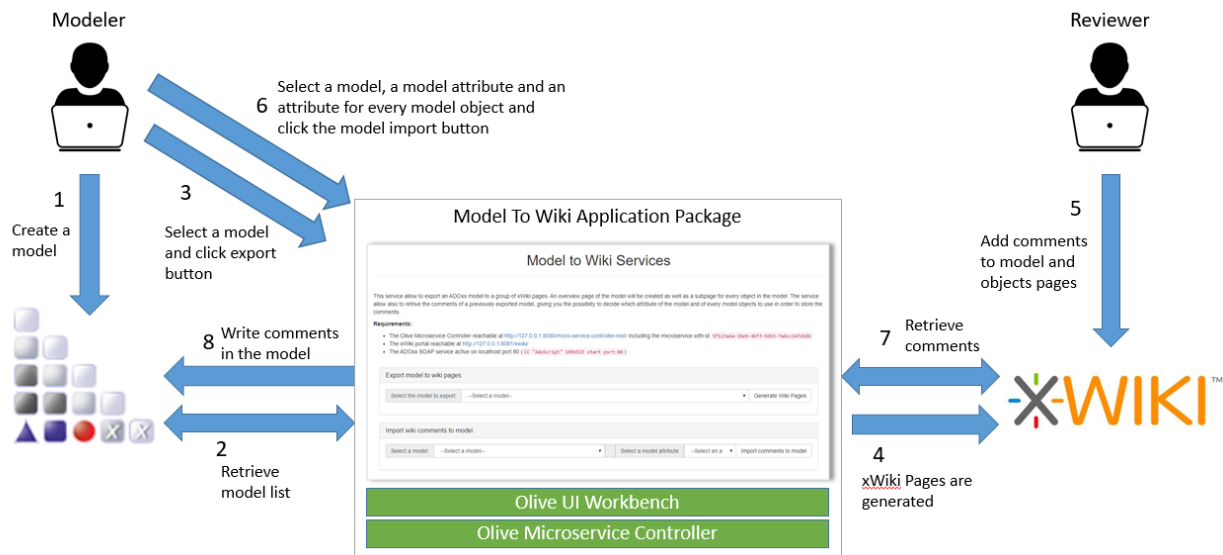


Figure 44 - Model Wiki Use Case Scenario

The web application uses the Olive microservices to communicate with the ADOxx modelling environment to (1) retrieve the list of all the available models, (2) retrieve all the attributes and objects of a specific model, (3) retrieve the image of a specific pages model, (4) retrieve all the attributes of a specific object and (5) write the comments in a specific attribute of a model or of an object. Additionally, Olive microservices are used for the communication with the xWiki platform to (1) create an xWiki page and (2) retrieve the comments of a specific xWiki page.

The user interface of the web application was developed using the Olive UI Workbench and is composed of one widget, named “Model-Wiki UI” displayed in the main rendering page.

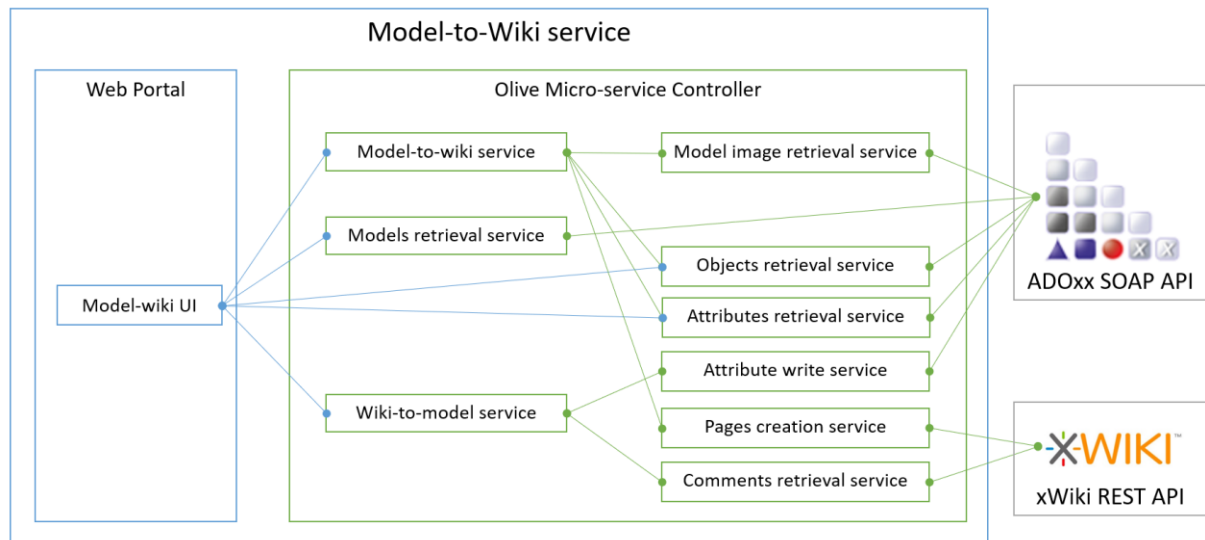


Figure 45 - Model Wiki Architecture

The xWiki REST APIs are used to perform all the required operations in the remote wiki platform.

The ADOxx Modelling environment is instead accessed programmatically using its SOAP APIs that need to be enabled using the AdoScript¹³ command `CC "AdoScript" SERVICE start port:80.`

The Model Wiki web application uses the following defined Olive microservices:

- **Model image retrieval service:** microservice that communicates with the SOAP¹⁴ interface of the ADOxx Modelling environment. It requires as input a model identifier (id) and returns the Base64 encoded image representation of the model.
- **Models retrieval service:** microservice that communicates with the SOAP interface of the ADOxx Modelling environment and returns a list of ids and names of all available models.
- **Objects retrieval service:** microservice that communicates with the SOAP interface of the ADOxx Modelling environment. It requires as input a model id and returns the list of all the objects inside the model.

¹³ <https://www.adoxx.org/live/external-coupling-overview>

¹⁴ <https://en.wikipedia.org/wiki/SOAP>

- **Attributes retrieval service:** microservice that communicates with the SOAP interface of the ADOxx Modelling environment. It requires as input the id of a model or of an object and returns the list of all its attributes.
- **Attribute write service:** microservice that communicates with the SOAP interface of the ADOxx Modelling environment. Requires as input the id of a model or object, the attribute name to use and the value to write in the attribute. Returns a confirmation code that reflects if the attribute has been written correctly or not.
- **Pages creation service:** microservice that communicates with the xWiki REST interface to create a page on xWiki. Requires as input a page id, page title and page content using the xWiki syntax.
- **Comments retrieval service:** microservice that communicates with the xWiki REST interface to retrieve comments on a specific xWiki page. Requires as input the page id of the xWiki page to lookup.
- **Model-to-Wiki service:** microservice that orchestrates all the Olive microservices required to generate xWiki pages from a model. The service will generate an xWiki page for the model with information on its graphical representation and its attributes and a sub-page for every model object containing a description of all the objects attributes. Requires as input the model ID and uses the “model image retrieval”, “object retrieval”, “attribute retrieval” and “page creation” microservices in the background.
- **Wiki-to-Model service:** microservice that orchestrates all the Olive microservices required to import comments of xWiki pages inside the respective model. Requires as input the model ID, a model attribute ID, and the list of model object IDs with a model object attribute. Uses internally the “comment retrieval services” to find all the comments in the model and uses the “attribute write services” to store the comments on the model.

About the frontend side the Model Wiki web application uses the following widgets defined using the Olive UI Workbench:

- **Model-Wiki UI:** This widget uses the “models retrieval service” to first obtain the list of all the available models in the ADOxx Modelling Environment and visualise them in a searchable selection box. As soon as a model is selected, with the “Generate xWiki Pages” button it is possible to generate the relative Wiki pages, using the underlying “Model-to-Wiki service”, while using the “Import xWiki Comments” it is possible to call the “Wiki-to-Model service” that is responsible to perform all the operations of importing the comments for all the xWiki pages back to the selected model.

Model to xWiki

Select the model: --Select a model--

Generate xWiki Pages

Import xWiki Comments

fac

Facade Renovation Process - Template(Business process diagram (BPMN 2.0))

Facade Renovation Process - Bilbao Instance(Business process diagram (BPMN 2.0))

4.2.2 Model Wiki Sample Use Case

This section contains an example of Model Wiki for the Facade Renovation process. This process is modelled in the ADOxx Modelling environment using the BPMN2.0 Library that allows to model processes using the BPMN2 standard notation.

As soon as the SOAP service is initiated in the ADOxx Modelling environment, the Model-to-Wiki UI is able to retrieve the list of all the available models. The user can now select the Facade Renovation process and click the “Generate Wiki Pages” button.

Model to xWiki

Select the model: Facade Renovation Process - Bilbao Instance(Business process diagram (BPMN 2.0))

Generate xWiki Pages

Import xWiki Comments

Figure 46 - Facade Renovation Process to Wiki

The generation of the xWiki pages may take time depending on the size of the model. As soon as the generation is completed the xWiki will contain a page describing the process model with its graphical representation and a description of all its attributes. Additionally, this page will contain a subpage for every task included in the model with its description.

The reviewer is now able to comment on the page relative to the model or on every subpage relative to the tasks. In this particular example a comment has been added on the Building Scaffold task.

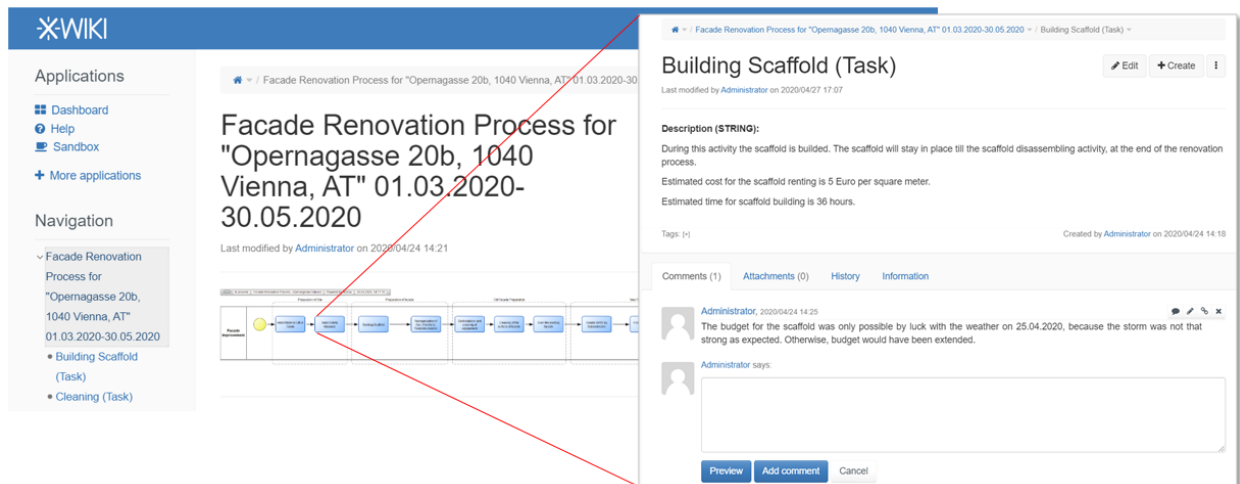


Figure 47 - Wiki Pages Generated for the Facade Renovation Process

When the modeler decides to check the status of the collaboration on the model, he can import the comments into the original model to be processed later. The modeler selects the Facade renovation process and by clicking the "Import comments to model" button the model will be updated, and the comments can be visualised in the attribute "Comment". In case of multiple comments all of them will be imported in the same attributes separated by a newline. Information about the user and the timestamp at the creation of the comments are reported as well.

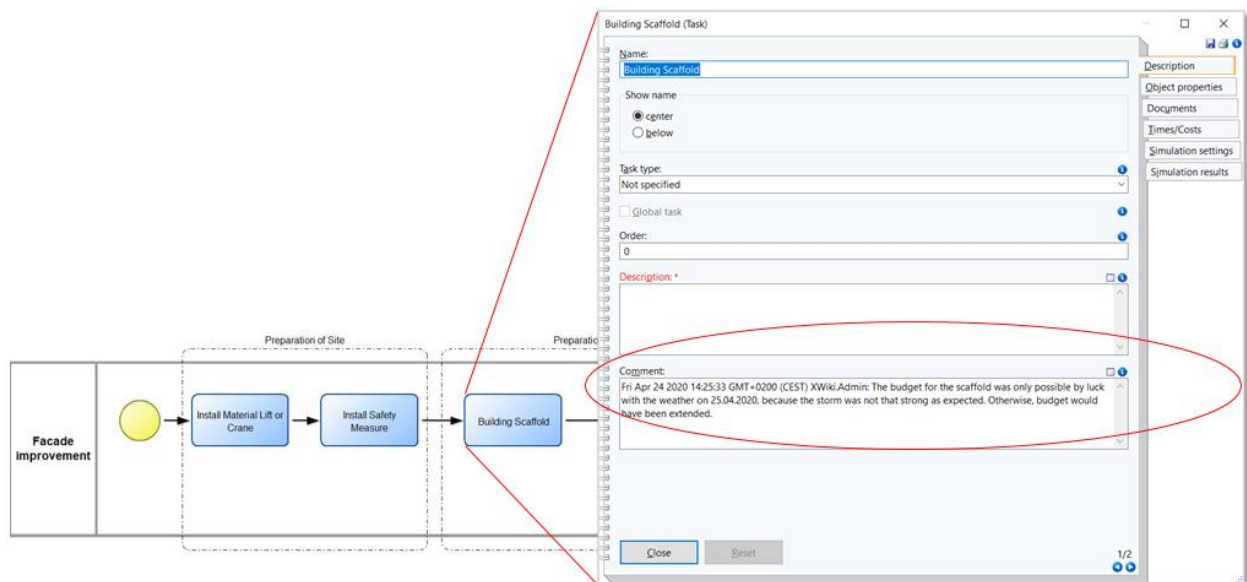


Figure 48 - Comments Imported for the Building Scaffold Task of the Facade Renovation Process

5. INTEGRATION WITH BIMERR TOOLS

This chapter contains a detailed view on the integration strategies used by the different components to interact with the BIF and between them.

5.1 INTEGRATION WITH BIF

As described in detail in the BIMERR Deliverables D4.5, D4.7 and D4.9, the BIMERR Interoperability Framework (BIF) essentially allows any application and tool developed in BIMERR to exchange building-related data, ranging from building data and occupancy data to renovation process data, in a meaningful and secure manner. In this context, the BIMERR Renovation Process Simulation Tool practically acts both as a building-related data provider and consumer to effectively enable the anticipated data exchanges with other BIMERR applications.

From the perspective of the BIMERR PWMA tool acting as a data provider, its respective developers access the integrated BIF platform interface and create as many data collection jobs as needed to upload the different renovation process data (depending on whether they intend to apply different access rules for different parts of the data). For each data collection job, they define the applicable ingestion method (that is typically a GET method exposed by the BIMERR PWMA tool) and configure all its related parameters (ranging from the authentication aspects and the query parameters to the ingestion schedule). Upon defining how the harvesting of the renovation process data will occur, they need to proceed with mapping and semantically lifting the data that are to be uploaded in BIF to the respective BIMERR data models (that are created based on the BIMERR ontologies described in section 5.2). They need to manually confirm whether the predicted mappings are correct and complement them with additional information related to the measurement units and the date-time formats, whenever applicable. Then, they need to define the metadata related to the specific data that are to be uploaded, e.g., the applicable building and project information, and the access policies that need to be applied (e.g., in terms of which applications should have or not have access to the specific data). Such a multi-step configuration at data collection time ensure that data will be

collected once or on a specific schedule from the APIs exposed by the BIMERR PWMA tool and shall be available for retrieval by other BIMERR applications. The data exposed, in particular, refer to the KPIs values simulated and calculated by the dashboard component of the BIMERR PWMA tool. As soon as the data are calculated it is exported as JSON and pushed to the configured BIF endpoint using a specific microservices created with the Olive framework.

From the perspective of the BIMERR PWMA tool acting as a data consumer, its respective developers access the integrated BIF platform interface and initiate a new query to identify the data they would like to access from other BIMERR applications with the help of the BIF. To this direction, they define the exact properties of the data they want to acquire and whether any of them should act as a query parameter according to their preferences (e.g., to retrieve very specific data for a specific id or all data for all ids in a single or multiple datasets). Once a query is created, the related datasets that include the requested information are identified and their access policies are enforced to check whether the BIMERR PWMA tool is authorized to access them. They are informed which are exactly the data that the BIMERR PWMA tool can access, they confirm whether the specific data are what they needed and they get a specific query identifier and the related information (such as an API key). Such information is utilized in the BIMERR PWMA tool to automatically retrieve the specific data that were selected in the query from the BIMERR APIs. In the BIMERR PWMA tool the data used refers to the building data provided by RenoDSS, like the square meters of a building as in the case of the renovation process of an external facade and are used by the KPIs Dashboard to calculate the values of different KPIs, like the renting cost of the building scaffold.

5.2 INTEGRATION WITH ONTOLOGY

The following sections describe the Key Performance Indicator and the Renovation Process ontologies, explaining the main concepts and properties used for their construction, and how they model performance indicators and processes utilized by the PWMA tools. It should be noted that the current model described in this document represents an evolution of the first conceptualization detailed in D4.2 (BIMERR

Consortium, 2020 D4.2) and D6.4 (BIMERR Consortium (2020). D6.4) and introduce the ontology that is described with more detail in D4.3 (BIMERR Consortium, 2020 D4.3).

By means of this semantic representation, PWMA can share their indicators and defined processes with other BIMERR applications interested in analyzing or displaying project management information.

5.2.1 KPI Ontology Description

This ontology described in BIMERR deliverable D4.3 aims to provide the vocabulary to represent indicators used to monitor the advancement of the project and verify if the goals or sub-goals established at the beginning of a task or process are being satisfied. For that purpose, the model should be able to represent not only conceptual information but also numerical information that will allow the project manager or any other stakeholder monitor the advancement of the renovation activities. This numerical information is the result of the assessment of several aspects of the renovation project, such as total time to finish the project or the planned cost related to the material to be used. The ontology also covers requirements coming from other BIMERR applications, such as RenoDSS, that also generates performance indicators.

One of the main entities of the ontology is the `kpi:Project` which is associated to one or more `kpi:Scenario`. The scenarios are linked to the concept `kpi:KPI` directly or through the concept `kpi:RenovationMeasure`, which is also linked to the `kpi:KPI`. During the development of the project the indicators defined at the beginning can be assessed to monitor the progress being made. The model enables to express this fact by establishing the relationship `s4city:quantifiesKPI` between a KPI Value and their corresponding definition (`kpi:KPI`). Also, if required, a time stamp can be added to the assessed KPI value, to indicate when this evaluation occurred (`saref4city:referestoTime`). The KPI metrics can be calculated based on a set of parameters (`kpi:CalculationParameter`), like the time or cost per unit. The current version allows the expression of a KPI assessments in terms of a range of values if necessary (`kpi:minValue` and `kpi:maxValue`), besides a tolerance property, an absolute and relative deviations are added to represent the permitted deviation from those limits.

The PWMA model also includes concepts for goals and sub-goals which main stakeholders of the project set at the beginning of the renovation process. In order to satisfy these requirements we introduce the class `kpi:Goal` and the property `kpi:hasSubGoal`. A KPI metric is not a fixed measure, it depends on a context that is defined by the `kpi:Project` conditions and the renovation `kpi:Scenario` finally chosen to be implemented.

To show how to annotate data using the KPI ontology we take the example provided in Section 6.2 of D6.3 (BIMERR Consortium, 2020 D6.3). This case describes the KPI model used to estimate the cost related to the “building scaffold” task. This metric needs as input several parameters: the m^2 of scaffold, the price per m^2 , and the number of days the scaffold is needed. These parameters are not obtained at once but created at different stages of the project. For example, the m^2 of scaffold and the prices per m^2 are estimated during the initial plan, meanwhile the usage time of the scaffold can only be known after signing the contract. In the use case, the target cost is estimated at 15 EUR per m^2 , and the total m^2 of scaffold needed is 1000 m^2 .

The Figure 49 depicts how this information can be projected into the KPI ontology. Blue boxes are the ontological concepts previously discussed, and the individuals are represented by white boxes under the concepts they belong to. The dashed lines are the relations or attributes described before connecting to individuals or literals, respectively. The cost measure, represented by the individual `data:ScaffoldCost`, is of type `kpi:KPI`, where their assessment is encoded by the `data:ScaffoldCostValue01` instance, which has a numerical value of “300000”, and a unit of measure of “EUR”. Additionally, we can link the KPI metrics to the parameters needed for its calculation. In this case, we have the parameter cost per unit, that is represented by the individual `data:ScaffoldCostPerUnit01`, encoding the value (15) and the unit of measure (EUR_M2). The `data:ScaffoldCostValue01` has been evaluated within the context of the project `data:BimerrProject01` and the renovation scenario `data:SimulatedScenario01`.

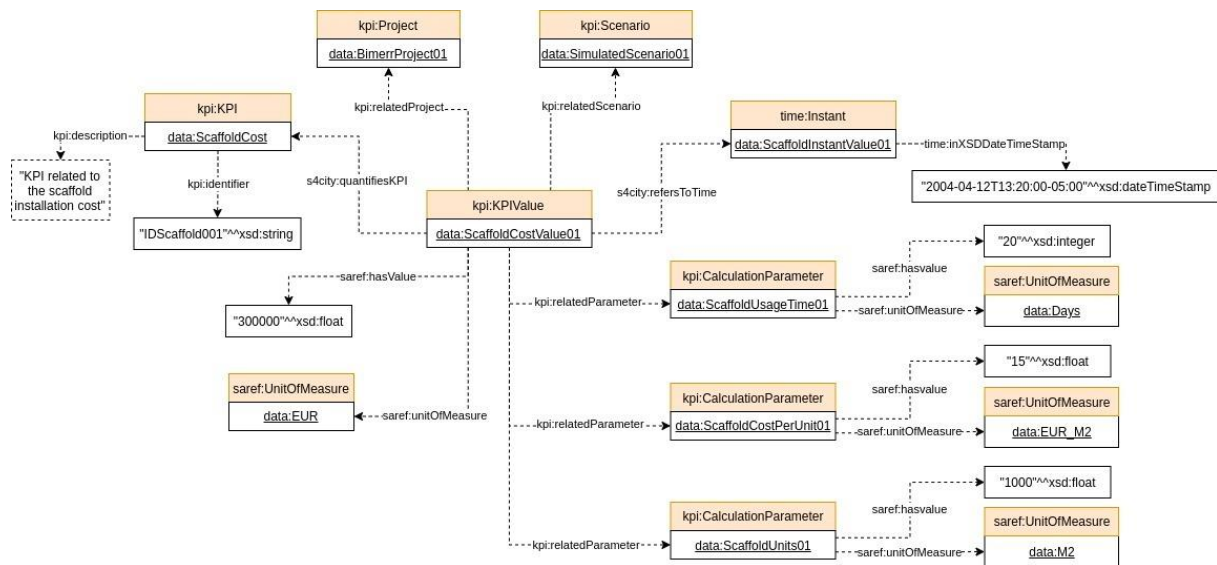


Figure 49 - Example of KPI Ontology Population

5.2.2 Renovation Process Ontology Description

This model represents the processes and tasks defined by PWMA tools to exchange information with ARIBFA application. The main entity of this ontology is `renp:Process`, which is linked to one or more `renp:Task` and `renp:Worker`. It is also possible to indicate to which `building:Element`, `building:Space` or `building:Storey` the process is related to. In addition, the processes and the task can provide `kpi:KPI` which are linked to their corresponding values (`kpi:KPIValue`). Finally, the process are defined by the attributes `renp:name`, `renp:identifier` and can be described by several attributes about the status and planned dates (`renp:executionStatus`, `renp:plannedStart`, `renp:plannedFinish`, `renp:actualStart`, `renp:actualFinish`).

Figure 50 shows an example of the renovation process ontology population. In this case we can see the process `data:Process001` about "Facade improvement process", which is associated to the project `data:Project01` and has two associated tasks `data:Task001` and `data:Task002`. The first task is to "Install material lift or crane" and it is executed by `data:Worker01` and its execution status is 100%. The next task is `data:Task002` which is performed over the building element `data:Wall1002` and its

execution status is 50%. Both tasks have also associated planned dates for start and finish.

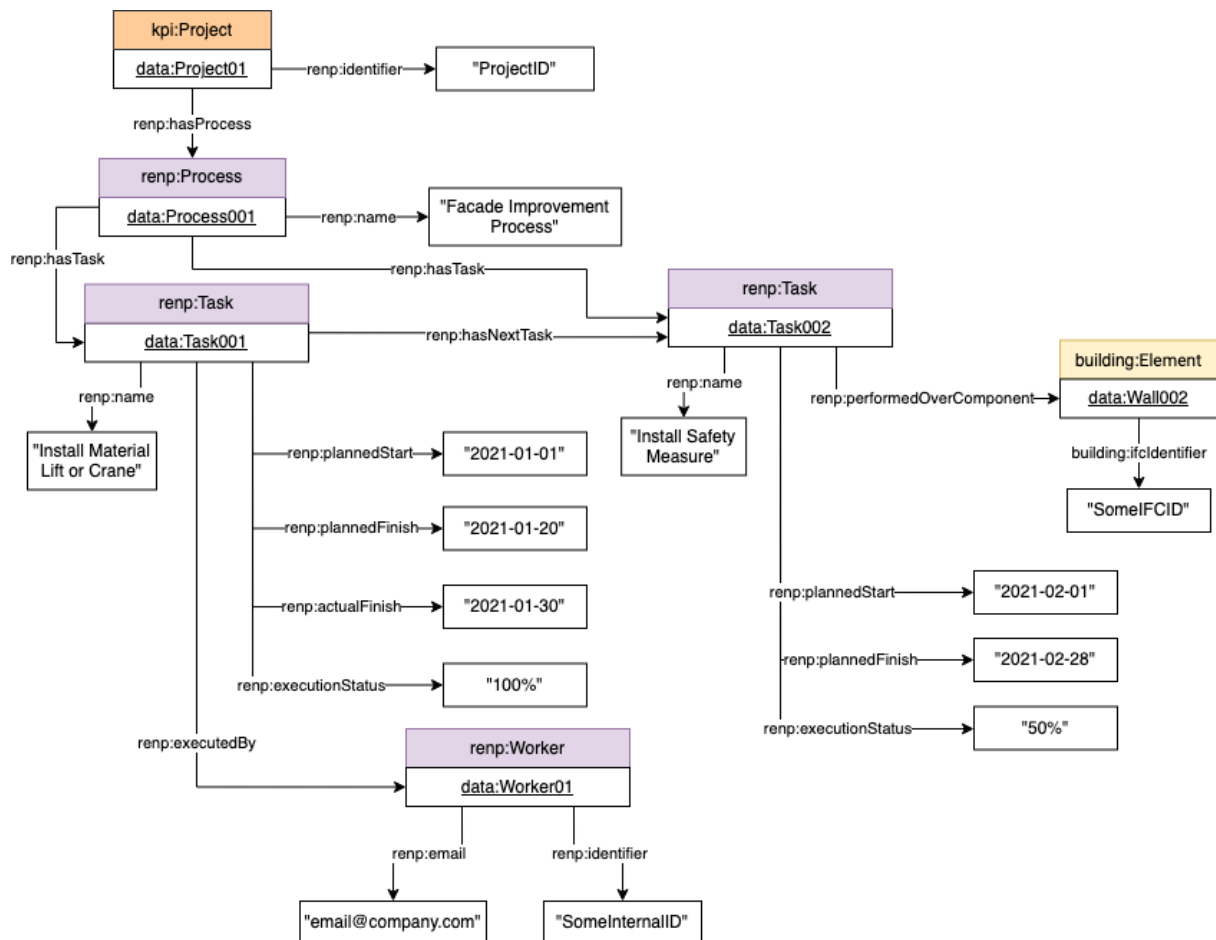


Figure 50 - Example of Renovation Process Ontology Population

5.3 INTEGRATION WITH THE ADAPTIVE WORKFLOW MANAGEMENT AND AUTOMATION TOOL

The modelling component will directly use the REST API provided by the Adaptive Workflow Management and Automation tool to upload a ready to execute workflow. Manual import is still possible using the BPMN DI export feature of the modelling environment and importing this from the workflow engine user interface. In case of automatic tasks modeled in the workflow, it is possible to define the task behaviour specifying parameters in a specific JSON structure that the workflow engine recognizes and uses to configure the service parameters (with values defined in the Master

Information Delivery Plan schedule), as shown in Figure 51, and perform the automatic action defined in the task, i.e., notify a resident about a health and safety related issue or send a notification in form of mail to instruct a building surveyor to start the BIM modelling process.

Service Parameters

```
{
  "requestedService": "internal notification",
  "data": {
    "receiver": "@mainWorker,@workorderManager",
    "message": "Workorder @workorderName has unexpectedly ended because alert at @currentTime"
  }
}

{
  "url": "https://adoxx.org/micro-service-controller-rest/rest/msc/callMicroserviceForced?microserviceld=76c014d7-328c-4907-a380-474ab5868746&operationId=sendMail",
  "method": "POST",
  "Content-Type": "application/json",
  "data": {
    "responsible": { "value":
      "_provide_here_the_surveyor_mail_on_first_row_" },
    "deliverable": { "value":
      "_provide_here_the_midp_deliverable_field_on_first_row_" }
  }
}
```

Workorder Service Tasks and Variables ? Type to search

Service Task action name: **Send Mail for MEP scanning exceeded time** Workorder Result ID: **2500**

Attribute: responsible Attribute pattern:

Attribute "responsible" value: Commit

Attribute: deliverable Attribute pattern:

Attribute "deliverable" value: Commit

Attribute: description Attribute pattern:

Attribute "description" value: Commit

Scottish Futures Trust - Master Information Delivery Plan (MIDP) Template

DELIVERABLE TITLE	DESCRIPTION	EXCHANGE FORMAT	PROJECT CODE	ORGANISATION CODE	SYSTEM CODE	LEVEL CODE	TYPE CODE	ROLE CODE	CLASSIFICATION/ANALYSIS CODE	NOTE CODE	CONTENT CODE	DATE REQUIRED TO BE SUBMITTED	STATUS
PROJECT CODE	ORGANISATION CODE	SYSTEM CODE	LEVEL CODE	TYPE CODE	ROLE CODE	CLASSIFICATION/ANALYSIS CODE	NOTE CODE	CONTENT CODE	DATE REQUIRED TO BE SUBMITTED	STATUS			

Figure 51 - Services Configuration UI in Workflow

Information on the running workflow is available through specific REST APIs provided by the workflow engine and returned in JSON format. The KPIs Dashboard of the BIMERR Renovation Process Simulation component can retrieve and visualise such information and use it for the KPI calculation process.

The workflow execution engine will receive from BIF information about the locations – sectors of the building. This information is going to be used to map the ongoing processes to the locations. This mapping is needed to generate notifications about the locations affected by the ongoing reconstruction process.

The workflow execution engine will provide information to other components via BIF. A JSON with the process log of the whole reconstruction process including all of its task's attributes (both planned and real recorded) will be made available via BIF.

The workflow execution engine will also provide notifications to other components. The already identified requirements to provide notifications are:

- List of planned affected locations.
- List of planned activities with time plan on selected location.
- List of finished tasks.
- List of rescheduled tasks.
- List of tasks with issues.
- Issue Reporting (two-way):
 - o From renovation manager as notification.
 - o From residents to Renovation Manager for review and acknowledgement.
- Health and Safety Notifications.

5.4 OPEN INTEGRATION FRAMEWORK

The Microservice Framework Olive¹⁵ is used as the basis for the development of all the services and functionalities of the BIMERR Design environment, related in particular to the integration with external components and BIF.

Olive is a platform that allows to create Web applications through configuration of existing components, both for the backend and for the frontend side. For the backend side such components are named Connectors, and their configuration results in ready to use REST microservices. For the frontend side such components are named Widgets and their configuration results in a web rendered ready to use user interface.

Both the Connectors and the Widgets are part of the Olive platform but can be extended in case of needs, using plug-ins. Connectors provide the functionalities of your backend services enabling the connection to external systems like databases and message buses. Olive allows to orchestrate such functionalities resulting in the definition of your business logic. Widgets on the other side are reusable components for the frontend and provide

¹⁵ <https://www.adoxx.org/live/olive>

the UI for sections of your web application. Widgets can be generics like visualising a grid layout or more specifics like visualising the simulation or the KPIs dashboard interfaces.

The strong point of Olive is its model awareness in the sense that such configurations are abstract enough that can be represented as models and the out-of-the-box integration with the ADOxx modelling environment allows to create the whole looks and behaviour of your web application, drawing models. This integration allows also to use models as data for microservices. An example is the process simulation microservice that simulates process models taking them directly from the ADOxx modelling platform, or the KPIs evaluator microservice that evaluates KPIs defined in models available in the ADOxx modelling platform.

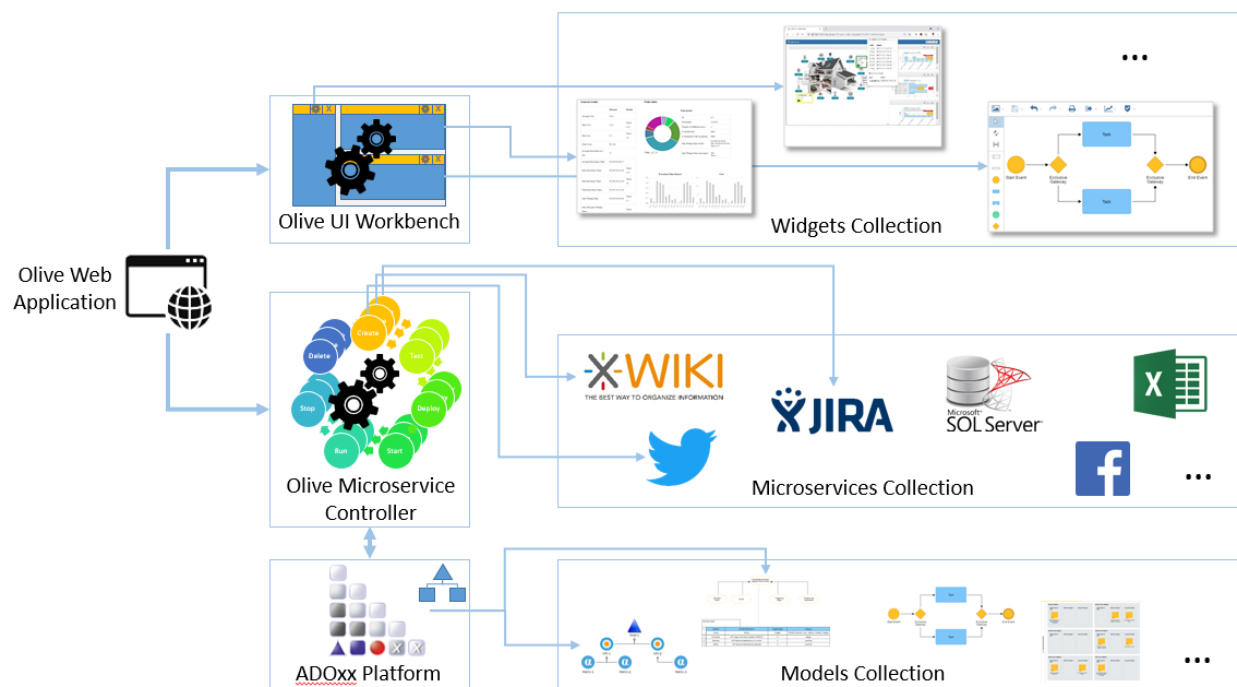


Figure 52 - Olive High Level Overview

The Olive platform provides a cloud environment where the user can define the microservices and the user interfaces of its web applications, expose it to the public and allow to control its lifecycle.

The Microservice Controller part of the Olive framework, is used in the PWMA environment of BIMERR as integration framework, allowing to create microservices that

collect and exchange data with the involved BIMERR components. The Olive Microservice Controller is a backend component that allows to define and manage Microservices in a novel way, following the configuration approach. A Microservice in Olive is defined only through the configuration of an existing platform component named Connector.

A Connector is a component developed in form of OSGi plug-in that allow to provide a specific functionality, like perform a query on a MySQL database or publish a post on Twitter. The name Connector derive from the fact that usually such functionalities depend on external systems (like the database) and the Connector is responsible to connect to such systems to exploit their features.

Olive Microservice Controller allows to manage the configurations of such Connectors, giving the possibility to create Microservices and control their whole lifecycle. Is responsibility of the lifecycle management component to (1) generates an instance of the REST microservice from the configuration, (2) allows to start the microservice, (3) keeps the microservice running in an isolated environment, (4) allows to stop the microservice and (5) allows to dismiss it.

The OSGi Connectors Loader component is responsible to load all the Connectors and make them available to the platform. It is built on the OSGi framework Apache Felix¹⁶ and will dynamically check the presence of the OSGi bundles (plug-ins) defining Connectors, loading, and unloading them on request.

As soon as the Microservices have been defined, they can be combined to achieve the business logic task, thanks to the Orchestrator component. This component is responsible to combine existing microservices using the Enterprise Integration Pattern¹⁷ notation. To support a higher level of freedom the orchestrator allows also to use the JavaScript scripting language to combine microservices following so a more programmatic approach.

¹⁶ <https://felix.apache.org/>

¹⁷ <https://www.enterpriseintegrationpatterns.com/>

The Olive Microservice Controller expose all this functionality both with Java and REST APIs. The former is used to integrate the Olive platform in local and desktop application. The latter is used to integrate the Olive platform with remote applications. Over the REST APIs, a management web user interface has been made available that allows to exploit all the features of the Olive Microservice Controller through the web browser.

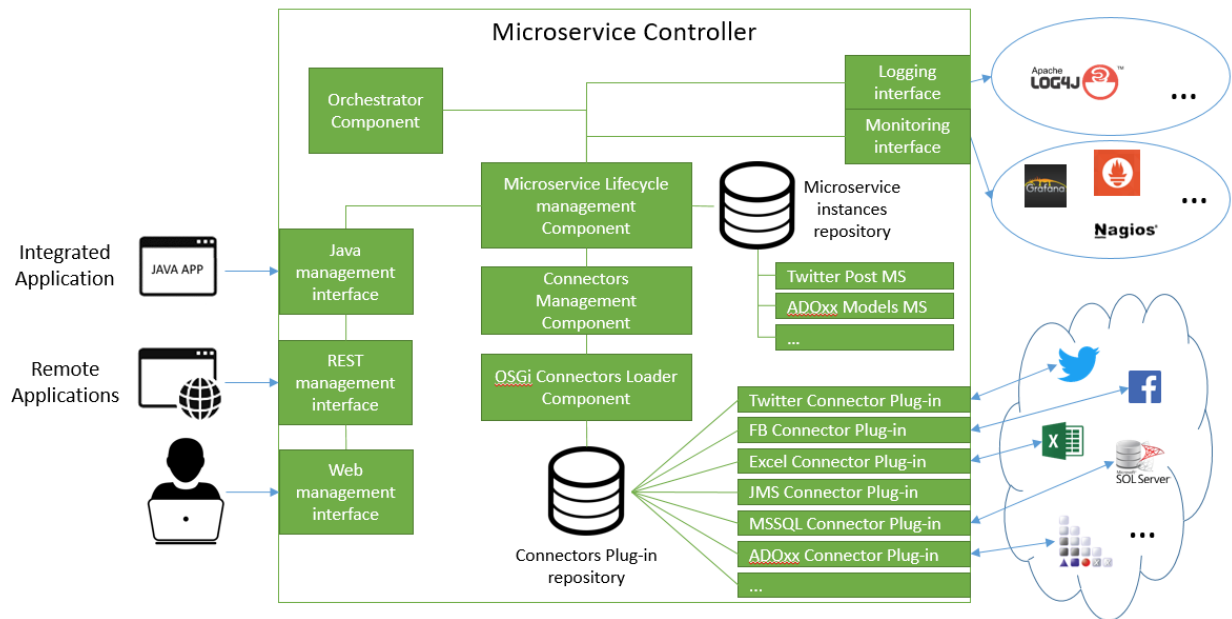


Figure 53 - Olive Microservice Controller Architecture

5.4.1 *Microservice Definition Model Type*

The Microservice Definition Model type is an ADOxx library that allows to model the exact behaviour of microservices and publish them in the Olive Microservice Framework. The definition of microservice using models is possible thanks to the nature of the Olive platform that defines microservices through configuration. In this way the models can be used not only to document but also to configure the microservice behaviour.

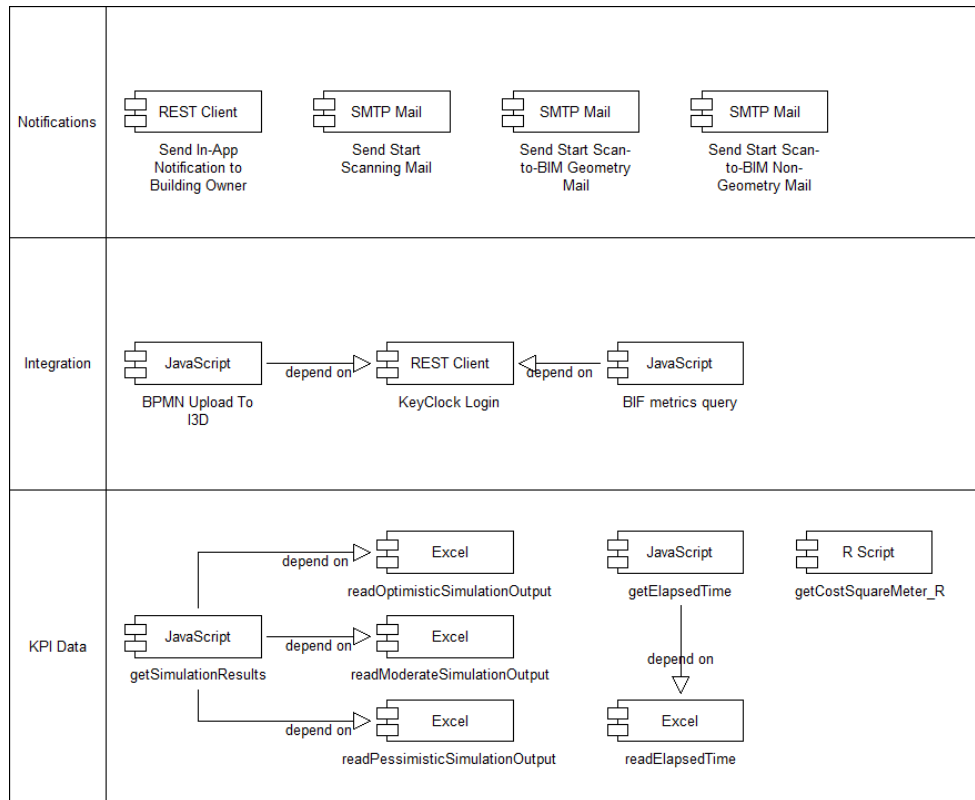


Figure 54 - Microservice Definition Model Sample

A Microservice Definition Model is composed of objects representing microservices of different types, based on the Olive Connector used, and relations representing dependencies between microservices. Each of the specific microservice type objects has a set of common and specific attributes accessible from its notebook. Common attributes are the description of the microservices and its auto-start value as well as all the attributes related to the input and output. The microservice inputs can be specified in a tabular form where each row is an input with information about its id, placeholder, description, and sample. The output can be instead adapted providing a specific JavaScript algorithm with the relative description of the new output.

KeyClock Login (REST)

ID	Placeholder	Description	Sample
1	username	\$username	username
2	password	\$password	password

Output Adaptation JS:

Output Description:

A JSON object in the following format:

```
{
  dataMIME: 'text/' / 'application/json' / 'all the other cases',
  dataText / dataJson / dataBase64: '_PlainText_' / '{_JsonObject_}' / '_ContentBase64_',
  moreInfo: {
```

Close Reset

Description
I/O
Health
Start
Call

Figure 55 - Microservice Definition Model, Input and Output Attributes

Health information are the last set of common attributes to all microservice objects and, like in the web interface, the user is allowed to specify the JavaScript algorithm that will be applied in order to validate the microservice output and check if the service is running properly or not.

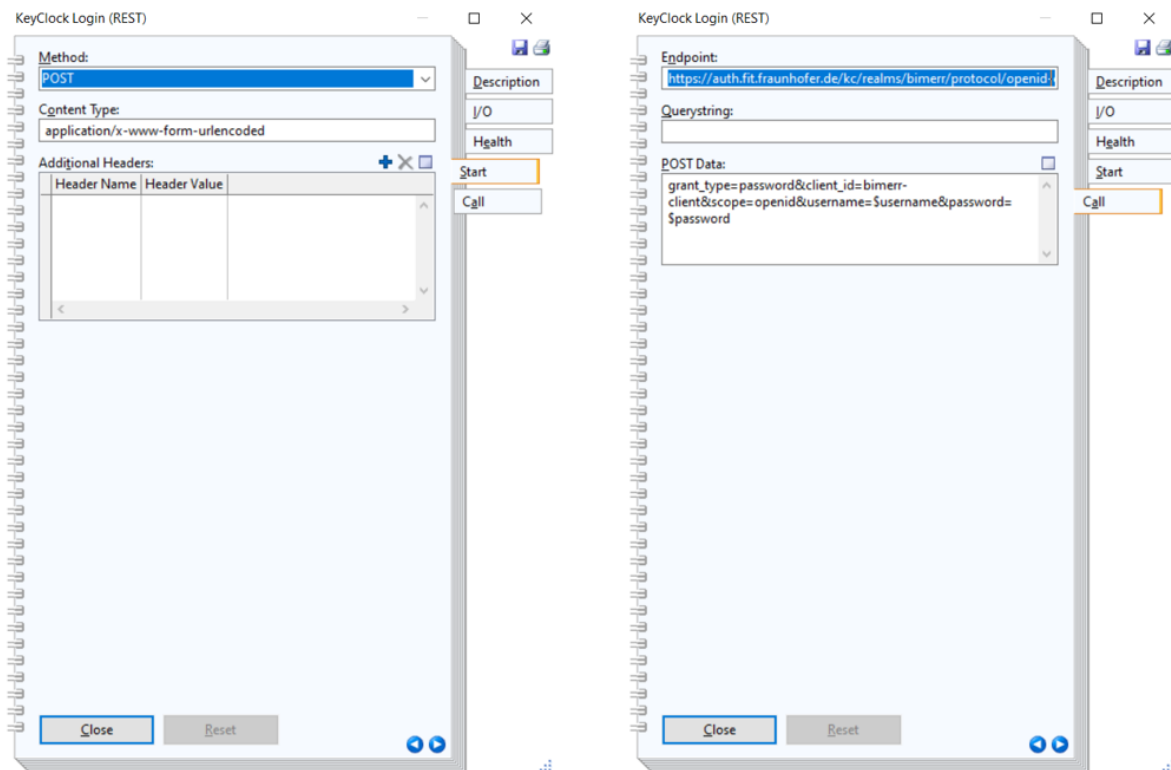


Figure 56 - Microservice Definition Model, Start and Call Attributes

The notebook sections named “Start” and “Call” contain instead the microservice specific attributes dependent on the Olive Connector used. In the example in Figure 56 the microservice is based on the Olive REST Connector and the “Start” related attributes allow to specify the HTTP method, Content type and headers of the REST request to perform, while in the “Call” section there are attributes related to the REST endpoint, query-string, and the optional data to post.

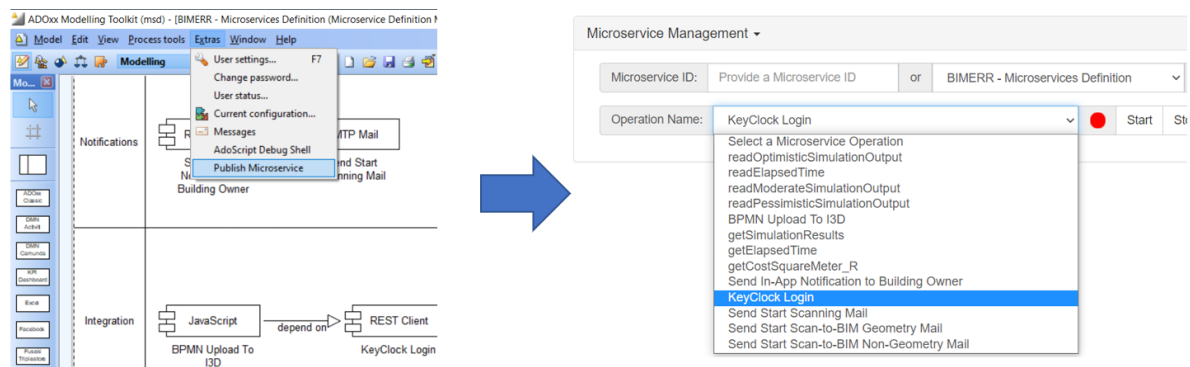


Figure 57 - Microservice Definition Model, Publishing feature

As soon as the microservice model is completed it can be published directly from the modelling environment thanks to the “Publish Microservice” function available in the “Extra” menu bar. The function will prompt for the endpoint of the Olive Microservice framework to use for the deployment, and it will automatically make it live.

6. CATALOGUE OF TOOLS FOR RENOVATION PROCESSES

This chapter describes where to download and how to setup and configure every tool presented in this deliverable.

6.1 DESIGN TOOLS

The design tool is provided in two versions: a community edition that everyone can download based on the desktop version of the ADOxx application, and a cloud version, deployed in the BOC cloud at https://bimerr.boc-group.eu/ADONISNP10_0/, that can be downloaded and installed locally only with a license. In the following the setup instruction of both cases are reported.

6.1.1 Community version of the Renovation process and KPIs design tool

The community version of the renovation process and KPI design tool require the installation of the ADOxx modelling platform and subsequently the installation of the library for modeling BPMN and the one created in BIMERR for the KPIs modeling. The following instructions will guide you through the installation of the community version of the renovation process design tool:

1. Download the ADOxx platform at <https://www.adoxx.org/live/download-guided>
2. Install it following the instructions provided in the page relative to your operating system.
3. Download the BPMN2.0 library from here: <https://git.boc-group.eu/bimerr/fast-deploy-package/-/blob/master/MODELS/BPMN/BPMN2Library.abl>
4. Download the KPI library from here: <https://git.boc-group.eu/bimerr/fast-deploy-package/-/blob/master/MODELS/KPI/KPIMMLibrary.abl>
5. Install the BPMN and the KPI library in the ADOxx platform following the instruction provide in this video: https://www.adoxx.org/live/import_new_application_library
6. Download the sample BPMN models from here: <https://git.boc-group.eu/bimerr/fast-deploy-package/->

[/blob/master/MODELS/BPMN/BIMERR%20-%20Facade%20Renovation%20Processes%20-%20BPMN%20Model.adl](#)

7. Download the sample KPI model from here: <https://git.boc-group.eu/bimerr/fast-deploy-package/-/blob/master/MODELS/KPI/BIMERR%20-%20Building%20Scaffold%20-%20KPI%20Model%20v2.adl>
8. Import the downloaded sample models following the instruction provided in this video: https://www.adoxx.org/live/import_models_adl

6.1.2 Cloud-Based Renovation Process Design Tool

The cloud version of the renovation process design environment is accessible using the BOC cloud deployed instance at https://bimerr.boc-group.eu/ADONISNP10_0/.

The integration with the BIMERR Identity provider Keycloak is currently ongoing so an internal login system is currently used. Credentials can be freely requested on faq@adoxx.org and as soon as the integration with the BIMERR Identity provider is completed the modelling environment could be accessed in Single Sign On with the same credentials of the other BIMERR tools.

6.1.3 Standalone package of the Cloud Renovation Process Design Tool

The standalone package contains an easy way to deploy the cloud modelling environment for both production and testing purposes. The repository <https://git.boc-group.eu/bimerr/adonis-fast-deployment-package> contains the deployment package to download. The repository is private and accessible only after the acquisition of a valid setup license.

The only requirement for this package is the presence of Microsoft SQLServer already installed. Instructions are available inside the package in order to correctly deploy it.

6.2 MONITORING, EVALUATION, REFLECTION, AND INNOVATION TOOLS

The monitoring and evaluation tools demonstrated in chapter 3 as well as the reflection and innovation tools demonstrated in chapter 4 can be downloaded in an all-in-one standalone package that simplify the deployment and testing of the demonstrated features. The standalone package is available in the ADOxx community portal for BIMERR project at <https://adoxx.org/live/web/bimerr/downloads>. Here the users can also find all the demonstrated tools in form of single packages ready to be tested locally. Additionally, demo videos and documentation are available on the same page for every package in order to document the deployment process and usage.

6.3 OPEN INTEGRATION FRAMEWORK OLIVE

This section will contain the instructions to build and setup the Olive Microservice Controller framework. This framework is a dependency for most of the tools presented in this deliverable and its presence is required to execute them.

The Olive framework is accessible from the main ADOxx page at <https://www.adoxx.org/> through the “GET ACCESS” green button visible in Figure 58. From this page is possible to download all the available Olive packages and access all the documentation materials needed to get started and work with the Olive framework.

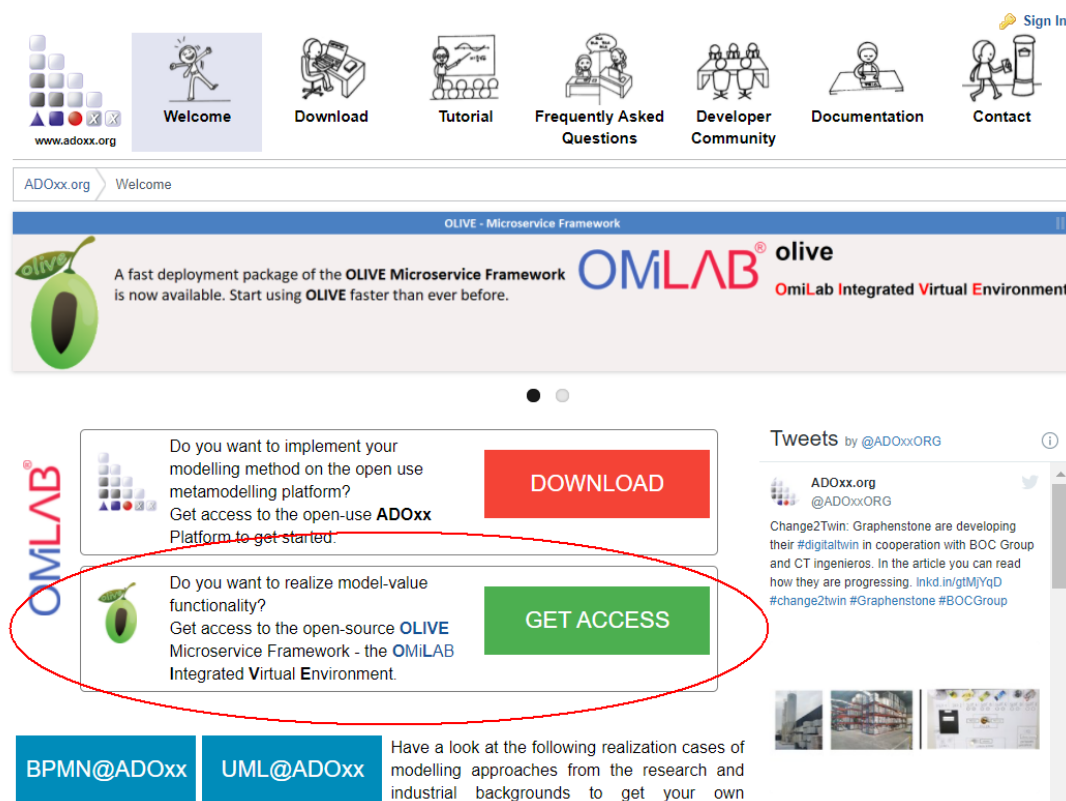


Figure 58 - ADOxx Olive Homepage

Four different deployment modalities have been provided for the Olive Framework:

- **Source code compilation:** to have the full control and perform changes in the code. Suggested for development.

- **Manual setup:** to have the full control of the deployment process. Suggested for production deployment in legacy systems that cannot use the Docker technology.
- **All in one standalone package:** useful for local testing of the platform. The full product is provided in a standalone package without external dependencies and that do not need installation. This modality will work only on window operating system.
- **Docker setup:** a production ready deployment based on Docker container. This is the best option to test on machines supporting the Docker technology and for production deployment.

All the packages (available at <https://www.adoxx.org/live/olive>) have a “getting started” and “usage manual” available in order to document their features and installation procedures.

7. CONCLUSION AND OUTLOOK

This deliverable demonstrates the functional capabilities of the PWMA ecosystem supporting the renovation process management and in particular the process simulation, formal verification, monitoring and evaluation as well as the integration with other BIMERR components. The technology that is described in this deliverable corresponds to the approach that is described in D6.3 “Adaptive Renovation Process & Workflow Models 2”.

This document explains the renovation process management tool ecosystem as follows:

- Firstly, using the meta-modelling platform ADOxx, and configuring it for renovation process management, KPI Models and Data Models in so-called ABL files. The platform can be downloaded for academic use at www.adoxx.org, whereas the used ABL files can be downloaded at www.adoxx.org under developer communities/developer spaces/BIMERR.
- Secondly, the Microservice Framework Olive is used to provide a set of functional capabilities for the models related to the integration with other BIMERR components. The framework is provided as a download package at ADOxx.org following the link to Olive download.
- Thirdly, the BIMERR specific set of microservices that provide the functionality described in this document, is also provided as a package. It can be downloaded on ADOxx.org, following the link to download Olive and selecting the latest BIMERR download package to get the latest version.
- Fourthly, in case of accessing and integrating third party applications like the process mining tool or the xWiki platform, the corresponding download links to the development communities are listed in the development space on ADOxx.org as the ABL files mentioned in topic one above.

BIBLIOGRAPHY

BIMERR Consortium (2020). D4.2 BIMERR Ontology & Data Model 1

BIMERR Consortium (2020). D4.3 BIMERR Ontology & Data Model 2

BIMERR Consortium (2020). D4.5 BIMERR Building Semantic Modelling tool 2

BIMERR Consortium (2020). D4.7 BIMERR Information Collection & Enrichment Tool 2

BIMERR Consortium (2020). D4.9 Integrated Interoperability Framework 2

BIMERR Consortium (2020). D6.3 – Adaptive Renovation Process & Workflow Models 2.

BIMERR Consortium (2020). D6.4 – Renovation Process Simulation Tool 1.

Kaplan, Robert S., and David Norton (1992). "The Balanced Scorecard: Measures that Drive Performance." Harvard Business Review 70, no. 1.

Tchana de Tchana Y., Ducellier G. and Sébastien R. (2019). Designing a unique Digital Twin for linear infrastructures lifecycle management. Procedia CIRP 2019 – 84 - 545-549. 10.1016/j.procir.2019.04.176.

Remco M. Dijkman, Marlon Dumas, and Chun Ouyang. Semantics and analysis of business process models in BPMN. Information and Software Technology, 50(12):1281–1294, November 2008.

LearnPAd Consortium (2015). D4.1 - Quality Assessment Strategies for BP Models.

ANNEX

BPMN Mapping to Petri Net

Initially based on the work from (Dijkman, et al., 2008) and resulting from the LearnPAD EU Project (LearnPAD 2015, D4.1), the BPMN to Petri-net mapping has been extended to support more BPMN elements and better align with its semantics. The proposed approach constructs the petri-net incrementally, choosing the Petri-Net elements required to represent each specific BPMN object, based on its behaviour specified by the OMG standard¹⁸ and then connecting the petri-net elements following the BPMN relations in the model.

Additionally, to externalize the mapping definition and make the simulation possible also using models different from BPMN, we defined a Domain Specific Language (DSL) able to represent a mapping rule in textual format, understandable by the importer module of the simulation component, that will use it during the model parsing in order to automatically generate a petri-net. In the following lines, the syntax of the created DSL for mapping rules is provided:

```
MappingFormula = MapRules ";" InputRelations ";" OutputRelations
```

```
MapRules = elementTypeList ":" RuleList
```

```
InputRelations = "in :" relationList
```

```
OutputRelations = "out :" relationList
```

```
elementTypeList = ElementTypeName [ "|" elementTypeList ]
```

```
ElementTypeName: the type name of the object you want to map
```

```
RuleList = Rule [ "," RuleList ]
```

```
Rule = from ">" to | from
```

¹⁸ <https://www.omg.org/spec/BPMN>

from: name of the PN element you want to create/use for connection with the "to" element; this name must start with "p" in order to refer to a place or with "t" in order to refer to a transition. In case of place is it possible to specify the number of tokens between parenthesis "(numtoken)".

to: name of the PN element you want to create/use for connection with the "from" element; this name must start with "p" in order to refer to a place or with "t" in order to refer to a transition. In case of place is it possible to specify the number of tokens between parenthesis "(numtoken)".

```
relationList = relation [ "," relationList]
```

```
relation = RelationType "=" PNElementName
```

RelationType: name of the relation type you want to map.

PNElementName: name of the PN element defined in the from/to you want to use for the relation.

Considering the defined mapping rules syntax, the following formulas have been created for each BPMN element. Only the mapping for the most common BPMN elements will be presented in this section:

- **Start Event:**

```
start: p(1)>t ; in: message=t; out:sequence=t
```

A start event is converted in a place containing one token followed by a transition. It has no incoming sequence flows but can have incoming message flows that will be attached to its transition. and can have only outgoing sequence flow relations connected to the transition as starting point. This will reflect the semantic of a start event that in case of multiple outgoing sequence flows, they are executed in parallel and in case of multiple incoming messages the start will not occur as soon as all the messages are arrived.

- **End Events:**

```
end|terminate|error: p0>t,t>p1 ; in: sequence=p0 ; out: message=t
```

The end event as well as the terminate and the error events are converted to three petri-net elements. A place can be followed by a transition followed by another place. The incoming sequence flows go to the first places while the outgoing message flows start from the transition. No outgoing sequence flows or incoming message flows are allowed in an end event.

- Task:

```
task: p>t; in:sequence=p, message=t; out:sequence=t, message=t, bound=p
```

A task is represented as a place followed by a transition. The same mapping applies to all type of tasks like Sub-Task, User-Task, Service-Task, Manual-Task, Business-Task, Receive-Task, Send-Task and Script-Task. All the incoming sequence flows are connected to the place while the outgoing sequence flows will be connected to the transition. This happens because a task in BPMN will start as soon as each input flow arrives and at its terminate it will activate all the outgoing flows in parallel. Incoming and outgoing messages are instead attached to the transition because each incoming message flow will block the execution of the task until the message arrives. Tasks can also have bounded events attached. In such cases the outgoing connection to the event will be connected to the task place.

- Looping Task:

```
taskLoop: p0>t0,t0>p1,p1>t2,p1>t1,t1>p0 ; in:sequence=p0, message=t0 ;  
out:sequence=t2, message=t0, bound=p0
```

The looping task follows the same concepts of the task but in this case as soon as it is completed, it can be executed again. So, it is mapped to a place followed by a transition followed by a second place followed by two transitions, one of which go back to the first place. Inputs and outputs sequence flows follows the same logic of the task while the messages are connected to the internal transition in order to wait for the message to arrive in order to continue the task execution.

- Exclusive Gateway:

```
xor : p ; in:sequence=p ; out:sequence=p
```

The Exclusive gateway (both Converging and Diverging) as well as the Event-Based gateways are mapped to a single petri-net place accepting all the incoming and outgoing sequence flows. In this way it is executed as soon as a token arrives and only one choice is performed at the end for the outgoing sequence flows.

- Parallel Gateway:

```
and : t ; in:sequence=t ; out:sequence=t
```

The Parallel gateway (both Converging and Diverging) are mapped to a single petri-net transition, accepting all the incoming and outgoing sequence flows. In this way it is executed only when all the incoming sequence flows have a token and after its execution will send a token to all the outgoing sequence flows.

The mapping process begins parsing first all the BPMN objects in the model and, applying the defined rules, the petri-net elements are created. Then all the BPMN relations are processed, and the created petri-net elements are connected looking at the input and output section of the rule for the specific element. This approach allows to create valid petri-net models that reflect the BPMN behaviour in a modular way and that can be adapted to map other executable models to petri-net.

Properties to Computation Tree Logic Mapping

In this section it is described how the properties that the verification component can analyze, are converted in CTL. The CTL syntax uses the following most common elements:

- A: means “All” and instructs to find all the paths starting from the current state.
- E: means “Exists” and instructs to find at least one path starting from the current state.
- G: means “Globally” and instructs to find all the subsequent paths.
- F: means “Finally” and instructs to find on at least one of the subsequent paths.
- \rightarrow : means “Implication” and is the classic “if ... then” logical connector.
- AND: is the logical AND operator.
- OR: is the logical OR operator.
- NOT: is the logical NOT operator.
- oo: is a numeric constant that means “infinite”.
- DEADLOCK: is a LOLA keyword indicating that no petri-net transitions are fireable.

Based on the supported syntax the following properties are defined:

Deadlock existence:

A deadlock exists when all the petri-net transitions are not fireable and the process is not yet terminated, meaning that there are still tokens in places that are not final. Based on this assumption the obtained formula is:

$\text{“EF (DEADLOCK AND (_place1_name_ > 0 OR _place2_name_ > 0 OR ...))”}$

Where $_place1_name_$ is the name of one non final place and the expression $_place1_name_ > 0$ that is repeated for every non final place, means that the numbers of tokens in that place is greater than zero.

This property will find the existence of at least one deadlock. To automatically check all the deadlocks, the verification engine will apply the same formula multiple times, excluding, after finding a deadlock, all the places involved in that deadlock from the next

formula. This iteration will continue as soon as no more deadlocks are detected or no more places to exclude are available.

Unboundedness existence:

The petri-net model is unbounded when at least one of its places is unbounded. Based on this assumption we can define the following formula:

"EF (_place1_name_ = oo)"

The formula checks if at some point the number of tokens in the place "_place1_name_" will grow to infinite and must be verified for all the places or at least for all the final places. In order to limit the state explosion problem, we decided to evaluate each formula independently and combine the results at the end checking if at least one failed, instead of creating a single formula with multiple OR conditions.

Reachability:

Checking if an activity is reachable means checking if the respective places in the petri-net model will contain a token at some point in time. As described in section 3.3.1 this type of check allows to perform four kind of analysis:

- Check if the activity is always reachable:
"AF (_place_name_ >0)"
- Check if the activity is sometimes reachable:
"EF (_place_name_ >0)"
- Check if the activity is always not reachable:
"AG NOT (_place_name_ >0)"
- Check if the activity is sometimes not reachable:
"EG NOT (_place_name_ >0)"

Where "_place_name_" is the name of the petri-net place associated to the specific activity.

Path Existence:

Checking a path existence between two activities means to check if the second activity will be executed at some point after the first one. As described in section 3.3.1 we have eight possible cases and assuming “_first_place_name_” and “_second_place_name_” to be the name of the petri-net places associated respectively to the first and second activities we can identify the following properties:

- Existence of at least one path from the start to the end activities:
 $\text{“AG ((_first_place_name_ > 0) } \rightarrow \text{EF (_second_place_name_ > 0))”}$
- Existence of at least one path that does not start with the first selected activity but does end with the second selected activity:
 $\text{“AG (NOT (_first_place_name_ > 0) } \rightarrow \text{EF (_second_place_name_ > 0))”}$
- Existence of at least one path that does start with the first selected activity but does not end with the second selected activity:
 $\text{“AG ((_first_place_name_ > 0) } \rightarrow \text{EG NOT (_second_place_name_ > 0))”}$
- Existence of at least one path that does not start with the first selected activity and does not end with the second selected activity:
 $\text{“AG (NOT (_first_place_name_ > 0) } \rightarrow \text{EG NOT (_second_place_name_ > 0))”}$
- Every path goes from the start to the end activities:
 $\text{“AG ((_first_place_name_ > 0) } \rightarrow \text{AF (_second_place_name_ > 0))”}$
- Every path does not start with the first selected activity but does end with the second selected activity:
 $\text{“AG (NOT (_first_place_name_ > 0) } \rightarrow \text{AF (_second_place_name_ > 0))”}$
- Every path starts with the first selected activity but does not end with the second selected activity:
 $\text{“AG ((_first_place_name_ > 0) } \rightarrow \text{AG NOT (_second_place_name_ > 0))”}$
- Every path does not start with the first selected activity and does not end with the second selected activity:

$\text{“AG (NOT (_first_place_name_ > 0) } \rightarrow \text{AG NOT (_second_place_name_ > 0))”}$