



Project Acronym: **BIMERR**
 Project Full Title: **BIM-based holistic tools for Energy-driven Renovation of existing Residences**
 Grant Agreement: **820621**
 Project Duration: **42 months**

DELIVERABLE D6.4 Renovation Process Simulation Tool 1

Deliverable Status: **Final**
 File Name: **BIMERR-D6.4-v1.1**
 Due Date: **20/06/2020 (M18)**
 Submission Date: **30/06/2020 (M18)**
13/10/2020 (revised)
 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	Ferrovia 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	Csaba	Kanóc	NT	Kanoc@novitechgroup.sk
2	Dominik	Demeter	NT	Demeter@novitechgroup.sk
3	Martin	Mojžiš	NT	Mojžiš@novitechgroup.sk
4	Tsakiris	Thanos	CERTH	atsakir@iti.gr

REVISION CONTROL

Version	Author	Date	Status
0.1	BOC	09/04/2020	ToC
0.2	BOC	04/06/2020	Merged UPM contribution
0.3	BOC	08/06/2020	Merged NT contribution
0.4	BOC	09/06/2020	Merged SUITE5 contribution
0.5	BOC	10/06/2020	Ready for Internal Review
1.0	BOC	22/06/2020	Ready for submission
1.1	BOC	13/10/2020	Integrating reworked UI for smart glasses (pp 56) from NT. Ready for re-submission.

TABLE OF CONTENTS

<i>List of Figures.....</i>	8
<i>List of Tables.....</i>	12
ACRONYMS.....	13
EXECUTIVE SUMMARY	14
1. Introduction.....	16
1.1 Objectives of the Deliverable	16
1.2 Introduction of Taxonomy and Methodology	17
1.2.1 Designing Tool for Renovation Processes	18
1.2.2 Monitoring and Evaluation Tool for Renovation Processes	19
1.2.3 Innovation and Reflection Tools for Renovation Processes	19
1.2.4 Development Methodology	20
1.3 Structure of the Deliverable	20
2. Design Tools for Renovation Process	22
2.1 Renovation Process and Workflow Design Tool	22
2.2 Renovation Process KPI Design Tool	27
3. Monitoring and Evaluation Tool for Renovation Processes.....	35
3.1 Renovation Processes-Oriented KPI Dashboards.....	35
3.1.1 Models-based Monitoring Dashboards demonstration	36
3.1.2 Models-based Monitoring Dashboards architecture	39
3.2 Simulation Tools for Renovation Processes.....	43

3.2.1	Simulation tool demonstration use case.....	43
3.2.2	Simulation tool architecture	48
3.3	Creation of Digital Twin with Workflow Execution	51
4.	<i>Reflection and Innovation Tools For Renovation Processes</i>	60
4.1	Process Mining of Renovation Process.....	60
4.1.1	Logs preparation for Celonis	60
4.1.2	Creation of Analysis Workspace	61
4.2	Collaborative Reflection of Renovation Process	64
4.2.1	Model Wiki application	64
4.2.2	Model Wiki sample use case	68
5.	<i>Integration with BIMERR Tools.....</i>	71
5.1	Integration with BIF.....	71
5.2	Integration with Ontology.....	72
5.2.1	Ontology Model Explanation.....	72
5.2.2	Model Instantiation.....	75
5.3	Integration with Workflow.....	76
5.4	Open Integration Framework.....	77
5.4.1	Microservice definition	80
5.4.2	Microservice instantiation.....	84
5.4.3	Microservice Controller User Interface	86
6.	<i>Catalogue of Tools for Renovation Processes</i>	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	Fast deployment of the cloud Renovation process design tool	92
6.2	Workflow Execution Tools	92
6.3	Monitoring and Evaluation Tools.....	93
6.3.1	Fast deployment package	93
6.4	Reflection and Innovation Tools.....	94
6.4.1	Process Mining with Celonis.....	94
6.4.2	Collaboration with Model Wiki	94
6.5	Open Integration Framework OLIVE	95
6.5.1	Source code compilation.....	96
6.5.2	Manual setup	97
6.5.3	Fast deployment package setup.....	97
6.5.4	Docker setup	98
7.	Conclusion and Outlook.....	99
	BIBLIOGRAPHY.....	101

LIST OF FIGURES

Figure 1 - Ecosystem overview.....	18
Figure 2 - Renovation process design tool community version.....	23
Figure 3 - Renovation Process Cloud Modelling Environment.....	24
Figure 4 - Renovation Process Cloud Modelling Environment Main Interface.....	25
Figure 5 - Renovation Process Cloud Modelling Environment Design Interface.....	26
Figure 6 - KPI model.....	28
Figure 7 - Goals attributes	28
Figure 8 - KPI Attributes.....	29
Figure 9 - Data calculation model.....	30
Figure 10 - Metric Attributes.....	31
Figure 11 - Data Items Attributes.....	33
Figure 12 - Renovation process KPIs design tool community version.....	34
Figure 13 - Renovation KPI cockpit use case	35
Figure 14 - Backward-looking Monitoring and Forward-Looking Simulation of KPI-Scaffold Costs.....	37
Figure 15 - Simulation Output for KPIs	38
Figure 16 - KPIs Dashboard architecture.....	39
Figure 17 - KPI dashboard chart widget.....	40
Figure 18 - KPI dashboard table widget.....	41

Figure 19 - KPI dashboard image widget.....	41
Figure 20 - KPI dashboard tree widget	42
Figure 21 - Renovation process Simulation inputs.....	43
Figure 22 - Renovation process simulation input C_START_EVENT	44
Figure 23- Renovation process simulation input C_TASK.....	45
Figure 24 - Renovation process simulation input calculation	46
Figure 25 - Renovation process simulation general results	46
Figure 26 - Renovation process simulation detailed results	48
Figure 27 - Renovation process simulation engine architecture.....	49
Figure 28 - Renovation workflow process imported to I3D	51
Figure 29 - Adjustable reconstruction process template	52
Figure 30 - Visualized running reconstruction process.....	53
Figure 31 - Interaction with the objects of the reconstruction process	53
Figure 32 - Evidence of planned and collected attributes.....	54
Figure 33 - Output of the execution engine – list of attributes and notifications connected to the tasks	54
Figure 34 - Export of all the executed instances connected to a reconstruction process template	55
Figure 35 - Sub processes of the task.....	56
Figure 36 – Smart glasses application for on-site support of workers running on Head Mounted Display	57

Figure 37 – Smart glasses application for on-site support of workers – log-in screen	57
Figure 38 - Application for on-site support of workers to execute assigned tasks – list of assigned tasks.....	58
Figure 39 - Details of the assigned task.....	58
Figure 40 - Celonis logs preparation.....	61
Figure 41 - Celonis Analysis Workspace for BIMERR.....	63
Figure 42 - Excel Output.....	63
Figure 43 - Model Wiki use case scenario.....	65
Figure 44 - Model Wiki architecture.....	66
Figure 45 - Model-to-Wiki UI Widget	67
Figure 46 - Wiki-to-Model UI Widget	68
Figure 47 - Facade Renovation Process to Wiki.....	68
Figure 48 - Wiki pages generated for the Facade Renovation Process.....	69
Figure 49 - Wiki-to-Model UI widget for the Facade Renovation process.....	70
Figure 50 - Comments imported for the Building Scaffold task of the Facade Renovation process.....	70
Figure 51 - KPI conceptualization.	75
Figure 52 - Example of KPI Ontology Population	76
Figure 53 - Olive high level overview.....	78
Figure 54 - Olive Microservice Controller Architecture.....	80
Figure 55 - Olive Microservice REST endpoint sample using SoapUI.....	85

Figure 56 - Olive Microservice Controller Management UI - Main view.....	86
Figure 57 - Olive Microservice Controller Management UI - Test view.....	87
Figure 58 - Olive Microservice Controller Management UI - Edit/New Microservice view collapsed.....	88
Figure 59 - Olive Microservice Controller Management UI - Edit/New Microservice view expanded	90
Figure 60 - Workflow execution registration form	93

LIST OF TABLES

Table 1 - Renovation process simulation general results details	47
Table 2 - Ontology Prefixes and Namespaces.....	73

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

EXECUTIVE SUMMARY

This document describes the first set of renovation process management tool, which we consider as an ecosystem of applications, Software as a Service Offerings, Microservices, as well as 3rd party applications.

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, KPI modelling notation and Data modelling notation by providing a full-fledged process model repository. The repository uses conceptual meta-models to define the modelling language, hence the modelling language can be adapted to the particular needs of BIMERR and aligned with the BIMERR ontology to use the same semantic. 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 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 of renovation processes, (b) the dashboard visualization 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 a workflow engine, (b) to import data from the BIMERR integration framework that are display the status of the renovation process, (c) to interact with Process Mining tool that analysis the process execution after the renovation process has been complete in order to create lessons learned for the next project.

The functional capabilities had been defined in the corresponding D6.2 “Adaptive Renovation Process & Workflow Models 1”. The deliverable at hand therefore explains the technical concepts, the tool functionality of the requested features and provide the different applications for download at www.adoxx.org.

In parallel to the iteration of D6.2 which improves the way how renovation process management is performed in BIMERR, the follow up proposal of this document D6.5 will correspondingly adapt the tool set to provide better support for the renovation process management as well as to better integrate 3rd party tools into the updated renovation process management ecosystem.

1. INTRODUCTION

1.1 OBJECTIVES OF THE DELIVERABLE

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

This deliverable corresponds with Deliverable D6.2 “Adaptive Renovation Process & Workflow Models 1” and provide the technological basis in order to perform renovation process management. This document therefore focuses on the tools, infrastructures and technical frameworks that are provided in order 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 case the process models are interpreted, and hence require to be stored with the corresponding semantic description – in our case we use conceptual meta-models – there is additional meta-data necessary, 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 queries, simulations or model transformation but on the other side require sophisticated repository technology.

Model-driven tools can follow one of two complementary approaches. First one is the standardized tool approach, aiming to implement a standard tool according 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 configured modelling approaches and personalized tool features. We provide the latter, as it enables us the flexibility of 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-modelling platform ADOxx, which is available as open use for academic use in the world-wide community ADOxx.org. Results of this EU project in form of the corresponding prototypes introduced in this deliverable are therefore available as open

use and partly open source in this community. The meta-modelling approach enables the configuration of the modelling language and therefore enable a personalised configuration of the platform modelling language. In our case we consider to adapt the modelling language BPMN to also include energy relevant concepts and apply a semantical enrichment technique to semantically uplift models and modelling languages to be align with the BIMERR ontology.

Beside the flexibility of the meta-model, we introduce the Microservice Framework Olive 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.

This deliverable introduces the first set of ADOxx meta-modelling configuration as well as the initial set of microservices to provide a first ecosystem for renovation process management.

1.2 INTRODUCTION OF TAXONOMY AND METHODOLOGY

The application ecosystem consists of the main application, in our setup this is the design component realised on the meta-modelling platform ADOxx providing all necessary modelling features. The Microservice Framework Olive spans 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 act as a proxy object and interact with a corresponding 3rd party tool.

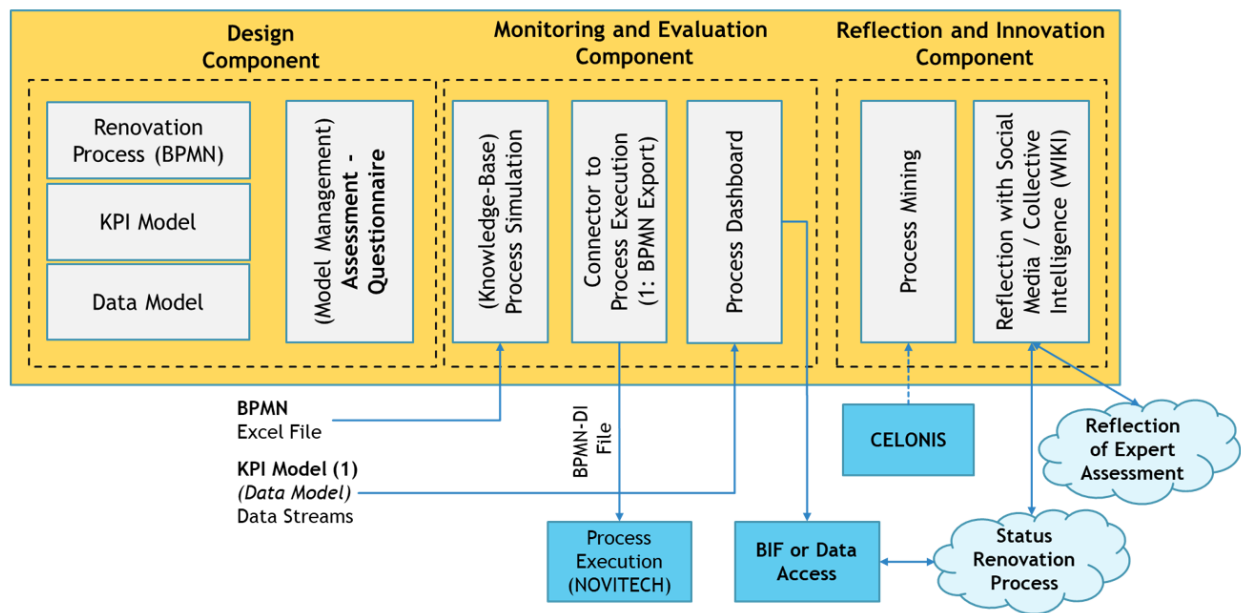


Figure 1 - Ecosystem overview

Figure 1 introduces the first ecosystem that is provided in form of this deliverable by providing a running demonstrator as well as the corresponding ADOxx configuration files, as well as Microservice Framework Olive with the realized Microservices for BIMERR as a downloadable package on ADOxx.org.

1.2.1 Designing 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 format. This transformation engine is important as it enable the graph-rewriting of a model and hence the transformation into another syntax to enable interaction with other tools.

The three boxes (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 which configuration file is in use.

Additional features – like a questionnaire or an assistant that supports the creation of a process model in particular when the engineer needs to instantiate the renovation project and create the project plan as well as the corresponding process model – are foreseen but currently not implemented.

Those additional features are introduced in the second iteration, when the users got familiar with the tool and hence can provide feedback on useful features when using it.

1.2.2 Monitoring and Evaluation Tool for Renovation Processes

This tool set is provided in form of microservices that interpret the KPI and Data Models and collect data from sensors. As we align the semantic description that is used in the BIMERR Integration Framework – BIF from WP4 – with our meta-model, the data that are received from the integration framework by the data sensors can be related to the models.

This knowledge-based enrichment enables the use of semantic information and domain specific concepts when reading the pure sensor data.

In order to test this, we introduce the renovation process context in both (a) a process execution that report each stage of the process as well as a forward-looking simulation that assesses how the process is likely to be in the near future. This forward-looking simulation is novel and the mechanisms in use are based on a discrete event simulation that simulates each 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. The next iteration of the tool set, will introduce advanced mechanisms to manage the knowledge without the exclusive usage of an Excel file.

The harmonised semantic from the models, the log-files and data streams from sensors – established because of aforementioned semantic alignment of the concept models in our design component and the ontology in the integration framework – allows to use a 3rd party process execution engine and hence demonstrate not only that renovation processes that can be assisted by the execution of a so-called digital representation – the process model – of the renovation process, but, due to the semantic alignment, the log files can be used within the BIMERR integration framework as well as in our monitoring environment.

1.2.3 Innovation and Reflection Tools for Renovation Processes

The final phase of the renovation process management is considered with learning and reflection. We consider reinforcement learning by using process mining – that is aligned with the semantic of the model as the model has been created in our design tool and uses the aligned semantic – to check if the process execution run as planned. In particular 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 form of updated models and simulation setting for the next project.

As the renovation process is a highly manual task in its execution and the execution engine can only represent a part in form of a so-called digital twin, we propose also the complementary use of a collaboration platform and provide a Wiki platform. 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 or 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 the decision afterwards.

In the second iteration of the prototype, we expect that the co-creative features will be strengthened by integration with legacy systems or the complementation of other tools and services.

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. First, the design-thinking approach is a top-down approach where ideas on the functional capabilities of the 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 D6.1 and the functional capabilities has been elaborated with the end users in D6.2. Second, 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 iteration, 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 will focus on the integration of other BIMERR and 3rd party services and adapt the prototype while being used by the end users. In overall we found the combination of an 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 STRUCTURE OF THE DELIVERABLE

The deliverable addresses the aforementioned objectives in form of:

- **Chapter 1** introduces the demonstration with an overview of the used tools and functionalities
- **Chapter 2** provide details of the Renovation process design environment. In this chapter the modelling environment for renovation processes and workflow and the environment for modelling KPIs associated to the renovation process are described.
- **Chapter 3** describe the monitoring and evaluation tools used in the demonstration. The first part of the chapter will provide details on the KPIs dashboard used to monitor the status of

some KPIs specifics for the considered use case while the second part will focus on the prediction of future behaviour through the renovation process simulation toolkit.

- **Chapter 4** introduce the innovation tools for the renovation process. The first section of the chapter introduce the analysis of data produced by the renovation process workflow engine using the Celonis toolkit, while in the second section the renovation process collaboration tool based on wiki concepts is presented.
- **Chapter 5** provide details about the integration with the other BIMERR tools, in particular with the BIF, the workflow system and data in the external world.
- **Chapter 6** describe details about the installation and instructions to use all the previously described tools.
- **Chapter 7** contain the conclusions and the outlook remarks.

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 ADOxx¹ meta-modelling platform that are used to create the renovation process and workflow design tools as well as the KPIs design tool.

In the following sections first the demonstration of the design environment for the renovation process and the renovation workflow are reported and then the design environment for the Renovation KPIs and Goals.

2.1 RENOVATION PROCESS AND WORKFLOW DESIGN TOOL

The business process design tool is an application build with ADOxx. ADOxx is a meta-modelling platform that allow to define your own meta-model and automatically generate the modelling environment for you accordingly it. In the business process design tool the meta-model is based on the BPMN2.0 standard meta-model. 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 D6.2 (BIMERR Consortium,2020) the concept of Template model has been introduced and all the created models are described in details. This is a renovation process abstract enough to be valid for all the renovation use cases, including all the practices to be taken into account. This template process has been designed using the BPMN2.0 meta-model. At a more detailed level the Renovation process instance represent the instanced templated model relative to the specific use case. Decisions specific for the use case here 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. Here constraints have been applied in order to have a model compliant with the BIMERR workflow engine such as necessity of converging gateways after every choice. Also 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 majorities of workflow execution engines on the market.

¹ www.adoxx.org

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.

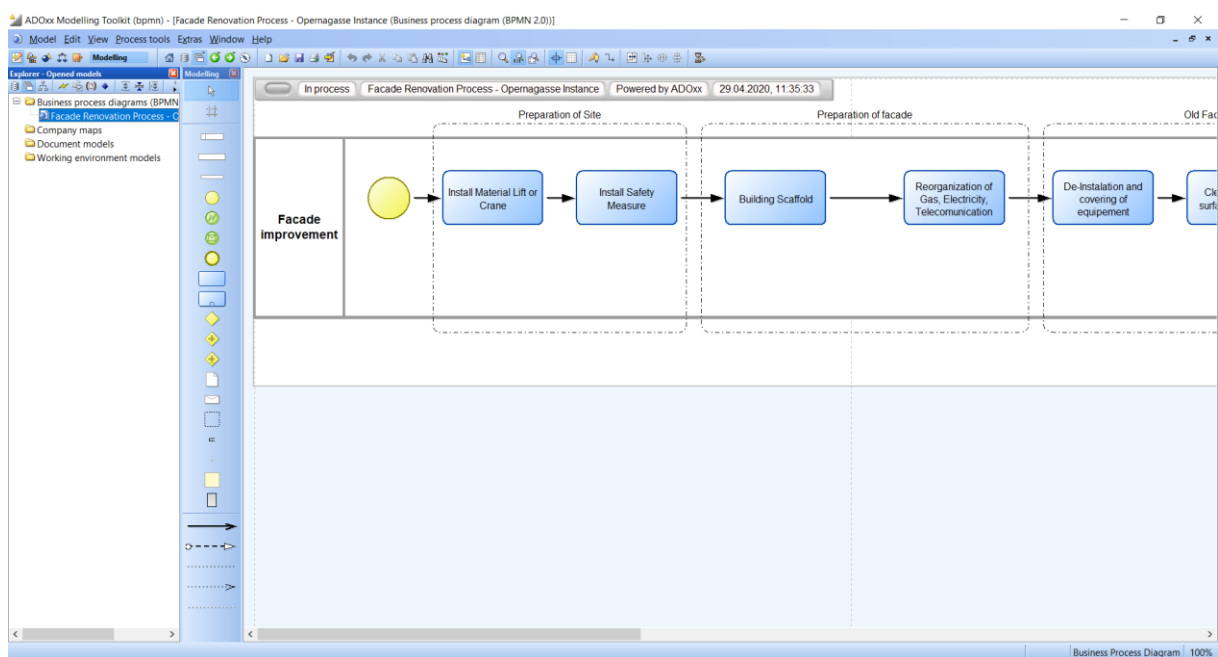


Figure 2 - Renovation process design tool community version

The cloud version of the Renovation Process Design tool is instead a BIMERR customized version of the cloud based ADOxx and is available under the endpoint https://bimerr.boc-group.eu/ADONISNP10_0/auth.view. The access is protected with a user and password authentication system that allow the profiling of the users and the compliance with security requirements. Credentials for testing the platform are user specific and can be requested via mail at faq@adoxx.org.

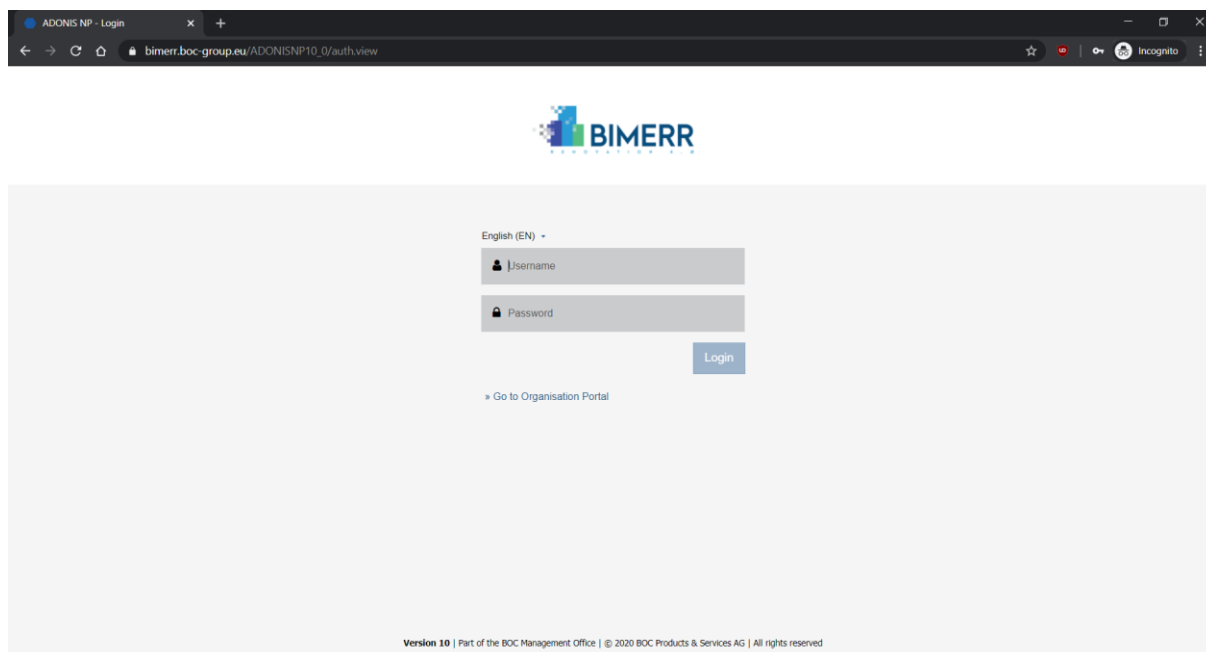


Figure 3 - Renovation Process Cloud Modelling Environment

The cloud version of the Renovation Process Design tool 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 4 - Renovation Process Cloud Modelling Environment Main Interface

In this section it is possible to explore all the existing models in a folder tree view, visualize them and create new ones using the BPMN2.0 modelling canvas. 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 in order to enable integration with other components of the BIMERR platform, in particular with the Workflow engine component.

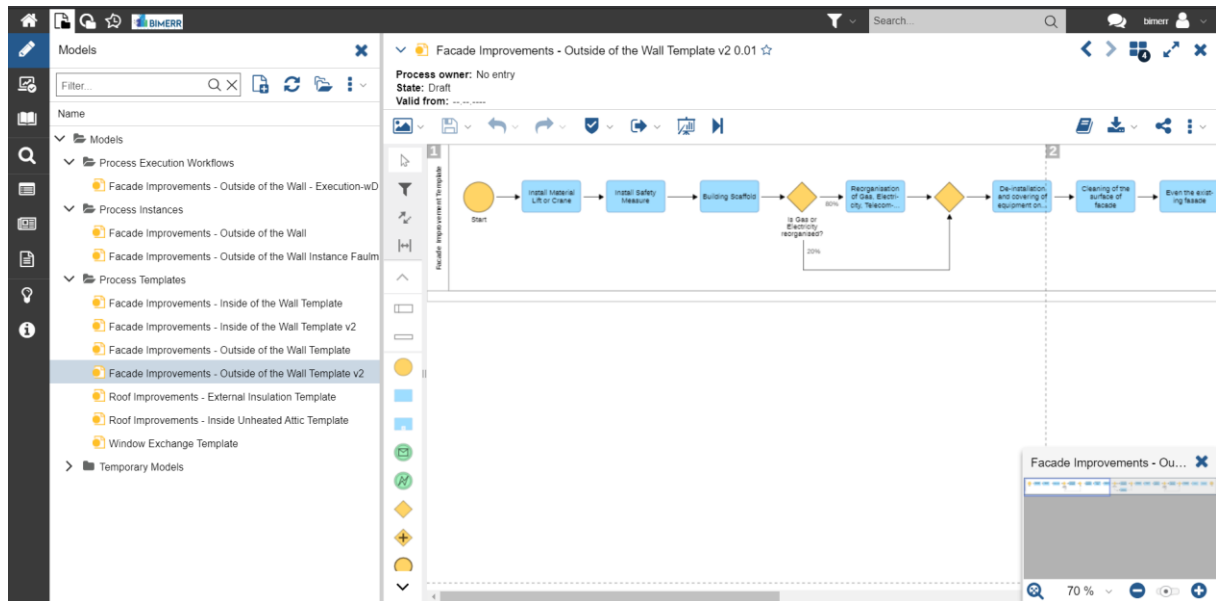


Figure 5 - Renovation Process Cloud Modelling Environment Design Interface

2.2 RENOVATION PROCESS KPI DESIGN TOOL

The Renovation process KPI design tool is an application build with ADOxx, a meta-modelling platform that allows to define your own meta-model and automatically generate the modelling environment for you accordingly it. Also in this case, like the renovation process design tool, a community version of the platform is available. The corresponding cloud version will be demonstrated in the final prototype. The community version library allows to create an ADOxx modelling environment as a windows desktop application that everyone can freely setup and redistribute.

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 D6.2 (BIMERR Consortium,2020) has been introduced the concept of KPIs for the scaffold cost of the renovation facade scenario. The first meta-model defined is the cause and effect model-type. It allows to define KPIs and Goals with their relations and group them in specific perspective. In particular:

- Perspective group similar KPIs, like grouping all “Financial” indicators or all “Time” or “Quality” dependent indicators.
- 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.

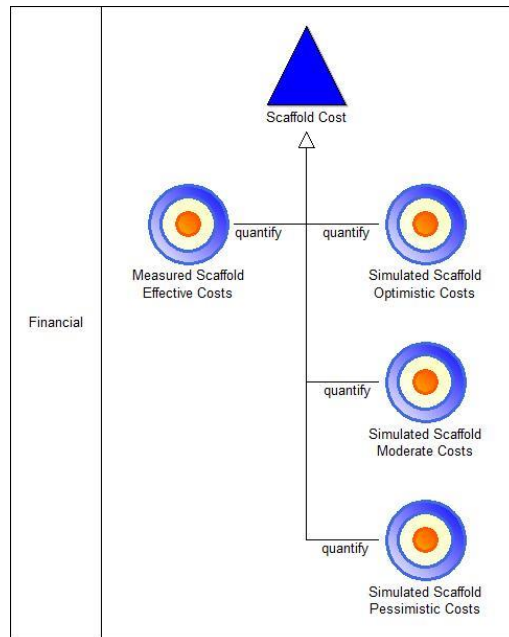
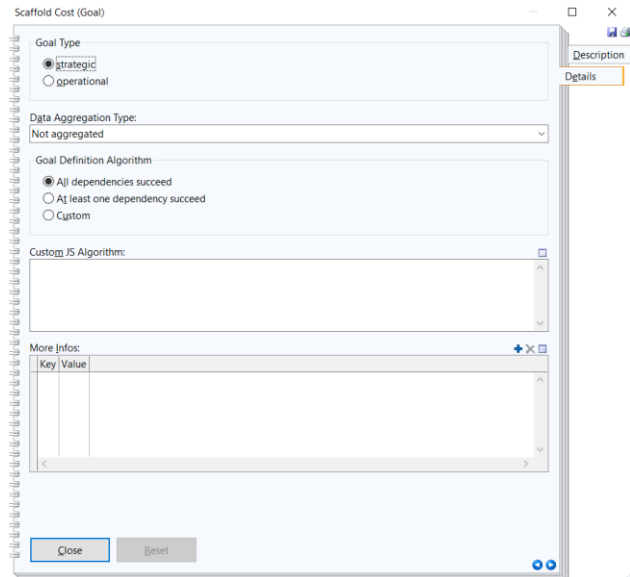


Figure 6 - KPI model

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 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 on details of the evaluation of the Goal is possible to specify the procedure used during its evaluation. The goal, therefore, can succeed if all its dependencies succeed or if at least one succeeds. 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 are relevant.



The screenshot shows the 'Scaffold Cost (Goal)' configuration window. It includes the following sections:

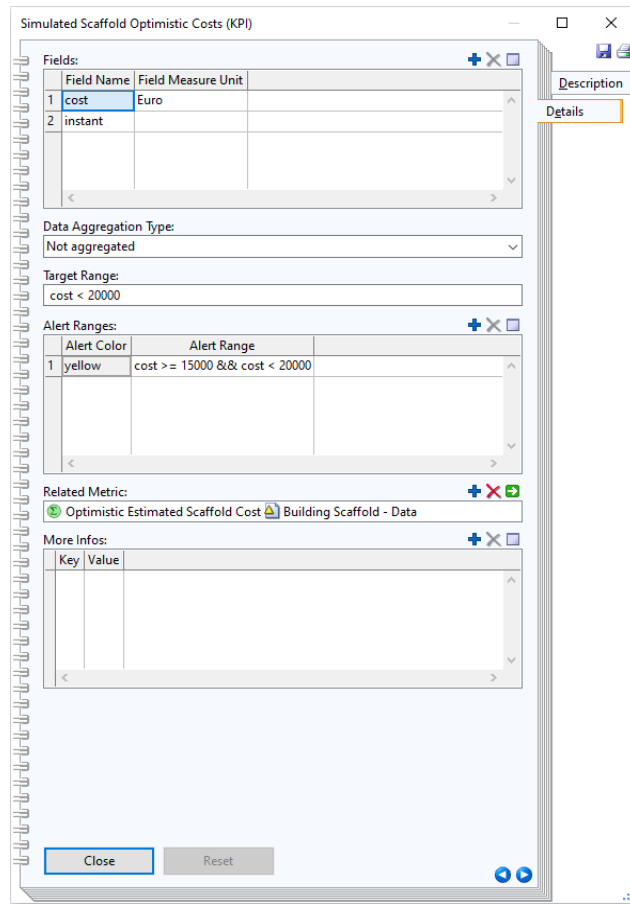
- Goal Type:** Radio buttons for 'Strategic' (selected) and 'Operational'.
- Data Aggregation Type:** A dropdown menu set to 'Not aggregated'.
- Goal Definition Algorithm:** Radio buttons for 'All dependencies succeed' (selected), 'At least one dependency succeed', and 'Custom'.
- Custom JS Algorithm:**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

Figure 7 - Goals attributes

In the case of KPI objects the specific attributes are relative to the fields of data available in the KPI and information on the target and alert ranges of the KPI value.

The Fields represent what kind of metrics are available in this KPIs. Usually there is a value field (cost in this case) that contain 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 fixed and always valid, so the user can provide as much as he need. Every field can also have a specific measure unit.

Also in this case it is possible to specify the aggregation type of the data representing how often the KPI is calculated. KPIs in order to be meaningful must be associated with thresholding that in this case are 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 (eg. Cost < 20000).



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 8 - KPI Attributes

Using the same approach, it is possible to define one or more alert range 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, so as example if the cost > 15000 there is a yellow/moderate alert while if the cost > 19000 there is 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 (eg. 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.

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

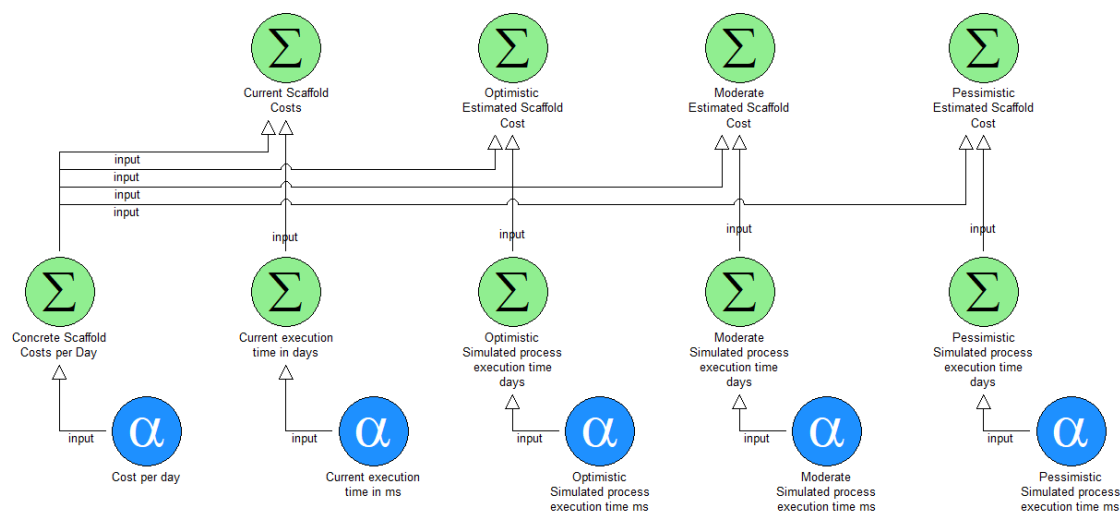


Figure 9 - Data calculation model

The data calculation model is composed of Metrics and Data Items including dependencies between them. Metrics (represented as green circles) represent a data in a specific format and contain information on how the value of this data have to be calculated using as inputs sub-metrics and data access indicators. The Data Items (represented blue circles) on the other side are able to describe how a data value is retrieved from an external system that can be as example a remote service, a sensor, 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 such external system.

Also 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 on the basis of 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 and start with a number.

The Fields represent what kind of data are available in this metric. Usually there is a value field (cost in this case) that contain the value of the metric and an instant time field that contain at what time the metric value has been calculated, but this are not fixed and always valid, so the user can provide as much as he needs. Every field can optionally have a specific measure 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 in the form of 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, so if we defined an alias for a sub-metric and such sub-metric have a field "cost" defined, in the formula this can be reached writing the alias name followed by a dot followed by the field name (eg. "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 (eg. 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 such behaviour creating first the aliases for the sub-metrics "Concrete Scaffold Costs per Day" and "Current execution time in days" naming respectively "a" and "b". Then the "cost" field of can be calculated using the function "a.cost * b.executionTime", where "a.cost" refer to the "cost" field of the "Concrete Scaffold Cost per Day" sub-metrics, while "b.executionTime" refer to the "executionTime" field of the "Current execution time in days" sub-metrics. 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".

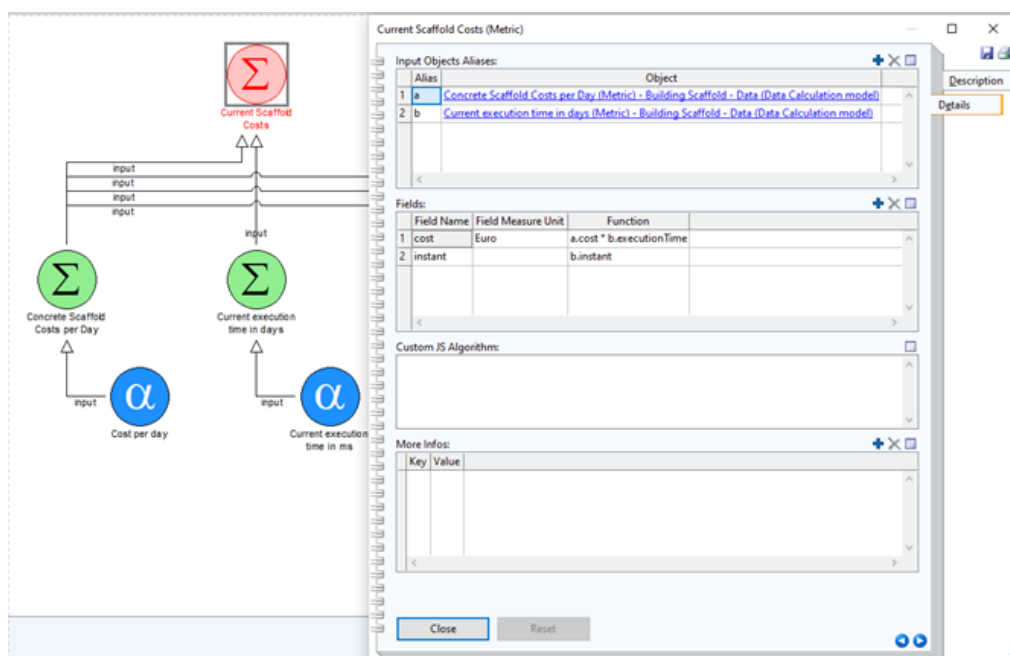


Figure 10 - Metric Attributes

The Data Items are able to represent how the data values are collected from external systems through the microservice framework Olive². For this purpose the microservice must return the data in a specific format in order to be recognized. The data must be a JSON object that contain a "columns" array containing 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"  
  
  }, {  
  
    "cost" : "1800",  
  
    "instant": "2020-01-28T09:40:15"  
  
  }]  
}
```

² <https://www.adoxx.org/live/olive>

Also in this case is important to define the data fields that the service return 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 appropriate value. This will be the same input that the microservice expect 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.

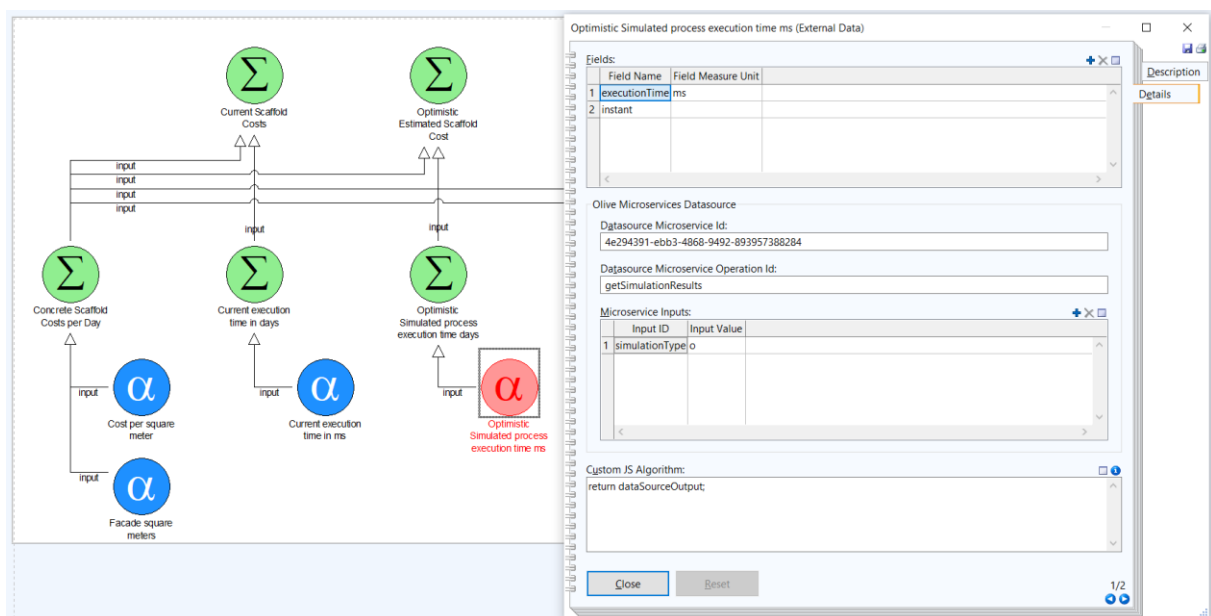


Figure 11 - Data Items Attributes

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 allow 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-on that the user can install from the ADOxx community portal www.adoxx.org.

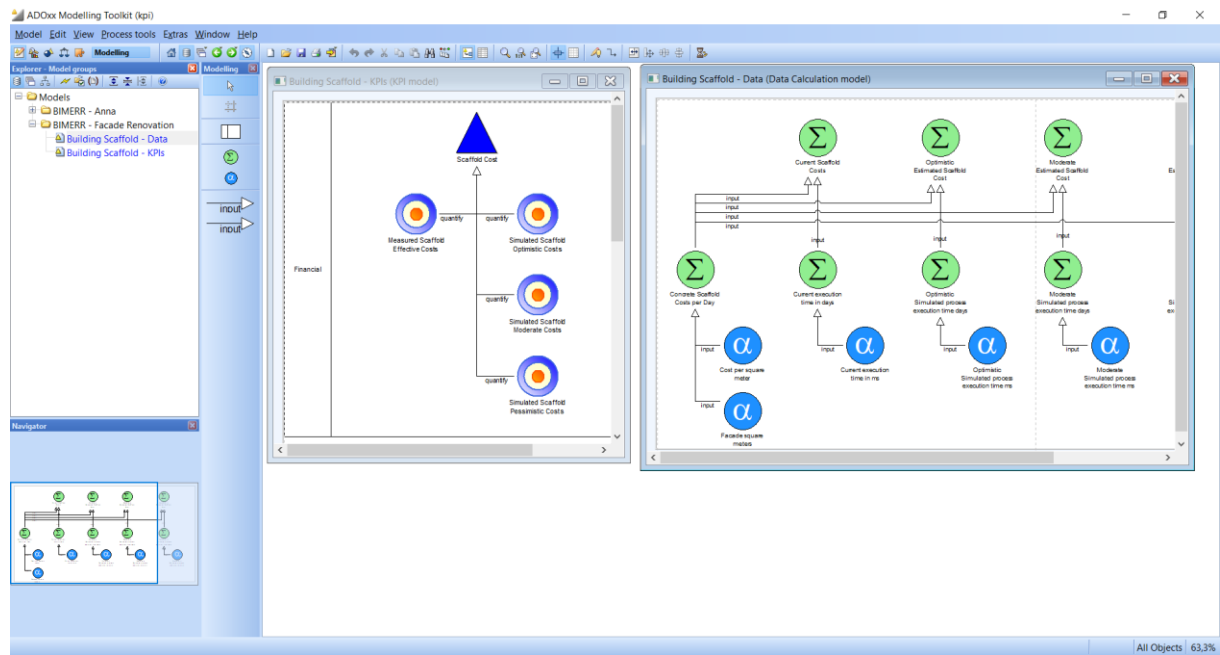


Figure 12 - Renovation process KPIs design tool community version

3. MONITORING AND EVALUATION TOOL FOR RENOVATION PROCESSES

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

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

3.1 RENOVATION PROCESSES-ORIENTED KPI DASHBOARDS

The KPI dashboard visualize in a combined view both the backward looking monitoring with the forward looking simulation results. The dashboard interface is based on configurable widgets where the KPIs can be visualized in different formats accordingly to the widget features. The KPIs definitions are taken directly from the renovation process KPIs design tool that is able to export them in a format recognized by the dashboard. The dashboard use the KPIs information reported in the model in order to automatically evaluate them, calculating the relative metrics and retrieving the data from the data sources using the right Olive microservice.

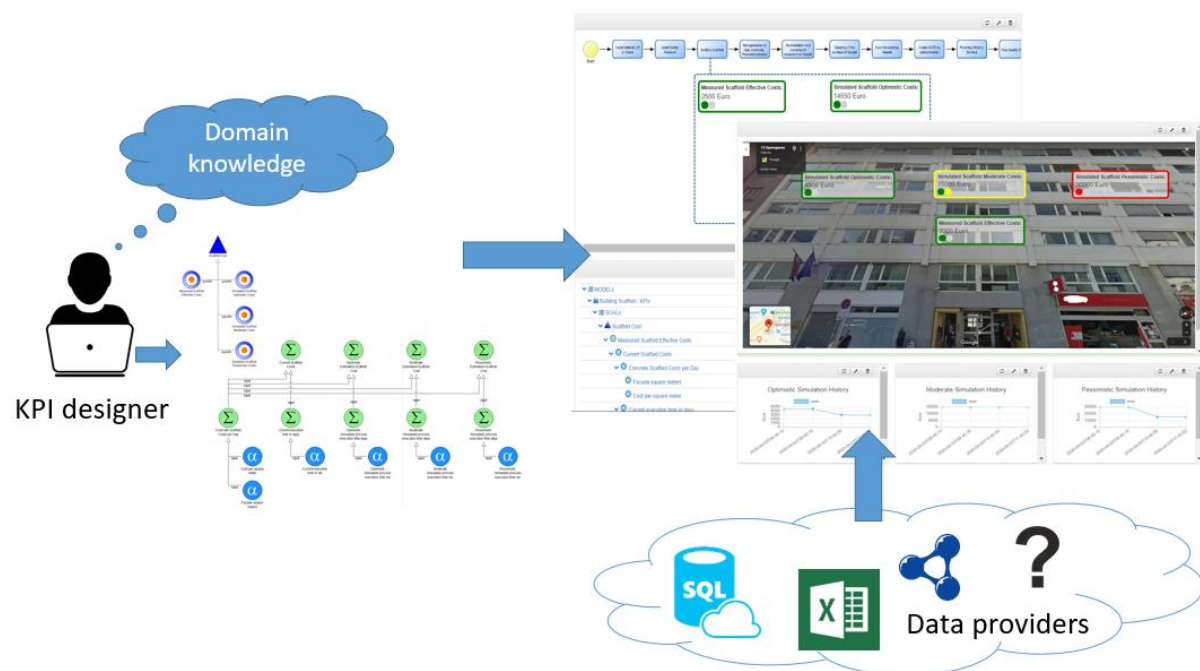


Figure 13 - Renovation KPI cockpit use case

3.1.1 Models-based Monitoring Dashboards demonstration

The KPI dashboard is strongly based on the KPI and data model used. A detailed description of the models created for the BIMERR demonstration is available in the D6.2 (BIMERR Consortium,2020). This models are focused on the scaffold costs and contain the definition of the following KPIs including their relative metrics and data sources:

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

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

The first dashboard use four widget to visualize the KPIs: one image map widget and three line chart widgets. The image map visualize the building facade to renovate using an image from Google Street View and overlay the values of the previously described KPIs using a color code that immediately reflect 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 line chart widgets are used to display the historical trend of the three simulation results visualizing in a Cartesian chart the estimated scaffold cost for every performed simulation over time, allowing to analyze the alignments of the simulation inputs with the real data in the past.

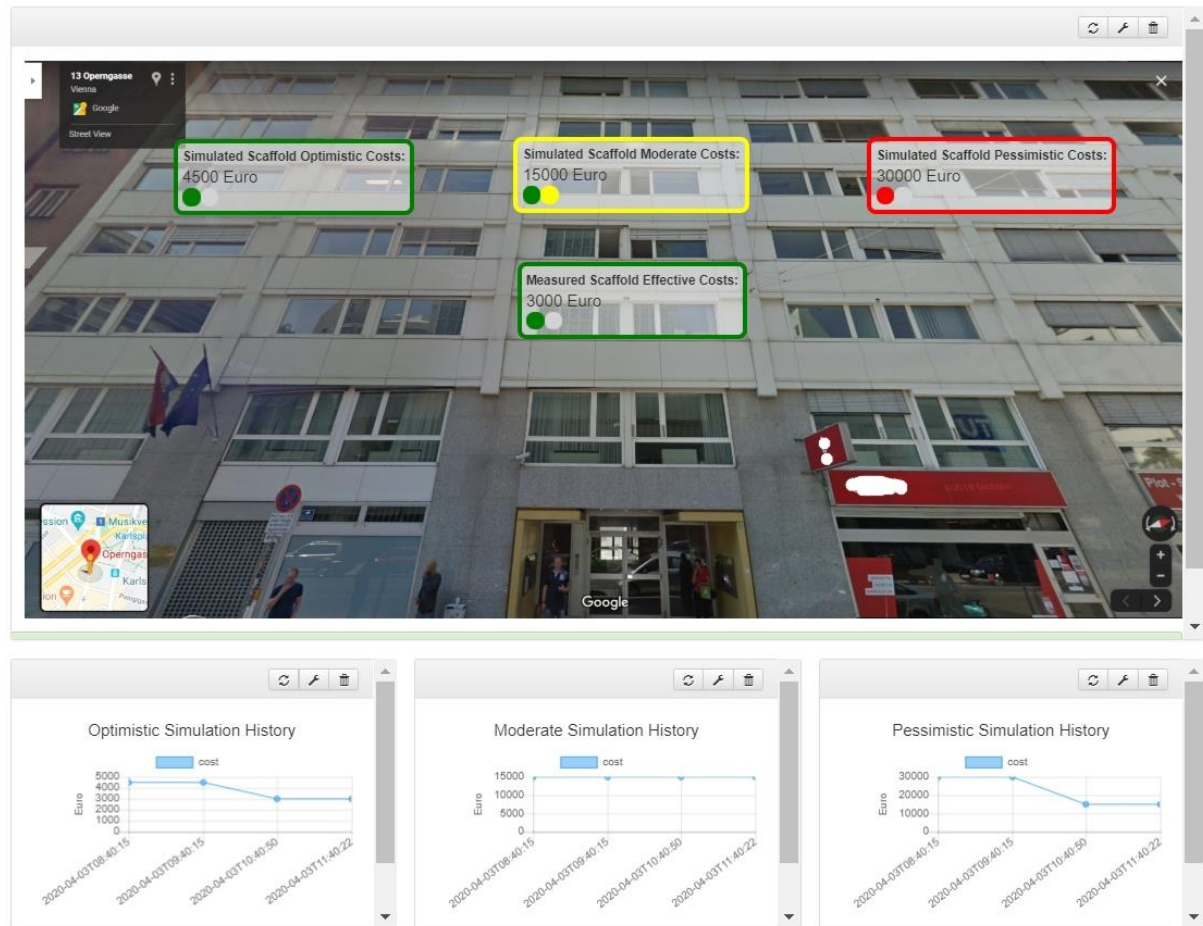


Figure 14 - Backward-looking Monitoring and Forward-Looking Simulation of KPI-Scaffold Costs

The second dashboard demonstrates the Process-Oriented Context of the dashboard by the possibility to link KPIs to different phases of the process. For 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 help to understand the cause of a failing KPIs checking how its dependencies behave, finding the root of the problem.

Hence the process-oriented representation is first an alternative visualization of the dashboard and second the possibility to additionally introduce the linkage of a process phases to a concrete KPI.

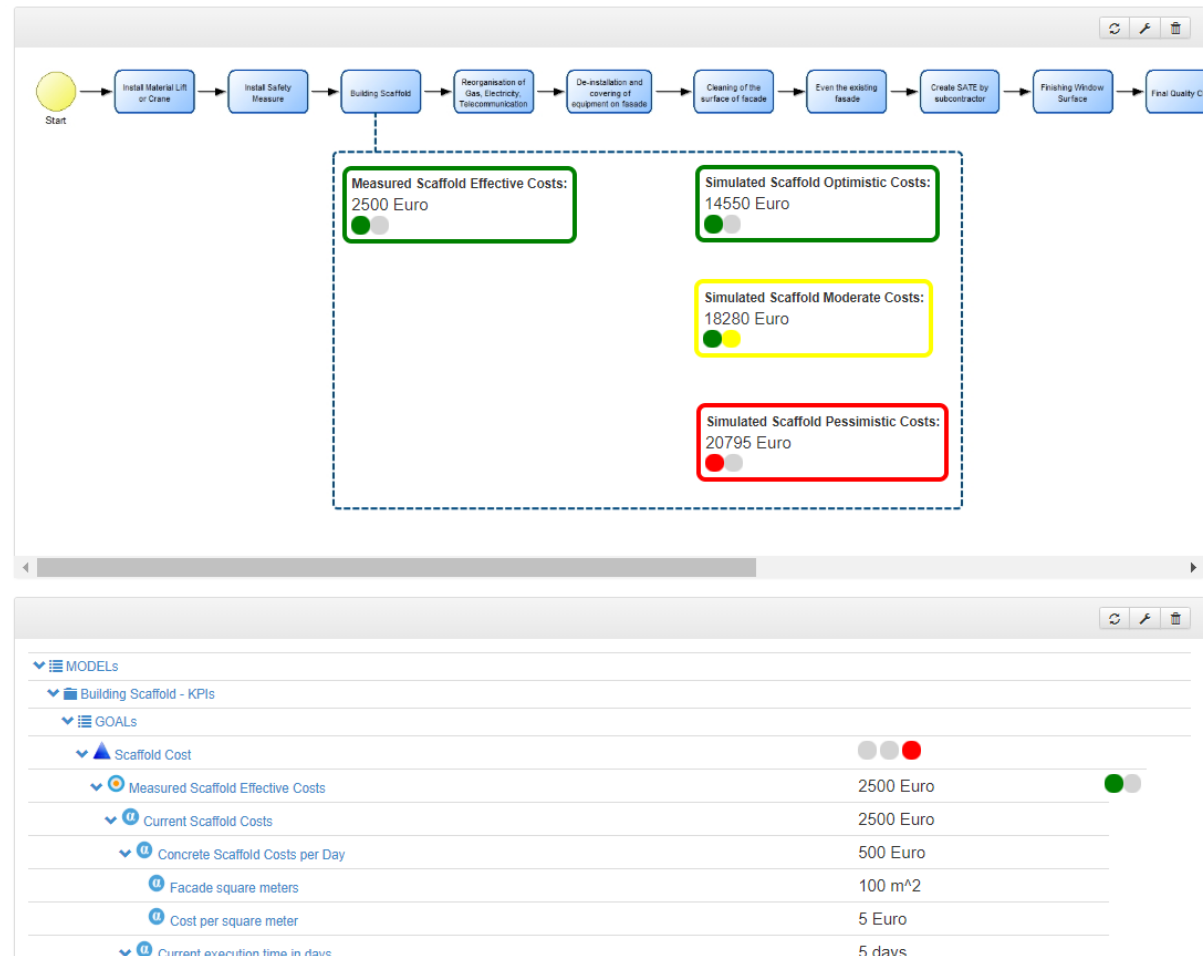


Figure 15 - Simulation Output for KPIs

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 in parallel running construction sites and each process for each construction site uses simulated KPIs, the aggregating complex process is the simulated using the dependencies of the underlying processes.

Such complex scenarios require complex modelling and knowledge externalization in the design phase, and hence may only be appropriate for 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 is able to perform a marriage between models and data resulting in a customizable web dashboard. The models contain information about how to retrieve the data and how to combine them in order to form 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 type of data sources. At 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.

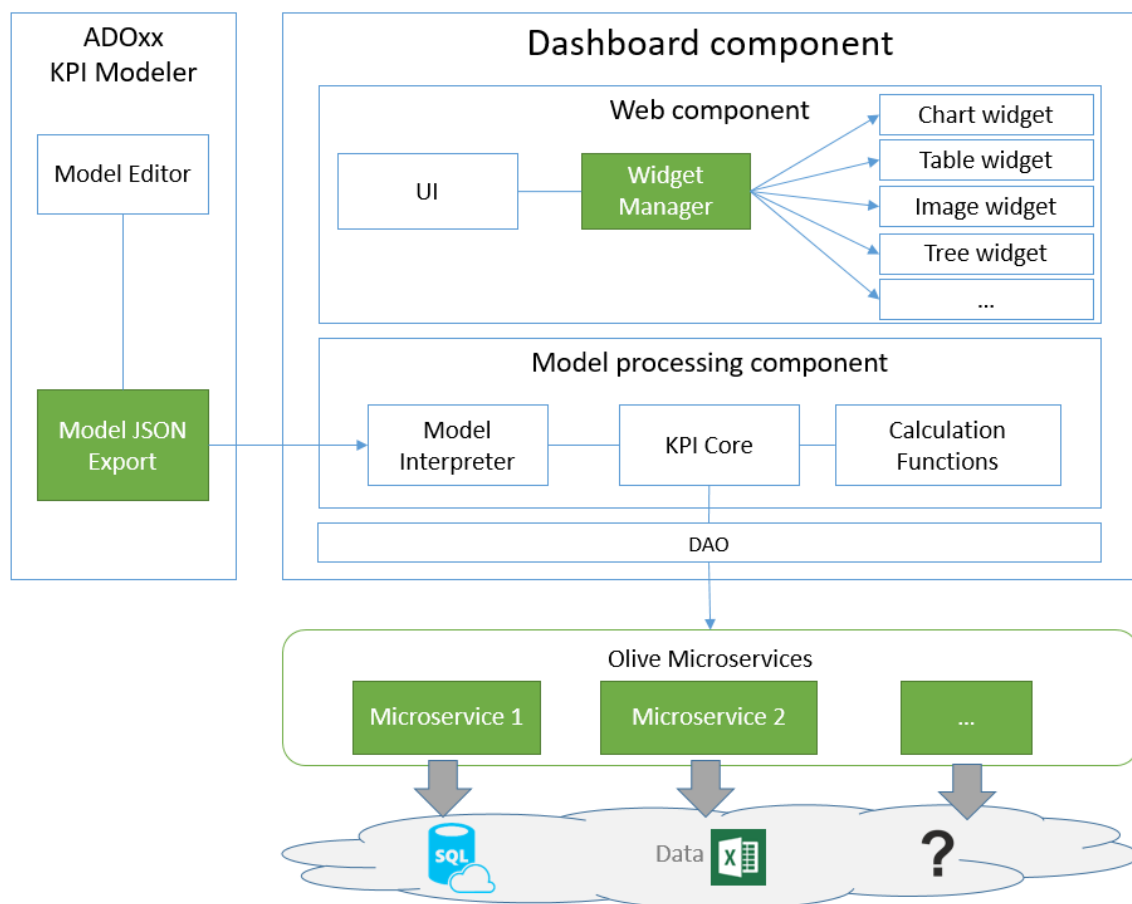


Figure 16 - KPIs Dashboard architecture

The dashboard accept KPIs models provided in a specific JSON format. The Renovation process KPIs design tool provide a feature to export the KPIs and data model in such specific JSON format. This model is processed by the model interpreter component that have 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 are processed by the KPI core component that is responsible to call the microservices described in the model in order to retrieve the data, apply the calculation functions as described in the model and make the resulting values available for the user interface. The interaction of the KPI core with the different microservices is helped by a DAO component that provide also caching features and an optimized microservice communication. The functions are evaluated in a component that is able to interpret and validate them in order to avoid security issues.

When the evaluated KPIs and Goals are available will be provided to the UI component on 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 provide out of the box four most commonly used widgets but extension are possible through a plug-in based mechanism:

- **Chart widget:** is able to visualize 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 can specify which KPI field show for every axis and an optional threshold line. As example the “Simulated Scaffold Optimistic Cost” KPI containing the data fields “cost” and “instant” can be visualized in a line chart selecting for the X axe the “instant” field and for the Y axe the “cost” field.

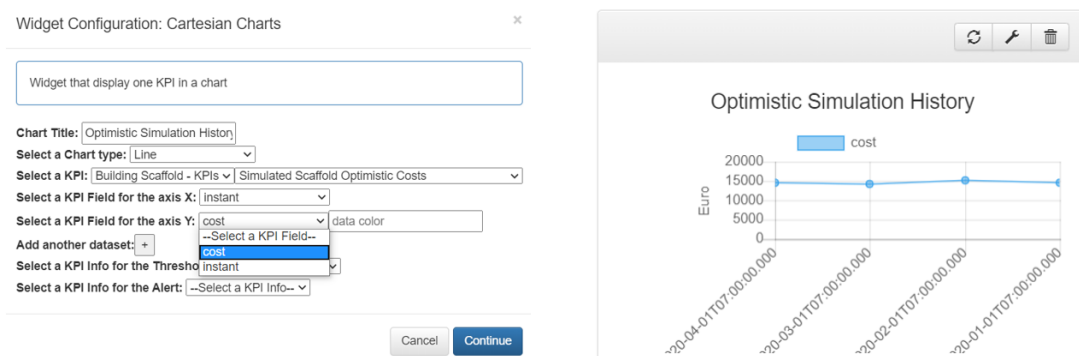


Figure 17 - KPI dashboard chart widget

- **Table widget:** this widget allow to visualize all the details of a configured KPI in a table format. All the values will be visualized as well as evaluation of the KPI status with a green, yellow, red indicator.

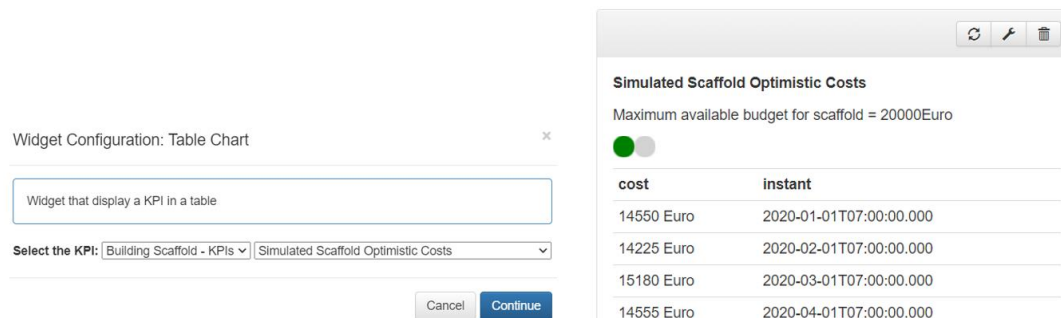


Figure 18 - KPI dashboard table widget

- **Image widget:** This widget allow to overlay one or more KPI details over an image. The KPIs are displayed with a color code that reflect the KPI status and details are visualized moving the mouse over the indicator.

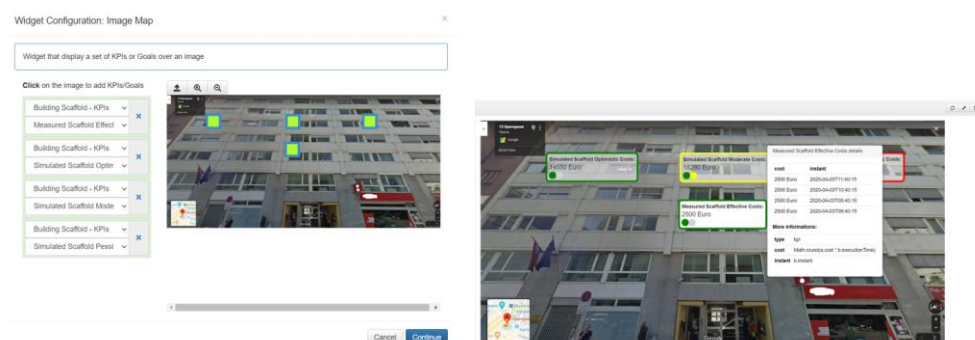


Figure 19 - KPI dashboard image widget

- **Tree widget:** This widget allow to visualize all the KPI in a collapsible tree view organized hierarchically or linearly on their dependencies. In this way is possible to identify the root cause of a problematic KPI or goal simplifying the behavior analysis.

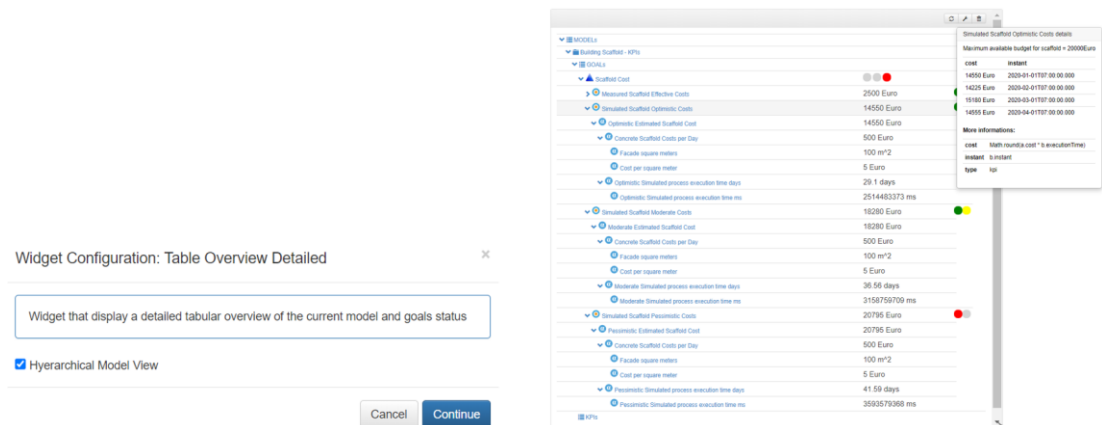


Figure 20 - KPI dashboard tree widget

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

3.2 SIMULATION TOOLS FOR RENOVATION PROCESSES

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

3.2.1 Simulation tool demonstration use case

The simulation tool requires as input first of all the renovation process workflow in standard BPMN format. The process to simulate 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. This format has been used as a temporary solution to give the user freedom to provide input using complex formulas and will be replaced in the final prototype by an integrated interface.

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



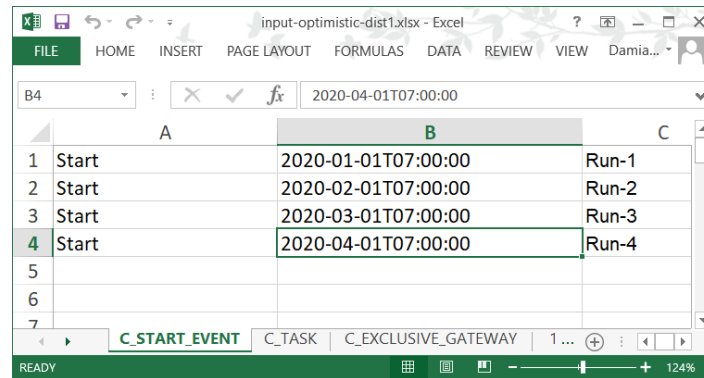
Figure 21 - Renovation process Simulation inputs

The Excel input file must be created accordingly the details described in D6.2 (BIMERR Consortium,2020).

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

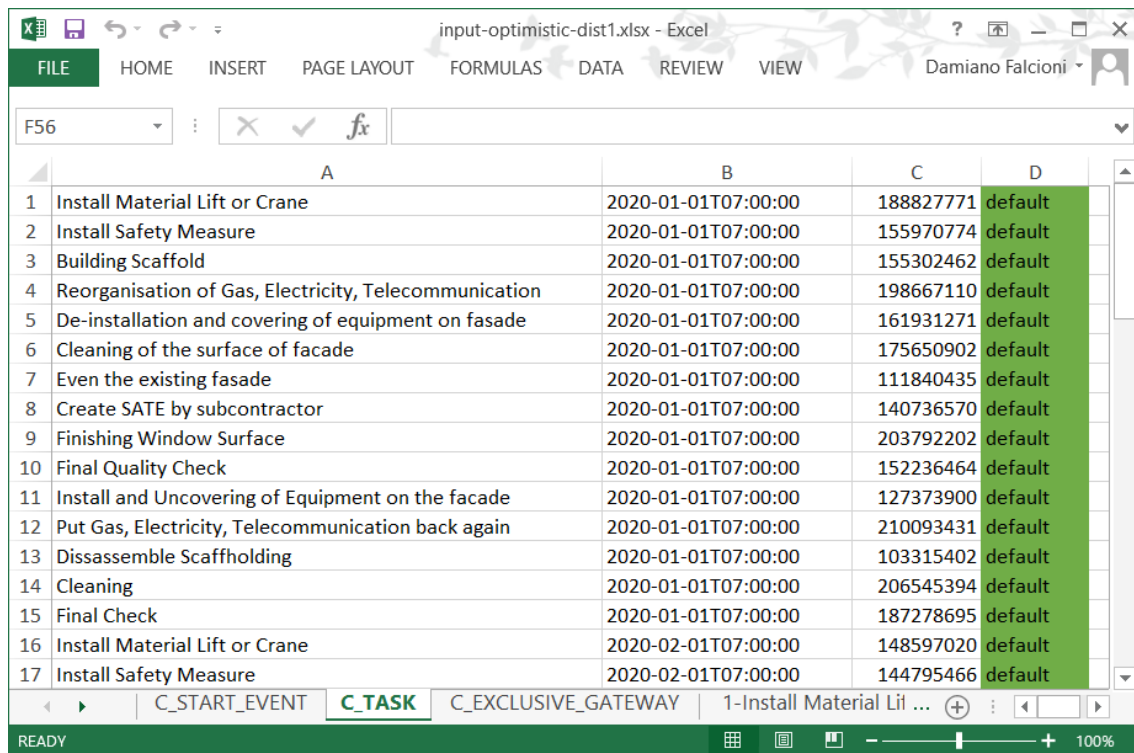
The C_START_EVENT sheet provide information on the number of simulation to run with their starting time and unique identifier. The first column of this sheet must contain the name of the starting event as reported in the BPMN model. The second column must contain the starting time of this specific event while the third column an unique id to associate to the simulation run in order to have a reference later 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 22 - Renovation process simulation input C_START_EVENT

The C_TASK sheet contain 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 is an optional starting time that can be used to provide an additional waiting time before the activity start. By default the activity start as soon as the previous one is terminated. The third column represent 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 forth column can contain the unique id of the simulation run to be used for or the value default 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 23- Renovation process simulation input C_TASK

In case of choices the C_EXCLUSIVE_GATEWAY sheet must be filled. This allow 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 an optional waiting time to postpone the choice execution and the third column the probability value of all its outgoing sequence flows.

As introduced previously, additional sheets can be present in order 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 D6.2 (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-contractors problems and (5) the delay introduced by unexpected events.

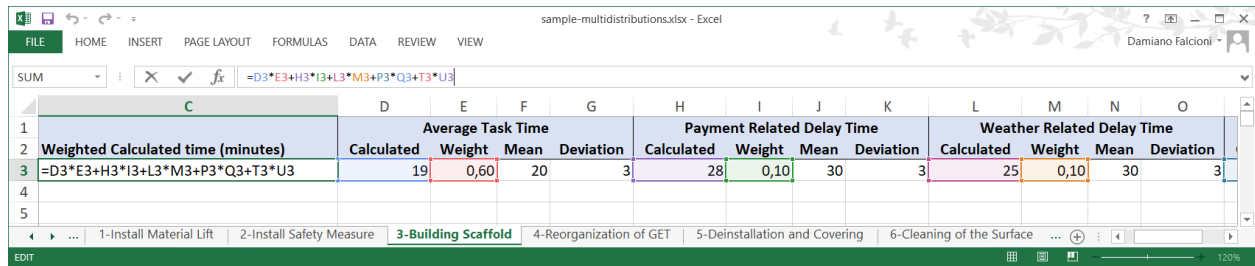


Figure 24 - Renovation process simulation input calculation

As soon as both inputs are ready the simulation can start. Once completed, the results are visualized in two different forms: an overview of times and cost with path probabilities and generic information of the process and a detailed view in 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

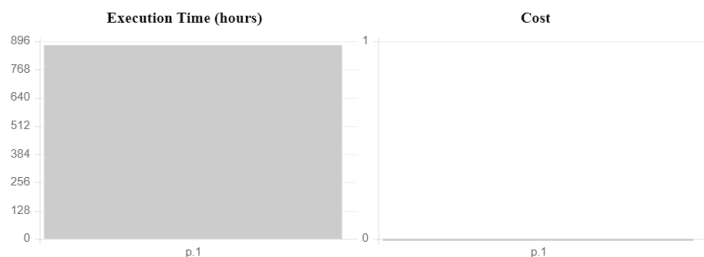
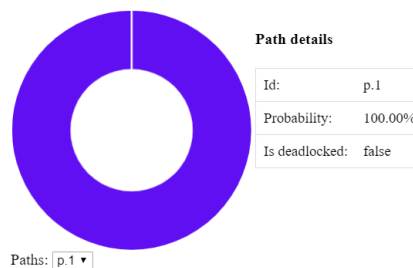


Figure 25 - Renovation process simulation general results

The general results in particular contains the following data:

Name	Measure	Details
Average Cost	Average cost during all of the simulation runs	-
Max Cost	Max cost during all of the simulation runs	Trace name that contains this cost

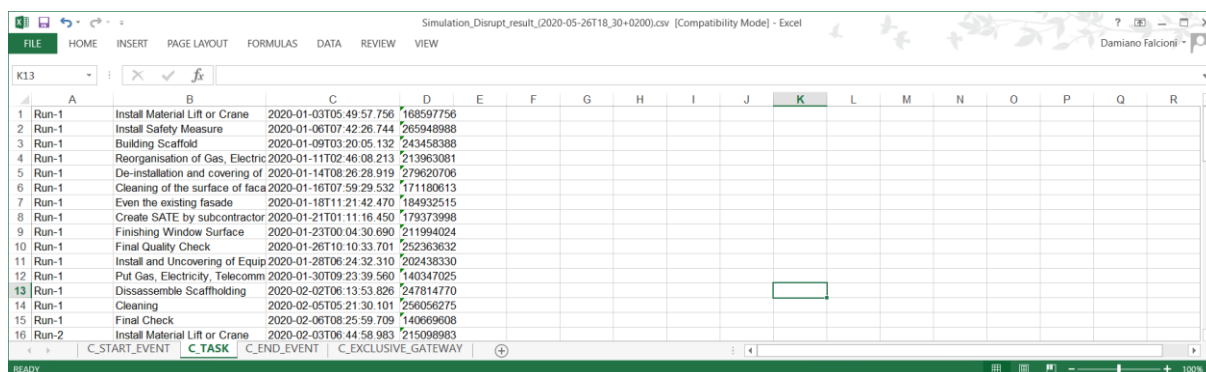
Min Cost	Min cost during all of the simulation runs	Trace name that contains this cost
Total Costs	Sum of all costs during all of the simulation runs	-
Average Executions Time	total execution time / total simulation runs number	-
Max Executions Time	Max execution time during all of the simulation runs	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 simulation 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 of the same structure used as input but with detail on the simulated starting and execution times.

The C_TASK sheet in particular will contain 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 contain for every started simulation its ending time.



	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	2020-01-11T02:46:08.213	213963081														
5	Run-1	De-installation and covering of	2020-01-14T08:26:28.919	279620706														
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	Disassemble Scaffolding	2020-02-02T06:13:53.626	247814770														
14	Run-1	Cleaning	2020-02-05T05:21:30.101	256056275														
15	Run-1	Final Check	2020-02-06T08:25:59.709	140669608														
16	Run-2	Install Material Lift or Crane	2020-02-03T06:44:58.983	215098983														

Figure 26 - 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 use the Petri Net logics in order to simulate processes and workflow provided in BPMN2.0 formats and it 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 show the results in a user friendly way.

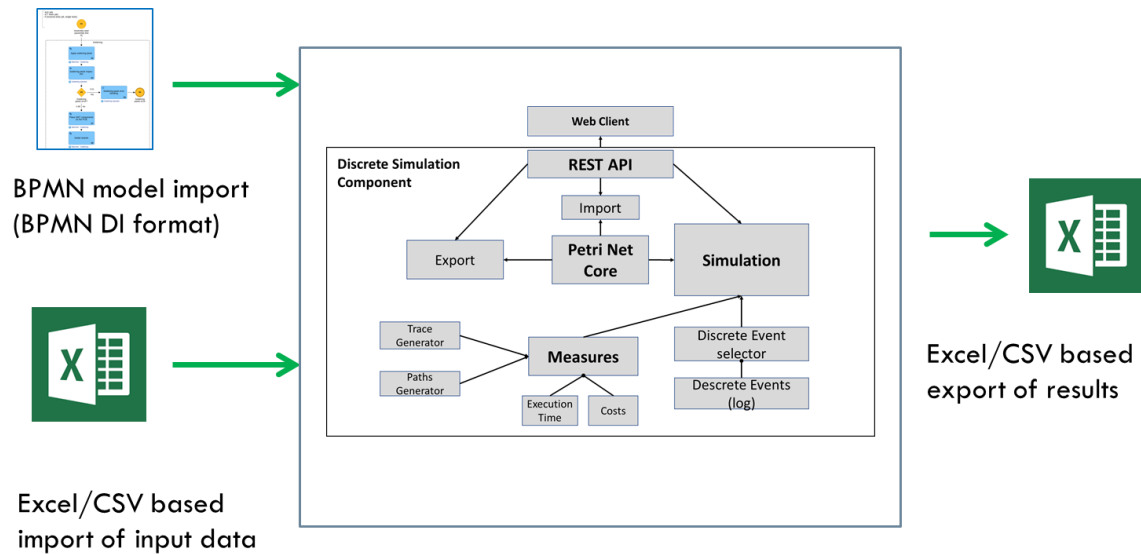


Figure 27 - Renovation process simulation engine architecture

A description of the main component of the simulator are provided in the following:

The **petri net core module** is the component that contain the main logic of a petri net and manage its semantic. The simulation service use this component in order to evaluate at each step which transition can be enabled.

The **import module** is an easy to extend component that is able to automatically recognize the format of the provided model and convert it in the internal petri net structure. It manage separately the logic of document parsing and of object mapping in order to reuse the same mapping logic for multiple file format (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.

The **export module** is for diagnostic only. It give the possibility to export the internal petri net structure in PNML standard format in order to be visualized in any supported editor.

The **simulation measures module** is an easy to extend component that give the possibility to define listeners for the simulation event. Each listener produce 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 in order to calculate some final indexes (like average values).

The ***discrete event selector module*** is the component that perform the choice of the transition to execute between the available one. The module provide a base mechanism that perform a fair choice between parallel transitions and a user defined probabilistic choice between concurrent transitions. The base mechanism has been also extended in order to support dynamic probability evaluation using a scripting system.

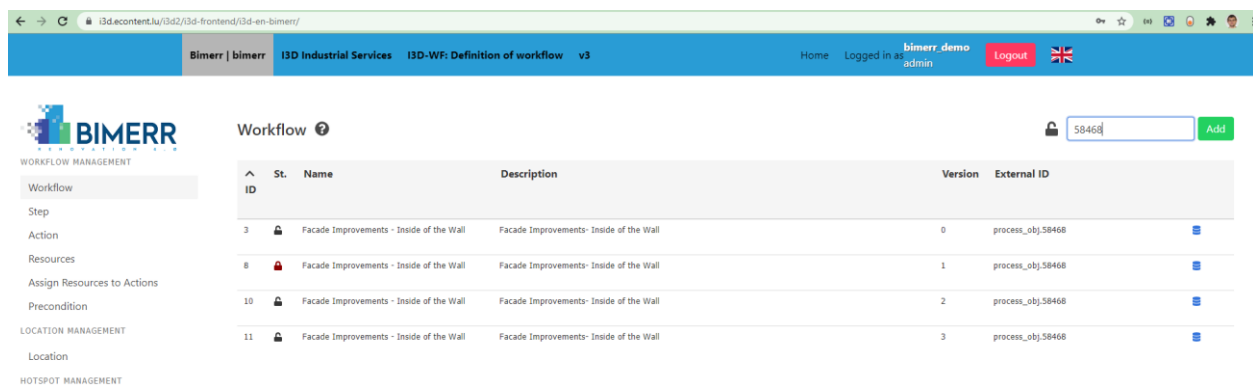
The ***simulation module*** is the component that manage all the simulations, invoking the functionalities of the measures module and of the transition choice. It is also responsible for the generation of the simulation output in a structured format.

The full documentation including the source code and example of features extension is available to the community through the ADOxx portal <https://www.adoxx.org/live/dashboard-version-2>.

3.3 CREATION OF DIGITAL TWIN WITH WORKFLOW EXECUTION

Once the process model of the reconstruction process has been created, it is time for execution. The only input used by the Workflow Execution tool is a BPMN-DI file, which contains the tasks of the reconstruction process with all the attributes modeled in the previous phases of the preparation.

The process model in form of BPMN-DI file is imported to the I3D system and is transformed to the internal structure of it. It is handled as a workflow template and in case of need can be adjusted before it is used.



ID	St.	Name	Description	Version	External ID
3	🔒	Facade Improvements - Inside of the Wall	Facade Improvements- Inside of the Wall	0	process_obj.58468
8	🔒	Facade Improvements - Inside of the Wall	Facade Improvements- Inside of the Wall	1	process_obj.58468
10	🔒	Facade Improvements - Inside of the Wall	Facade Improvements- Inside of the Wall	2	process_obj.58468
11	🔒	Facade Improvements - Inside of the Wall	Facade Improvements- Inside of the Wall	3	process_obj.58468

Figure 28 - Renovation workflow process imported to I3D

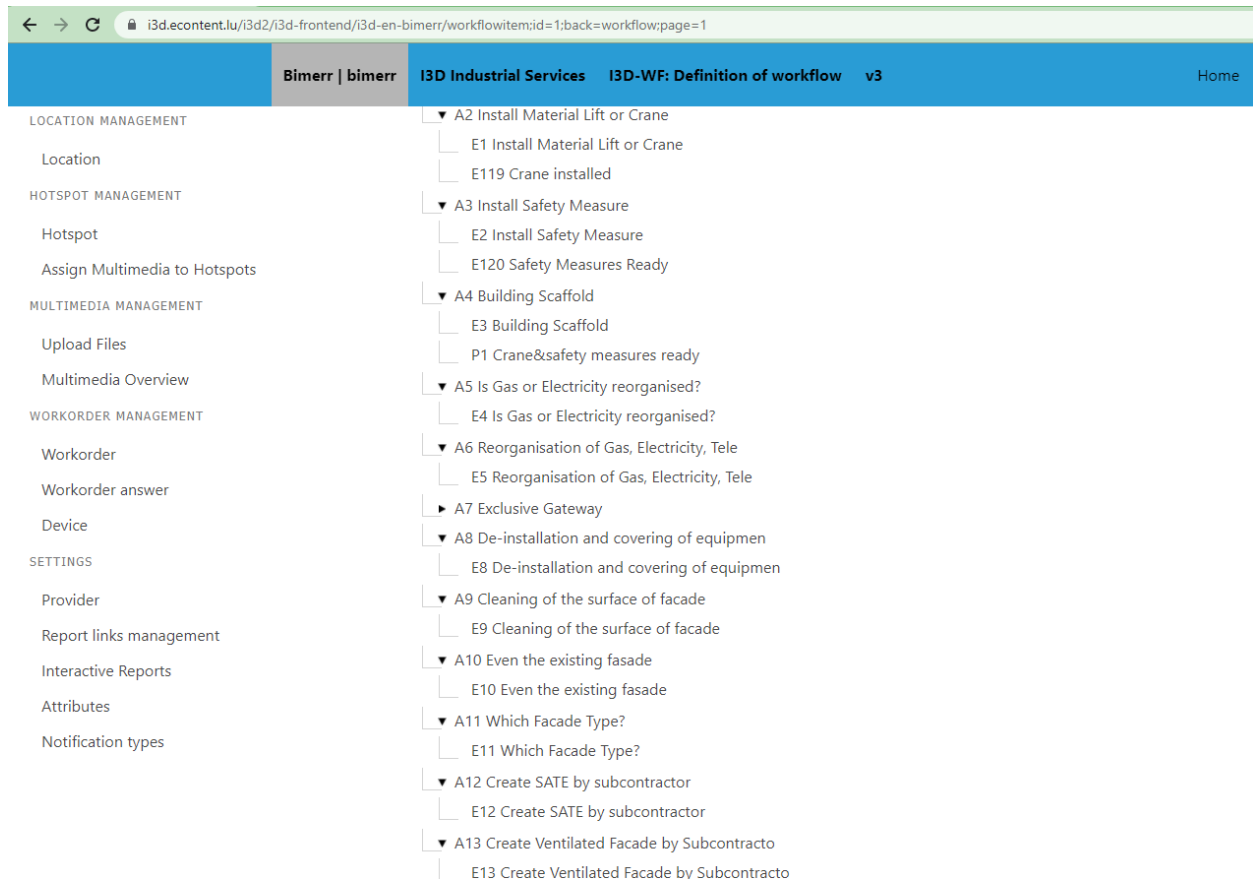


Figure 29 - Adjustable reconstruction process template

The template is used by the project manager to generate a real instance of the reconstruction process, which is called workorder in the I3D system. A visualization of the workorder is available for the project manager to make it easier to follow the project and to do interactions with its objects.

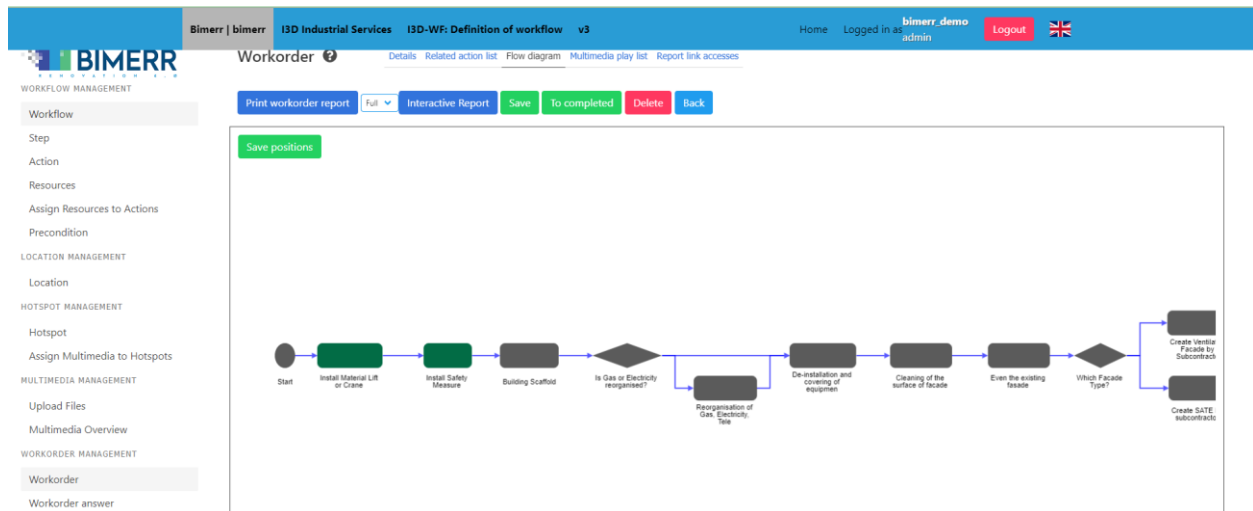
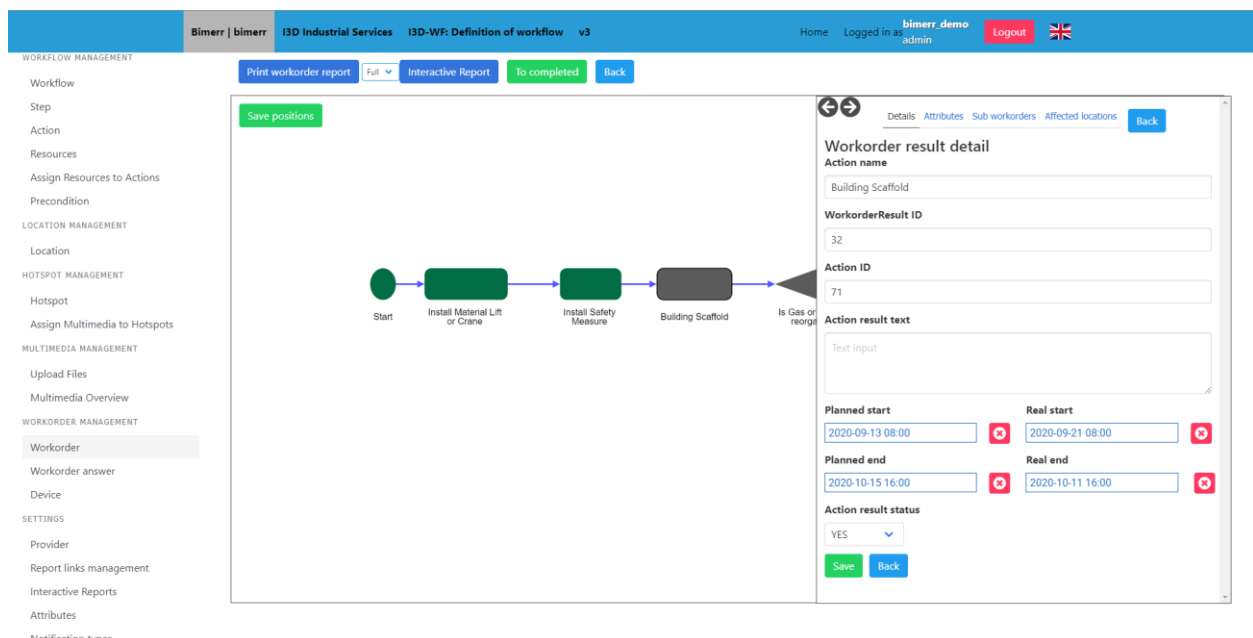


Figure 30 - Visualized running reconstruction process



This screenshot shows the 'Workorder result detail' form within the BIMERR Workorder interface. The top navigation bar and left sidebar are identical to Figure 30. The main content area features a 'Print workorder report' button and a dropdown menu for 'Full', 'Interactive Report', 'To completed', and 'Back'. Below this is a 'Save positions' button and a flowchart showing the process up to the 'Is Gas or Electricity Reorganised?' decision point.

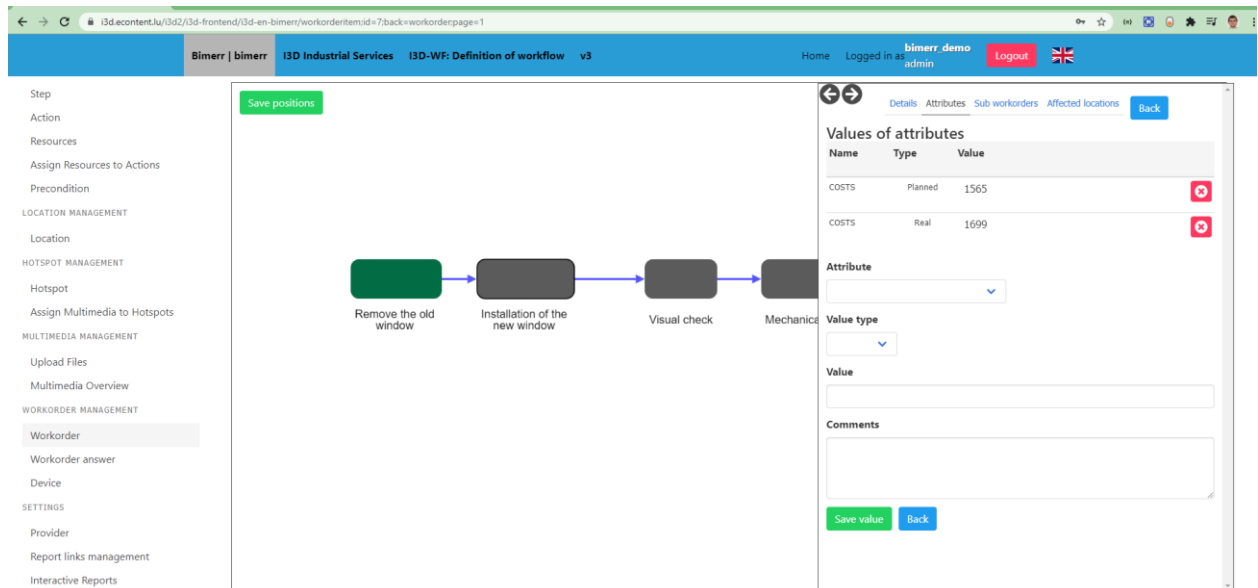
The 'Workorder result detail' form is open, displaying the following fields:

- Action name:** Building Scaffold
- WorkorderResult ID:** 32
- Action ID:** 71
- Action result text:** Text input field
- Planned start:** 2020-09-13 08:00
- Real start:** 2020-09-21 08:00
- Planned end:** 2020-10-15 16:00
- Real end:** 2020-10-11 16:00
- Action result status:** YES (dropdown menu)

At the bottom of the form are 'Save' and 'Back' buttons.

Figure 31 - Interaction with the objects of the reconstruction process

All the attributes presented in the BPMN-DI are used and displayed as planned values and the project manager can confirm them or record his own values as real data generated during the reconstruction process.



The screenshot shows the BIMERR application interface. On the left is a sidebar menu with categories like Step, Action, Resources, Assign Resources to Actions, Precondition, LOCATION MANAGEMENT, Location, HOTSPOT MANAGEMENT, Hotspot, Assign Multimedia to Hotspots, MULTIMEDIA MANAGEMENT, Upload Files, Multimedia Overview, WORKORDER MANAGEMENT, Workorder, Workorder answer, Device, SETTINGS, Provider, Report links management, and Interactive Reports. The main area displays a workflow diagram with four steps: 'Remove the old window', 'Installation of the new window', 'Visual check', and 'Mechanical'. A 'Save positions' button is at the top left. On the right, a panel titled 'Values of attributes' shows a table with columns 'Name', 'Type', and 'Value'. It lists 'COSTS' with 'Planned' value 1565 and 'Real' value 1699. Below the table are fields for 'Attribute', 'Value type', 'Value', and 'Comments', with 'Save value' and 'Back' buttons at the bottom.

Figure 32 - Evidence of planned and collected attributes

All the planned and collected data are provided for further analysis and re-planning of the process with the goal to make the next reconstructions more effective.

```

55 *      {
56         "type": "electrical",
57         "notes": "Please be advised that during this task, electricity will be cut in this area"
58       },
59 *      {
60         "type": "noise",
61         "notes": "Please be advised that during this task, noise levels will be higher than normal"
62       }
63     ],
64   },
65   "building_components": [
66     {
67       "id": "sdgskgwfsdg545ldfd",
68       "type": "Wall",
69       "name": "Wall3484"
70     },
71     {
72       "id": "dknfdk49h30152shfjhd",
73       "type": "DoorFrame",
74       "name": "DoorFrame36"
75     }
76   ],
77   "progress_kpis": [
78     {
79       "id": "kpiId",
80       "name": "work completed",
81       "value": "0.3"
82     },
83     {
84       "id": "kpiId2",
85       "name": "hours spent vs hours planned",
86       "value": "0.8"
87     }
88   ]
89 },

```

Figure 33 - Output of the execution engine – list of attributes and notifications connected to the tasks

```
{
  "workflow": {
    "header": {
      "id": 53,
      "name": "Facade Improvements - Outside of the Wal",
      "building_id": 379930,
      "project_id": "process_obj.58134"
    },
    "workorders": [
      {
        "header": {
          "id": 32,
          "name": "Test flow graph2",
          "description": "Facade Improvement Template",
          "start": {
            "planned_start": "2020-05-21 10:08:00",
            "actual_start": null
          },
          "finish": {
            "planned_finish": "2020-05-21 10:08:00",
            "actual_finish": null
          }
        },
        "tasks": [
          {
            "id": 403,
            "name": "Start",
            "description": "obj.58004| Start for Start",
            "execution_time": {
              "planned": [
                ],
              "real": [
                ]
            }
          }
        ]
      }
    ]
  }
}
```

Figure 34 - Export of all the executed instances connected to a reconstruction process template

Since the process imported from the BPMN model can have different level of details, the UI for the project manager allows to assign sub-workflows to any task of the reconstruction process. This allows to create tasks with higher granularity, than the overall process.

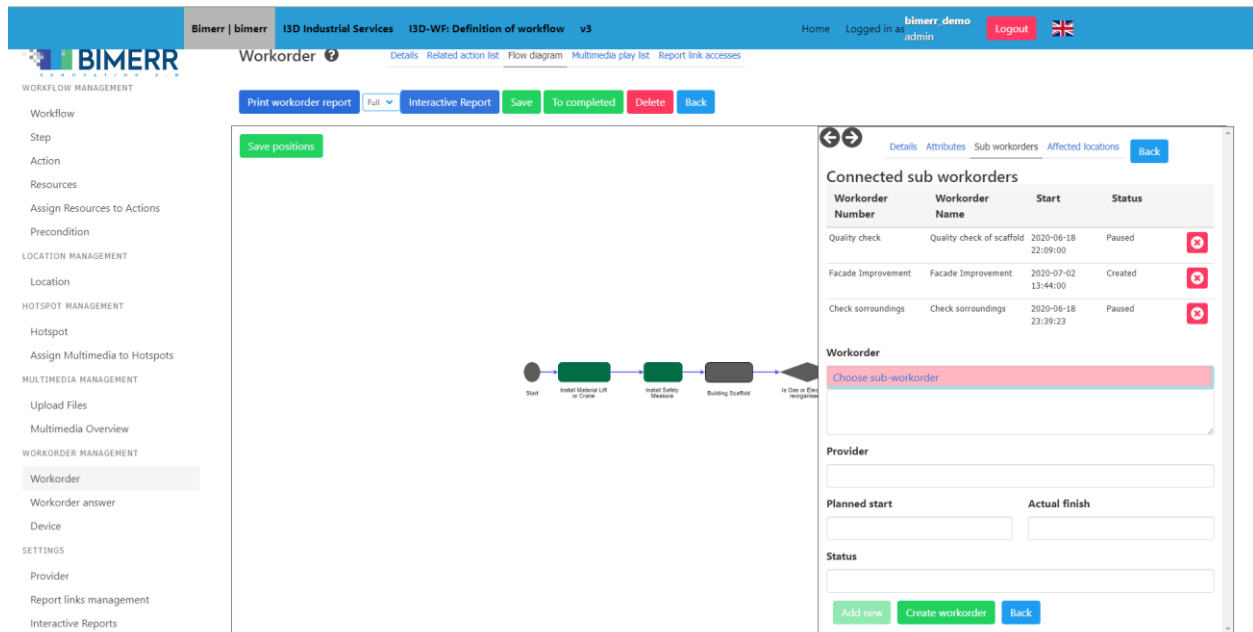


Figure 35 - Sub processes of the task

The sub processes can be executed and monitored via the web-based UI (prototype displayed on figures above). The application for on-site support of the workers to execute the assigned processes is displayed on figures below. It is designed to be used primarily on Head Mounted Devices – smart glasses (Figure 36). The current version is optimized for Head Mounted Device Realwear HMT-1, the final version of the application will support Microsoft HoloLens 2, too. The design of the application is optimized both for indoor and outdoor use where the dark background with bright text provides the best readability for the user under different light conditions (bright text on dark background worked the best for the users of the application). The navigation of the application is made with a focus to provide the user the option to select the control which fits the best to the actual user need and smart glasses used. The built-in voice-recognition of Realwear HMT-1 glasses is fully supported. As alternative option, a gyroscope-based controller has been implemented, which allows to navigate in the menu by head movements. Alternative controllers, such as the touchpad in Google EE2 or pointer in ThirdEye Gen X2 smart glasses are supported, too. The same application can be alternatively installed and used on mobile phones and tablets as a back-up solution for on-site users (e.g. in case of broken smart glasses).



Figure 36 – Smart glasses application for on-site support of workers running on Head Mounted Display

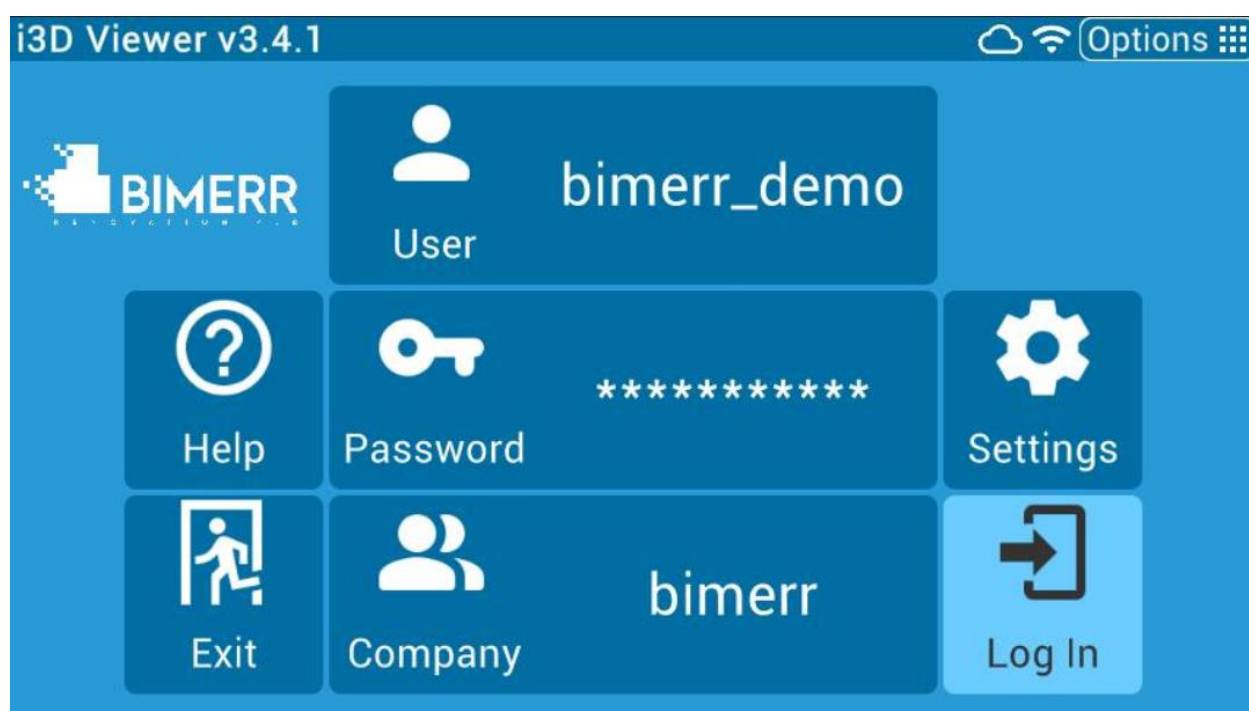


Figure 37 – Smart glasses application for on-site support of workers – log-in screen

Select work order Options

^ Scroll Up

1. Doors installation
2. Check surroundings
3. Quality check of scaffold
4. Windows installation
5. Facade Improvement
6. Scaffold installation
7. Windows replacement

v Scroll Down

Work Order Description

Description: Doors installation

Start: 2020-06-18 22:00:00

End: 2020-09-18 22:00:00

← Back

ⓘ Details

✓ Confirm

Figure 38 - Application for on-site support of workers to execute assigned tasks – list of assigned tasks

Details Options

← Back

Doors installation

Status: Running

Planned Start: 2020-06-18 22:00:00

Planned End: 2020-09-18 22:00:00

Planned Duration: 0m 0s

Actual Start:

Actual End:

Actual Duration:

Lead Provider: 2

Author: 2

ⓘ DETAILS

⌕

Figure 39 - Details of the assigned task

In case, the task has assigned sub-processes, the attributes are aggregated from the sub-task and summarized.

The web-based UI provides for the Project manager also the option to mark, which locations are affected by each task or subtask as well as to indicate Health and Safety issues connected to the tasks of the running reconstruction workorder.

4. REFLECTION AND INNOVATION TOOLS FOR RENOVATION PROCESSES

In this chapter the set of component 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 Celonis³ 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 allow to analyze log files and construct custom analytical dashboards. Its free version Celonis Snap 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 will require some manual processing in order to be accepted as valid inputs for Celonis Snap. Improvement on this process with some automations will be made available in the final prototype.

4.1.1 *Logs preparation for Celonis*

Celonis requires one single CSV file as input. The results of the workflow must so be combined on one Excel worksheet that, afterwards, must be converted to a CSV file with field separator

³ <https://www.celonis.com/>


“;” 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 in order to interpret its data. This association have to 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.



#	COL_0	COL_1	COL_2	COL_3
1	Run-1	Install Material Lift or C...	2019-06-03T02:06:53...	1406836
2	Run-1	Install Safety Measure	2019-06-03T02:31:37...	1321106
3	Run-1	Building Scaffold	2019-06-03T02:06:53...	1285785
4	Run-1	Reorganisation of Gas...	2019-06-03T02:54:01...	1483585
5	Run-1	De-Installation and co...	2019-06-03T03:15:28...	1344267
6	Run-1	Cleaning of the surfac...	2019-06-03T03:40:16...	1286459
7	Run-1	Even the existing fesa...	2019-06-03T04:00:49...	1488195
8	Run-1	Create SATE by subco...	2019-06-03T04:21:44...	1232795
9	Run-1	Finishing Window Surf...	2019-06-03T04:44:11...	1255389
10	Run-1	Final Quality Check	2019-06-03T05:07:58...	1347177
11	Run-1	Install and Uncovering...	2019-06-03T05:28:15...	1427117
12	Run-1	Put Gas, Electricity, Tel...	2019-06-03T05:47:42...	1217234
13	Run-1	Disassemble Scaffold...	2019-06-03T06:11:26...	1666510
14	Run-1	Cleaning	2019-06-03T06:36:58...	1424376
15	Run-1	Final Check	2019-06-04T01:23:52...	1531602
16	Run-2	Install Material Lift or C...	2019-06-04T01:23:52...	1432115

Case ID	Activity Name	Timestamp
Run-1	Install Material Lift or Cra...	2019-06-03T01:23:26.836
Run-1	Install Safety Measure	2019-06-03T01:45:27.942
Run-1	Building Scaffold	2019-06-03T02:06:53.727
Run-1	De-Installation and cove...	2019-06-03T02:54:01.579
Run-1	Cleaning of the surface ...	2019-06-03T03:15:28.380
Run-1	Even the existing fesa...	2019-06-03T03:40:16.233
Run-1	Create SATE by subcont...	2019-06-03T04:00:49.2...
Run-1	Finishing Window Surface	2019-06-03T04:21:44.417
Run-1	Final Quality Check	2019-06-03T04:44:11.594
Run-1	Install and Uncovering o...	2019-06-03T05:07:58.711
Run-1	Put Gas, Electricity, Telec...	2019-06-03T05:28:15.945
Run-1	Disassemble Scaffold...	2019-06-03T05:47:42.455
Run-1	Cleaning	2019-06-03T06:11:26.831
Run-1	Final Check	2019-06-03T06:36:58.433

Figure 40 - Celonis logs preparation

4.1.2 Creation of Analysis Workspace

First, a new workspace for analysis have to be created. This 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 allow by default to visualize 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 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:** allow to visualize the workflow generated from the logs, including all the variants. It allow to compare it with the original workflow in order to identify incorrect behaviors.
- **Cases Variants:** Is a table widget that allow to visualize 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:** Is a table widget that allow to visualize the frequency of every choice. This is useful to identify if the data used for the process simulation was 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:** Is a table widget that allow to visualize 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")))

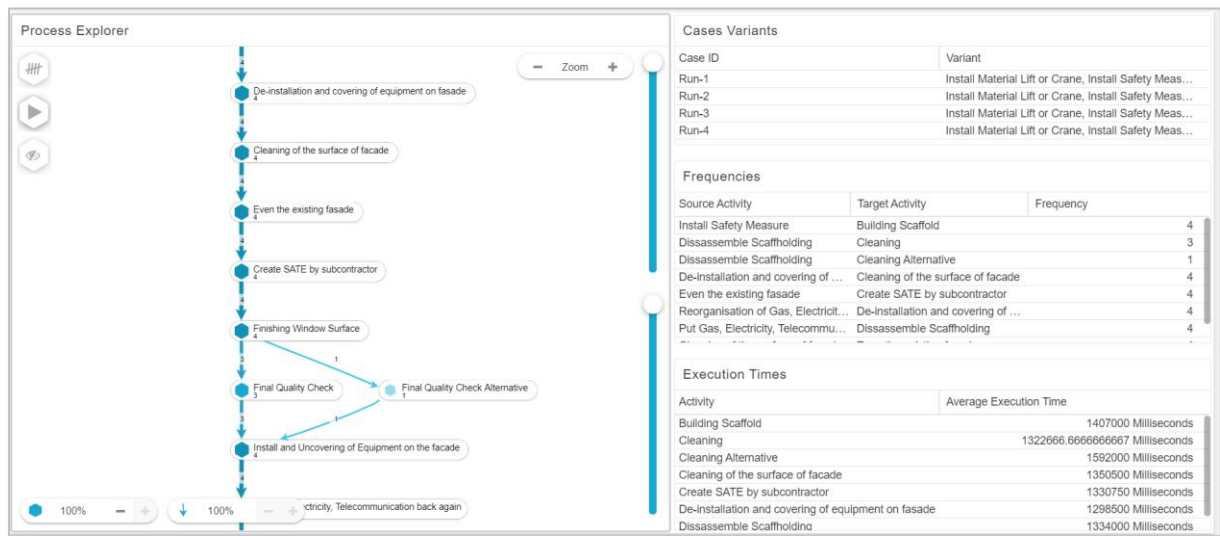


Figure 41 - Celonis Analysis Workspace for BIMERR

In order to enable the exports of the results the created workspace need to be configured. In the Edit mode the user must 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 allow to have a right click menu entry on every widget that allow to generate a BPMN file for the process in order to be imported in the modelling environment and compare with the original workflow, and to generate CSV files for every one of the tree table available in our workspace.

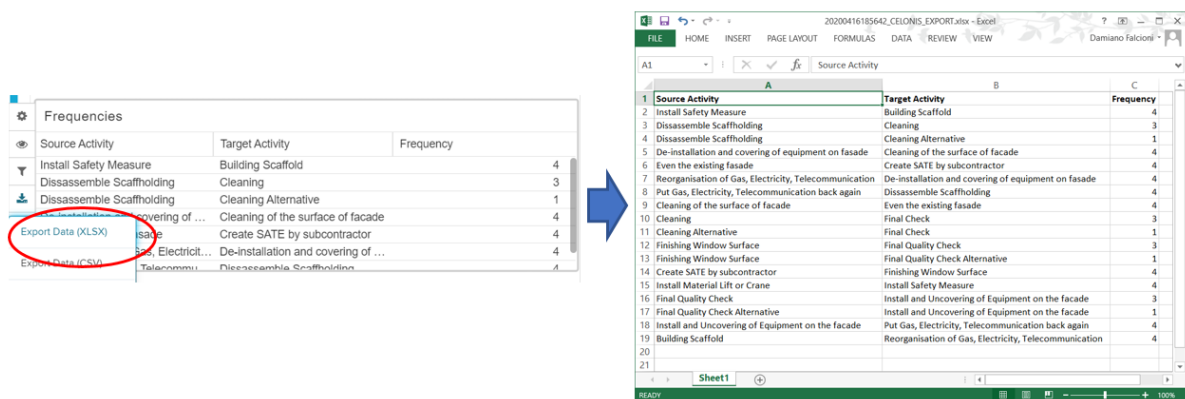


Figure 42 - Excel Output

4.2 COLLABORATIVE REFLECTION OF RENOVATION PROCESS

Beginning with people working in collaboration, a 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. Collective intelligence is measured by the collective IQ. 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. In order 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, Wikipedia 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 XWiki⁴ allows commenting models and retrieving comments. Furthermore, a kind of review process is included.

4.2.1 Model Wiki application

The Model Wiki web application allow to generate xWiki pages from any model in the ADOxx modelling environment and as soon as the pages are generated allow to import 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 created a model or imported in ADOxx, the web application is able to retrieve it and using the "Model Export" user interface allow the user to automatically generate a series of xWiki pages. An external reviewer can then look at the generated wiki pages and collaborate on the model, commenting the relative wiki page. Through the "Import wiki comments" user

⁴ <https://www.xwiki.org/xwiki/bin/view/Main/WebHome>

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

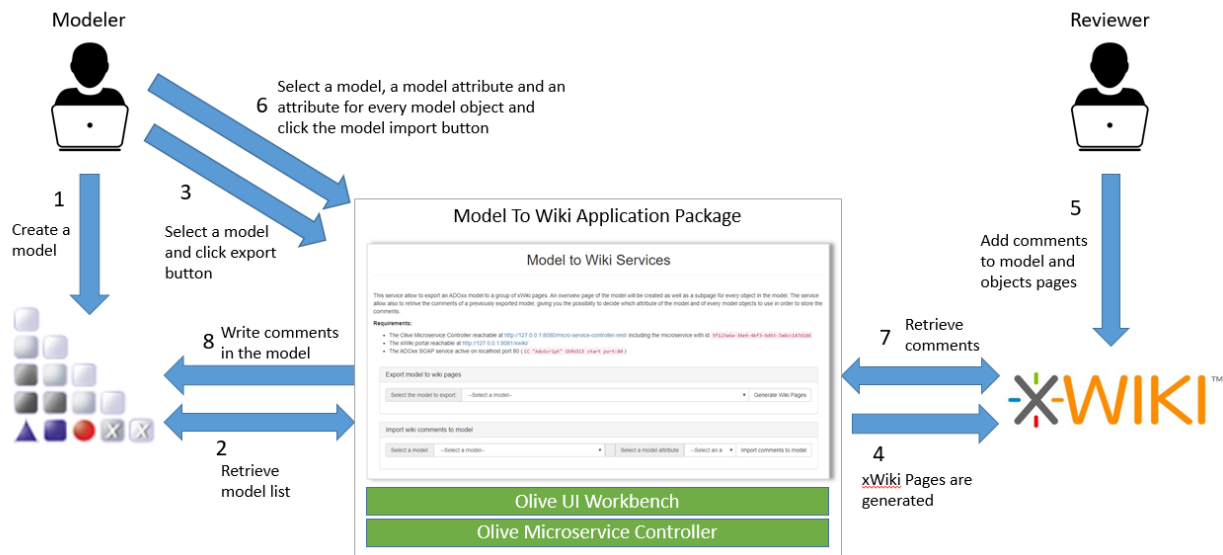


Figure 43 - Model Wiki use case scenario

The web application uses the Olive microservices to communicate with the ADOxx modelling environment in order 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 model, (4) retrieve all the attribute 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 also for the communication with the xWiki platform in order to (1) create an xWiki page and (2) retrieve the comments of a specific xWiki page.

The user interface of the web application has been made using the Olive UI Workbench. In particular the UI is composed of two widgets, named "Model-to-wiki UI" and "Wiki-to-model UI" and displayed in sequence in the main rendering page.

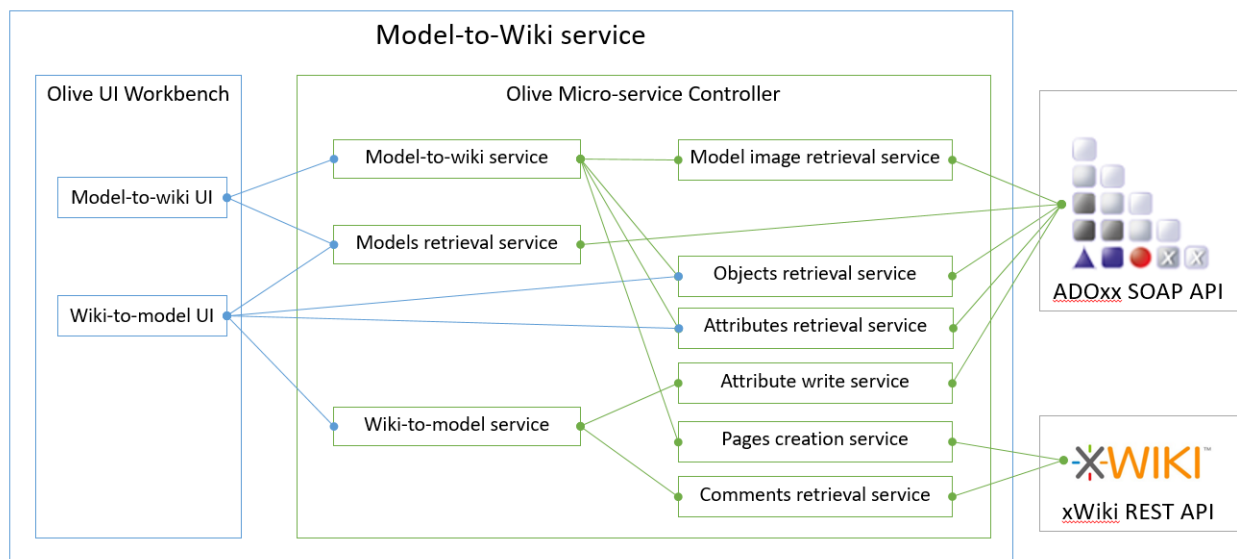


Figure 44 - Model Wiki architecture

The xWiki REST APIs are used in order 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 AdoScript⁵ command `CC "AdoScript" SERVICE start port:80.`

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

- **Model image retrieval service:** microservice that communicate with the SOAP interface of the ADOxx Modelling environment. Require as input a model id and return the Base64 encoded image representation of the model.
- **Models retrieval service:** microservice that communicate with the SOAP interface of the ADOxx Modelling environment and return a list of IDs and names of all available models.
- **Object retrieval service:** microservice that communicate with the SOAP interface of the ADOxx Modelling environment. Require as input a model id and return the list of all the objects inside the model.
- **Attribute retrieval service:** microservice that communicate with the SOAP interface of the ADOxx Modelling environment. Require as input the id of a model or of an object and return the list of all its attributes.

⁵ <https://www.adoxx.org/live/external-coupling-overview>

- **Attribute write service:** microservice that communicate with the SOAP interface of the ADOxx Modelling environment. Require as input the id of a model or object, the attribute name to use and the value to write in the attribute. Return a confirmation code that reflect if the attribute has been written correctly or not.
- **Pages creation service:** microservice that communicate with the xWiki REST interface in order to create a page on xWiki. Require as input a page id, page title and page content using the xWiki syntax.
- **Comment retrieval service:** microservice that communicate with the xWiki REST interface in order to retrieve comments on a specific xWiki page. Require as input the page id of the xWiki page to lookup.
- **Model-to-Wiki service:** microservice that orchestrate 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 objects containing a description of all the objects attributes. Require as input the model ID and use the “model image retrieval”, “object retrieval”, “attribute retrieval” and “page creation” microservices in the background.
- **Wiki-to-Model service:** microservice that orchestrate all the Olive microservices required to import comments of xWiki pages inside the respective model. Require as input the model ID, a model attribute ID and the list of model object IDs with a model object attribute. Use internally the “comment retrieval services” to find all the comments in the model and use the “attribute write services” to store the comments on the model.

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

- **Model-to-wiki UI:** This widget use the “models retrieval service” in order to first obtain the list of all the available models in the ADOxx Modelling Environment and visualize them in a select box, then allow to call the “Model-to-Wiki service” that is responsible to perform all the operations of generating the xWiki pages of the selected model, clicking the “Generate Wiki Pages” button.



Figure 45 - Model-to-Wiki UI Widget

- **Wiki-to-model UI:** This widget use the “models retrieval service” in order to first obtain the list of all the available models in the ADOxx Modelling Environment and visualize them in a select box. As soon as the user select a model it update the select box relative to the model attributes and the table containing all the model objects with a relative select of object

attributes. This done calling the “object retrieval” and the “attribute retrieval” microservices. The model attribute select allow to specify which model attribute to use in order to store comments related to the model (general xWiki page about the model). The object attribute selects allow to specify for every objects which attribute to use in order to store the comments related to the object (specific xWiki sub-page of the model about the object). When the needed attributes have been selected the widget allow to call the “Wiki-to-Model service” that is responsible to perform all the operations of importing the comments for all the xWiki pages related to the selected model in the selected model and objects attributes, clicking the “Import comments to model” button.

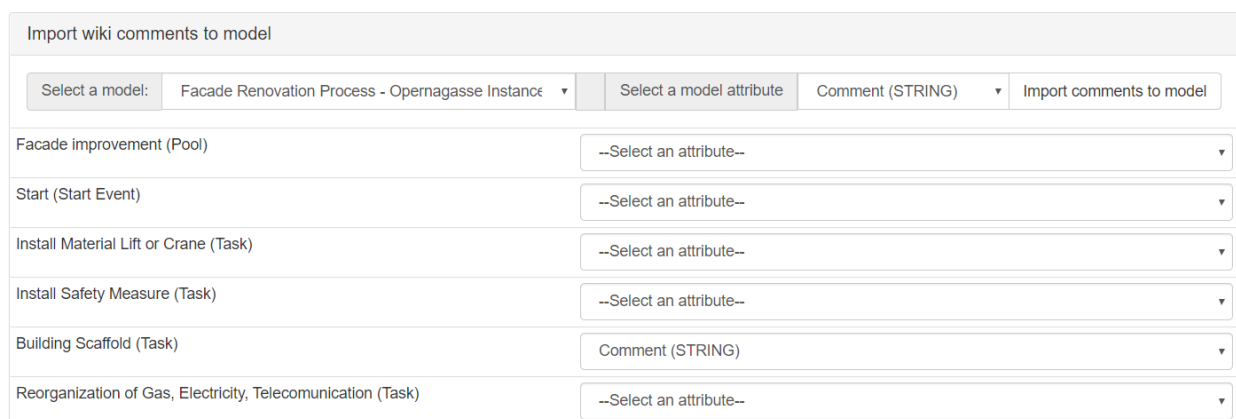


Figure 46 - Wiki-to-Model UI Widget

4.2.2 Model Wiki sample use case

This section containt 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 allow to model processes using the BPMN2 standard notation.

As soon as the SOAP service is started 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.

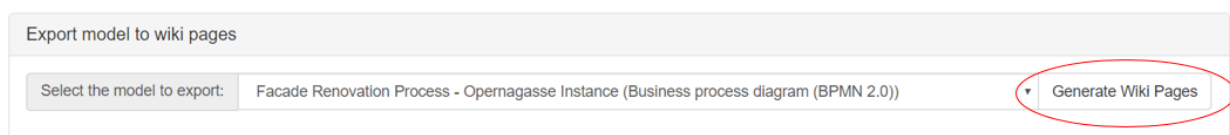


Figure 47 - Facade Renovation Process to Wiki

The generation of the xWiki pages may take some times 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 have now the possibility to comment on the page relative to the model or on every sub-page relative to the tasks. In this case a comment have been added on the Building Scaffold task.

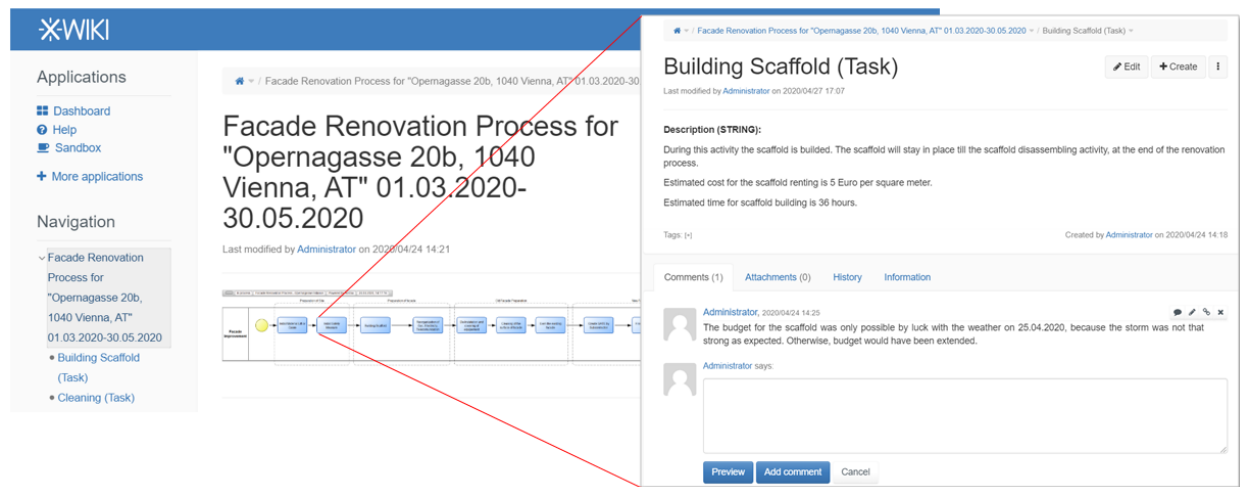


Figure 48 - Wiki pages generated for the Facade Renovation Process

When the modeler decide to check the status of the collaboration on the model, he can import the comments into the original model in order to be processed later. The modeler select so the Facade renovation process and the attributes to use to store the comments for every objects (in particular for this case only for the Building Scaffold task).

Import wiki comments to model

Select a model: Facade Renovation Process - Opemagasse Instance Select a model attribute: Comment (STRING) **Import comments to model**

Facade Improvement (Pool)	--Select an attribute--
Start (Start Event)	--Select an attribute--
Install Material Lift or Crane (Task)	--Select an attribute--
Install Safety Measure (Task)	--Select an attribute--
Building Scaffold (Task)	Comment (STRING)
Reorganization of Gas, Electricity, Telecommunication (Task)	--Select an attribute--

Figure 49 - Wiki-to-Model UI widget for the Facade Renovation process

Clicking the "Import comments to model" button the model will be updated and the comments can be visualized in the attribute "Comment" selected. 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.

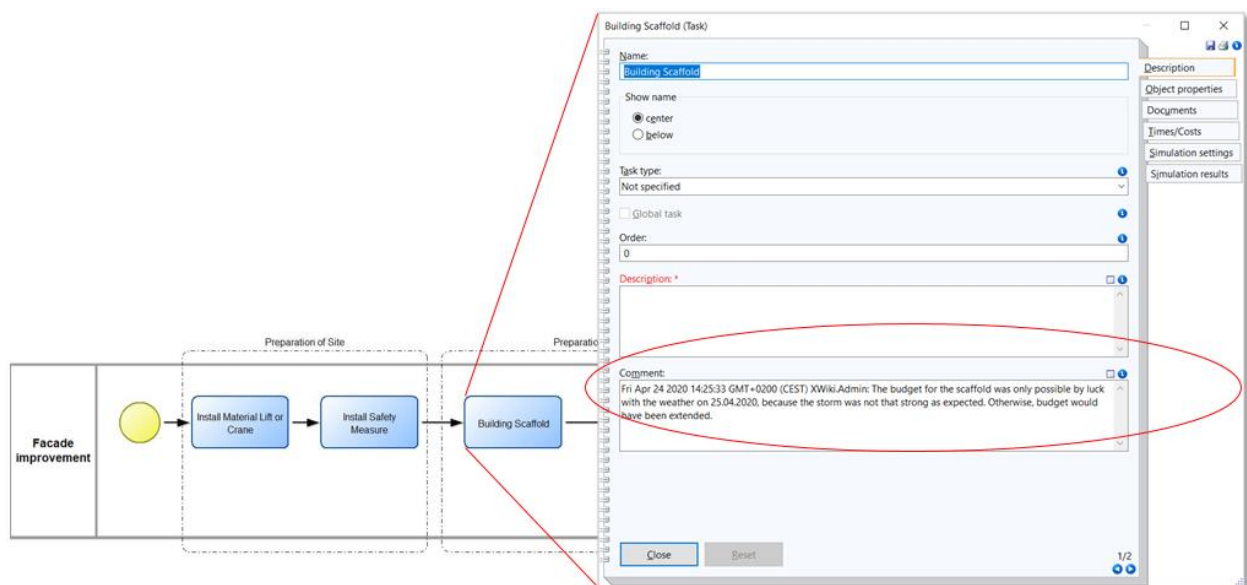


Figure 50 - Comments imported for the Building Scaffold task of the Facade Renovation process

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 D4.4, D4.6 and D4.8 (BIMERR Consortium, 2020 D4.4, D4.6 D4.8), 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 in order to effectively enable the anticipated data exchanges with other BIMERR applications.

From the perspective of the BIMERR Renovation Process Simulation Tool acting as a data provider, its respective developers access the integrated BIF platform interface and create as many data collection jobs as needed in order 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 Renovation Process Simulation 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 on the basis of 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 Renovation Process Simulation Tool, and shall be available for retrieval by other BIMERR applications.

From the perspective of the BIMERR Renovation Process Simulation Tool acting as a data consumer, its respective developers access the integrated BIF platform interface and initiate a new query in order 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 Renovation Process Simulation Tool is authorized to access them. They are informed which are exactly the data that the BIMERR Renovation Process Simulation Tool is able to 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 Renovation Process Simulation Tool in order to automatically retrieve the specific data that were selected in the query from the BIMERR APIs. They can create as many calls to the BIF as needed (through corresponding queries) in order to retrieve all the building-related data according to the requirements and specifications of the BIMERR Renovation Process Simulation Tool.

5.2 INTEGRATION WITH ONTOLOGY

The following section describes the Key Performance Indicator ontology, explaining the main concepts and properties used for its construction, and how they model performance indicators utilized by the PWMA tool. 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). More precisely in this version more specific alignments with the PWMA tools have been taken into account. However, some content might overlap with D4.2 description.

By means of this semantic representation, PWMA can share their indicators to other BIMERR applications interested in analyze or display project management information.

5.2.1 *Ontology Model Explanation*

This ontological model aims to provide the vocabulary to represent indicators used to monitor the advance 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 aspect 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. Figure 51 gives a general overview of the current state of the KPI conceptualization. The ontology is accessible through its permanent URI,⁶ which also provides the HTML documentation of the ontology as well a different ontological serializations.

The model constructed reuses terms from other well known ontologies listed in Table 2. The table indicates the URI of the ontologies, and the prefixes employed for their representation on the diagrams and in this description. For instance, the term `kpi:EconomicKPI` means that the full URI for this concept is: <http://bimerr.iot.linkeddata.es/def/key-performance-indicator#EconomicKPI>.

Prefix	Namespace
kpi	http://bimerr.iot.linkeddata.es/def/key-performance-indicator#
saref	https://saref.etsi.org/core/
s4city	https://w3id.org/def/saref4city#
time	http://www.w3.org/2006/time#

Table 2 - Ontology Prefixes and Namespaces

There exists a variety of KPIs that can be used to measure the performance of a renovation project, the model groups this set of KPI's into five categories: `kpi:ComfortKPI`, `kpi:EnergyKPI`, `kpi:SustainabilityKPI`, `kpi:EconomicKPI`, and `kpi:TimeKPI`. In this new iteration of the model, only the `kpi:TimeKPI` category was included taking into account the D6.2 documentation (BIMERR Consortium, 2020 D6.2), as cost was already modelled as `kpi:EconomicKPI`. The rest of KPI's were created to support energy simulation

⁶ <http://bimerr.iot.linkeddata.es/def/key-performance-indicator#>

tools (RenoDSS) requirements. We also assign an identifier for each KPI generated and a calculation period.

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. Also, if required, a time stamp can be added to the assessed KPI value, to indicate when this evaluation occurred. The KPI metrics can be calculated based on a set of parameters (`kpi:CalculationParameter`), like the time or cost per unit. This is in line with the data access model described in D6.2 (BIMERR Consortium, 2020 D6.2), which gives semantics to the way how KPIs are generated. In the first iteration of the model, the ontology only allowed the assignment of a singular quantity to a KPI value, however this new 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 is added to represent the permitted deviation from those limits.

The PWMA model described in D6.2 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`. The ontology also introduced the relation `kpi:monitorsIndicator`, which connects the goals with a set of KPI values, to verify that these objectives are being satisfied. 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.

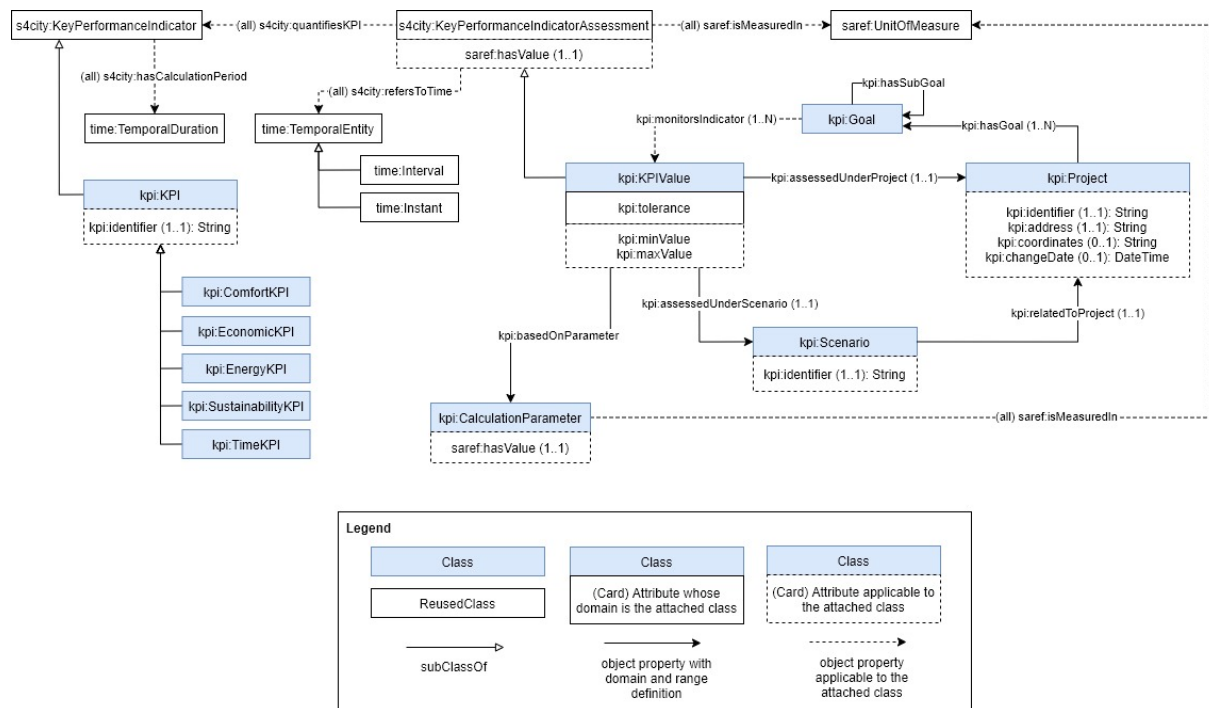


Figure 51 - KPI conceptualization.

5.2.2 Model Instantiation

In order to show how to populate the ontology we take the example provided in Section 6.2 of D6.2 (BIMERR Consortium, 2020 D6.2). 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 actually 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 52 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:EconomicKPI`, 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`.

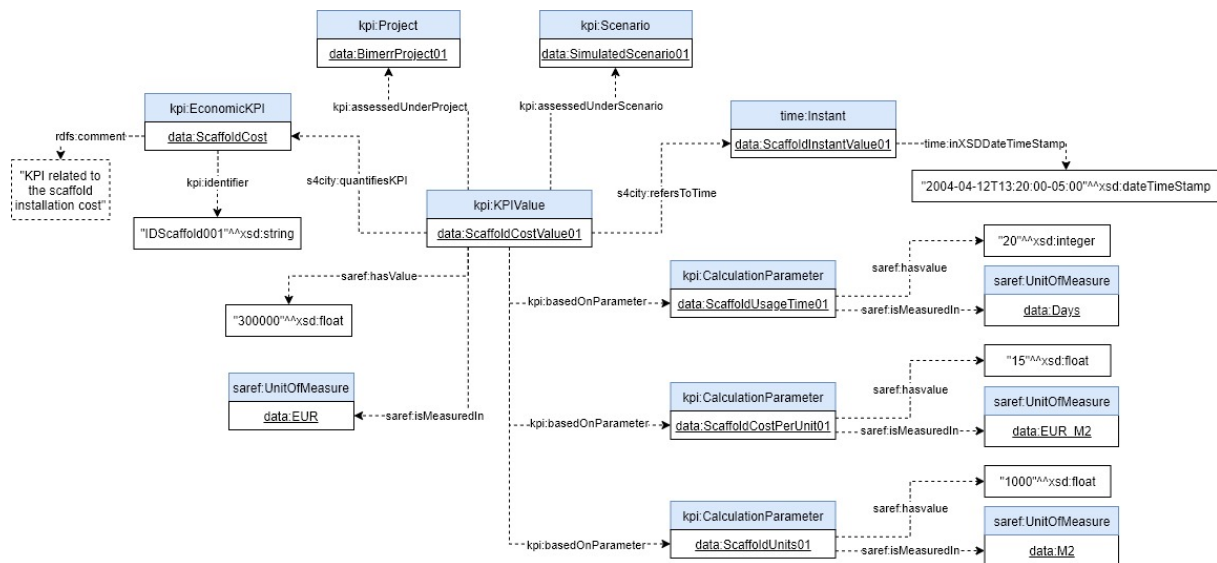


Figure 52 - Example of KPI Ontology Population

5.3 INTEGRATION WITH WORKFLOW

The workflow execution engine will be developed in two iterations. In the first prototype which is going to be presented, the orchestration of the reconstruction process will be made manually, and the tool will be used by the project manager to track the progress of the reconstruction and to record the actual data.

The final version is going to be a semi-automatic version, where external data can affect the reconstruction process. For relevant inputs it will be possible to map an API call (to BIF or external services) which will make the process more automatic. E.g. checking the weather forecast for reconstruction tasks, where weather conditions can affect the quality and result of the work.

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 tasks 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 actually 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)
 - From renovation manager as notification
 - From residents to Renovation Manager for review and acknowledgement
- Health and Safety Notifications

5.4 OPEN INTEGRATION FRAMEWORK

The Microservice Framework Olive⁷ is used as the basis for the development of all the services and functionalities of the BIMERR Design environment, in particular for the one related 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

⁷ <https://www.adoxx.org/live/olive>

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, through the use of plug-ins. Connectors provide the functionalities of your backend services enabling the connection to external systems like databases, CMS, message buses. Olive allow 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 the UI for sections of your web application. Widgets can be generics like visualizing a grid layout or more specifics like visualizing 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 allow to create the whole looks and behavior of your web application, drawing models. This integration allow also to use models as data for microservices. An example is the process simulation microservice that simulate process models taking them directly from the ADOxx modelling platform, or the KPIs evaluator microservice that evaluate KPIs defined in models available in the ADOxx modelling platform.

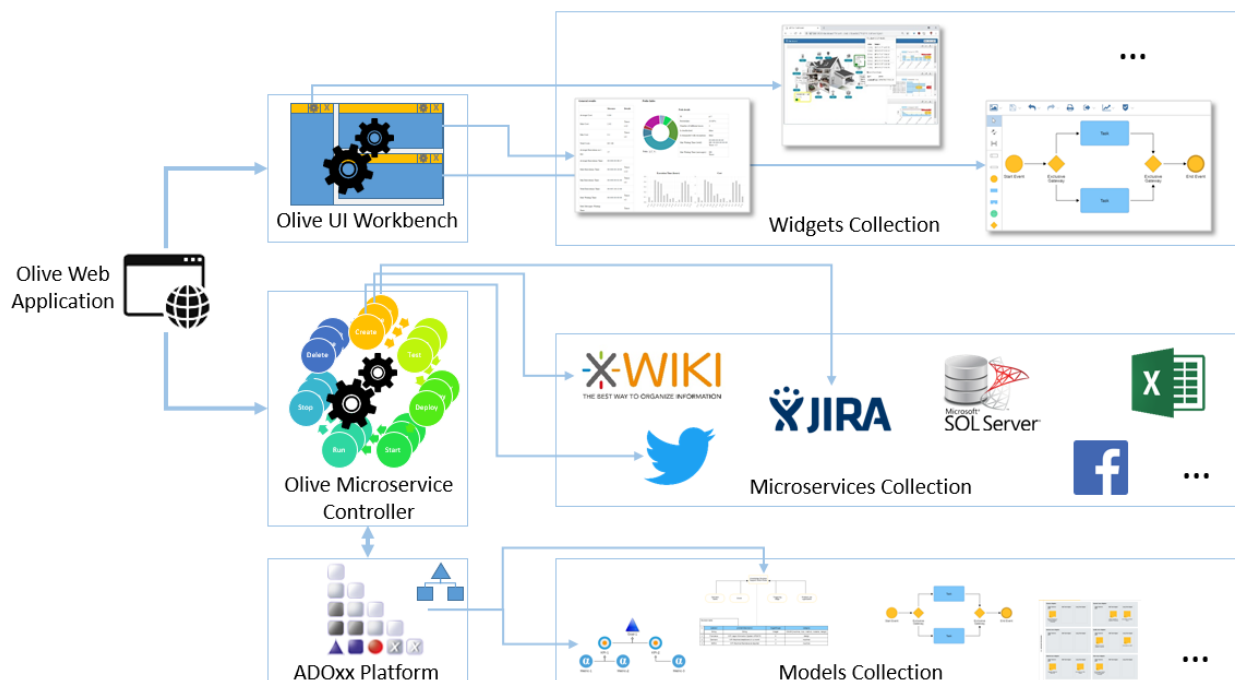


Figure 53 - Olive high level overview

The Olive platform provide so 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 in particular, is used in the PWMA environment of BIMERR as integration framework, allowing to create Microservices that collect and exchange data with the interested BIMERR components. The Olive Microservice Controller is a backend component that allow 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 from external systems (like the database) and the Connector is responsible to connect to such systems to exploit their features.

Olive Microservice Controller allow 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) generate an instance of the REST microservice from the configuration, (2) allow to start the microservice, (3) keep the microservice running in an isolated environment, (4) allow to stop the microservice and (5) allow 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 Felix⁸ 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 together to achieve the business logic task thanks to the Orchestrator component. This component is responsible to combine together existing microservices using the Enterprise Integration Pattern⁹ notation.

⁸ <https://felix.apache.org/>

⁹ <https://www.enterpriseintegrationpatterns.com/>

In order to support a higher level of freedom the orchestrator also allows to use the JavaScript scripting language to combine microservices following a more programmatic approach.

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

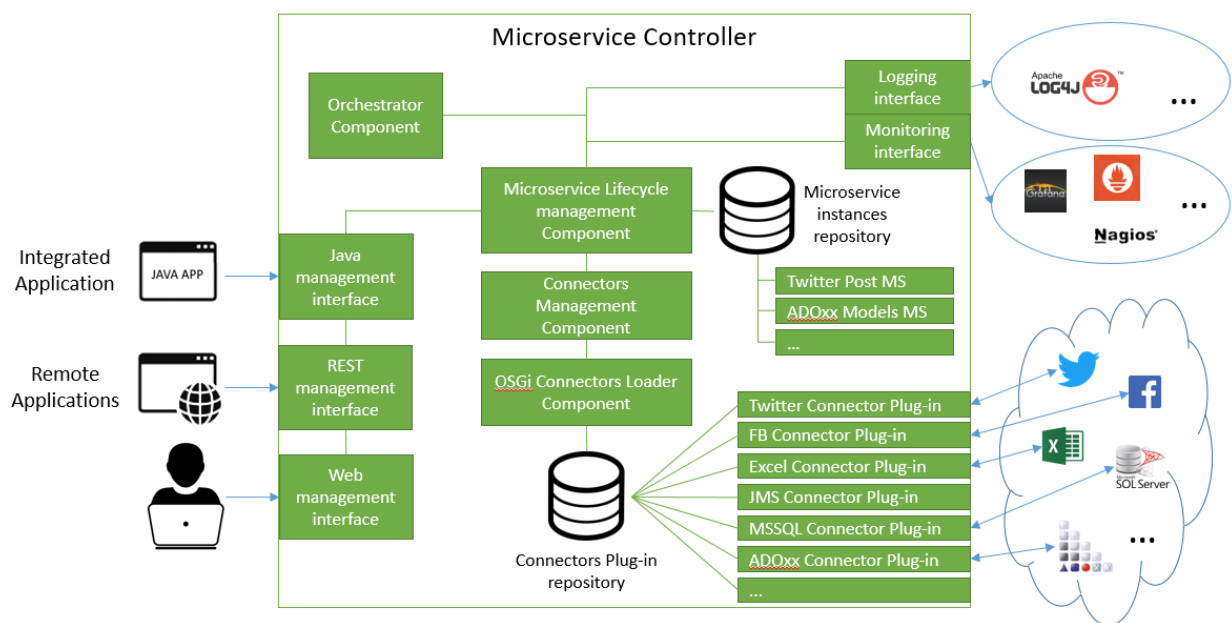


Figure 54 - Olive Microservice Controller Architecture

5.4.1 Microservice definition

A Microservice in Olive is defined through a JSON file that contains the configuration of a specific connector. The Olive platform is able to use this configuration file to create an instance of the connector and expose it through a REST API. A connector is a function provided by the Olive framework, responsible to perform a specific operation. An example is the MySQL connector that allows to query a MySQL database, or a Twitter connector that allows to post and retrieve posts on Twitter.

The connectors are structured in a way that allows to specify which part of the specific operation has to be performed at microservice start, execution and stop. As example, the MySQL

connector will establish the connection with the database during the microservice start phase, will perform the query during the execution phase and will close the connection during the stop phase. This allow to support connection pooling and reuse the same database connection for all the microservice request, increasing the response time. The configuration of the connectors are relevant to the start and execution phases. Considering again the MySQL connector, the configuration of the start phase (that perform the connection to the database) allow to specify the database endpoint address and port, the database name and the access credentials, while the configuration for the execution phase allow to specify only the query to perform.

With this configuration Olive generate a REST service that connect to the specified database and perform the configured query, returning the results in a tabular format as defined by the connector.

Olive integrate out-of-the-box 24 connectors. Custom connectors can be added to the platform as OSGi plugins. This allow to reuse existing OSGi based connectors like all the one provided by the Apache Camel¹⁰ project.

Olive distinguish two kind of connectors, depending on the communication pattern required:

- Synchronous connectors: Types of connectors that provide a functionality on request. Such kind of connectors are used to create REST microservices that once called perform some operations and returns the results to the users. An example is the MySQL connector that is used to create microservices that on user request will perform a query to the database and return the result to the user.
- Asynchronous connectors: Types of connectors that performs operations mainly in background. The Olive platform create REST microservices also from that connectors but they did not return the operations results to the users but will start to process the request in background. Such kind of services may also not require the interaction with the users at all. Due to that fact, the Olive platform allow to attach to such microservices a previously defined synchronous microservice used to process their results. A typical example is a microservice that listen to a message bus. This microservice continuously check in background the presence of new messages and as soon as a message is received will forward it to a microservice responsible to store it in a MySQL database.

¹⁰ <https://camel.apache.org/>

We can summarize that the synchronous connectors are used to create microservices that start to work as soon as the user request them, while asynchronous connectors create microservices that start to work as soon as the microservice is started.

Despite the fact that the main business logic of the microservice is provided by the used Connector, the inputs and the output format of the microservice can be adapted. Those adaptations are also specified defining them in the microservice configuration.

Olive allow also to check the status of the microservice. By default Olive automatically recognize if a microservice is started, stopped and if its connector incurred in an error. In addition the user can define how the output should look like in terms of format and data content. This allow to perform a more deep status check taking into account also the semantic of the output.

Microservices in Olive are organized in structures named Operations. Operations are methods that relate to the same microservice and are the objects that contain the configuration of the connectors, the definition of the inputs and the adaptation of the outputs. Operations are uniquely identified by their name inside a microservice, which is instead identified through a unique ID.

The definition of inputs for a microservice allows to have the microservice configuration partially customizable by the final user through its inputs. In the microservice definition is possible to define which inputs are required by the final users and how this inputs will impact microservice configuration. Due to the fact that the final user interact with the microservice only during its execution (and not during the start and stop phases), the inputs can go to affect only the microservice configuration section relative to the execution phase. As an example we consider a microservice that uses the MySQL connector to perform a query to a database. This connector allows to configure the database endpoint and credentials to use during the starting phase of the microservice, so this means that such information cannot be customized asking inputs to the final user. The configuration of the query to perform is defined instead in the execution phase so we can create a microservice input definition that customize the query. This means that we can request the whole or only a part of the query to the final user as microservice input.

The customization of the microservice configuration through microservice inputs is done using a match and replace mechanisms. In the configuration of the microservice is possible to specify placeholders that will be replaced (previous validation) at execution time with the matching microservice input provided by the final user. So the definition of a microservice input require only a unique name for the input and the name of the placeholder that is used in the configuration for the execution phase.

Using the previous example and imagining to query a database with the following SQL:

```
SELECT name FROM users WHERE mail="Damiano.falcioni@boc-eu.com";
```

If we want to have the mail as microservice input we must first add a placeholder to the query like:

```
SELECT name FROM users WHERE mail="$mail_input_placeholder";
```

And then define an input with name "mail_ID" and placeholder "\$mail_input_placeholder".

At this point when the final user call the microservice REST endpoint he must POST as input a JSON object like this:

```
{  
  
  "mail_ID": {  
  
    "value" : "Damiano.falcioni@boc-eu.com"  
  
  }  
  
}
```

The value provided for the input "mail_ID" will replace the placeholder string "\$mail_input_placeholder", the resulting query will be performed and the result returned to the final user.

At the definition of a microservice it is possible to adapt the output of the configured connector used in the microservice, providing an algorithm in the Javascript programming language. Such algorithm can be used to parse the connector output and convert in the required format or do complex data processing operations. Due to the high level of freedom left to the user, in this case there are some configurable restrictions about the maximum allowed execution time of the algorithm and about the allowed operations (for example operations on file system are denied). Despite such restrictions the following additional data and functions are available:

- output: a variable containing the JSON object returned by the connector.
- input: a variable containing the JSON object provided as input by the final users to the microservice POST endpoint.
- out({...}): a function used to return the final adapted output. This function must be called as the last instruction of your adaptation Javascript. Accept as input a JSON object.
- callMicroservice(microserviceId, operationId, microserviceInputs): a function used to call an existing microservice and obtain its output. It is used to simplify the creation of complex adaptation script, reusing existing microservices and can be used also to chain together microservice functionalities. Require as input the unique id of the microservice, the id of the microservice operation to perform and a JSON object containing the required microservice inputs. Return a JSON object containing the output of the called microservice.

5.4.2 *Microservice instantiation*

As soon as the microservice has been defined, Olive make available a REST endpoint that expose the microservice. At this point the microservice is ready to be used. The REST endpoint require the unique id of the microservice and the name of the microservice operation to execute as query parameters. Microservice inputs instead have to be provided in form of JSON object passed as POST data. The output is returned also as JSON object with a standard structure that encapsulate the microservice output. Olive uniform so the interfaces of all the defined microservices exposing them in a REST endpoint with POST method with fixed query string parameters, POST data and output format.

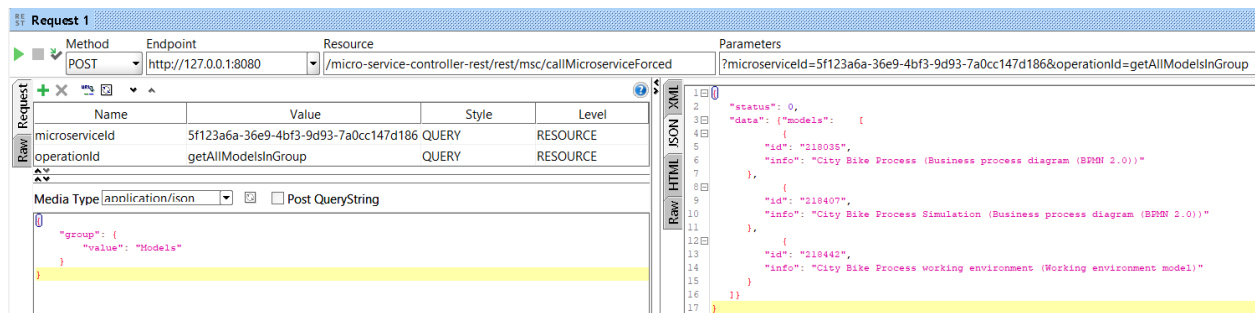


Figure 55 - Olive Microservice REST endpoint sample using SoapUI¹¹

If the microservice is called before being started it will start automatically. The Olive Microservice framework allow also to manually control the starting and stopping phase of a microservice in order to deallocate resources on the machine. These operations are all manageable through the APIs of the Olive Microservice Controller framework as well as through the management interface.

During the starting phase of a microservice a new thread is executed and kept active till its stopping. In the thread the configured connector used by the microservice is initialized. The configuration relative to the starting phase is used for this initialization. As soon as all the initialization operation are completed the microservice is in a started phase and ready to be executed. When the microservice is not used since a long time or on user request it can be stopped. All the stopping operations of the connector are executed and the thread will be terminated.

The thread isolation level is something that is not so common in microservice development due to its insecurity over shared variables. This is true but only if the user write insecure code. The Olive Microservice Controller, following the approach of defining Microservices through connector configuration, don't allow the user to create insecure code. The vulnerable point is in the development of new connectors that the user may require, but in this case is the user responsible for the deployment of a local instance of the Olive Microservice Controller and of its security, while if the new connector is proposed to the community it will follow a deep

¹¹ <https://www.soapui.org/>

review process before being released and available to download in the Olive Microservice Controller package¹².

5.4.3 Microservice Controller User Interface

The main interface is composed of a simple view that allow to select a microservice from a list of all the public microservices or provide the ID of the microservice in case of a private one. As soon as a microservice is selected or provided, the list of all its operation is filled. From this point the interface allow to delete the microservice using the "Delete" button, edit the microservice configuration via the "Edit" button, starting all its operation using the "Start" button and stopping all the started operations using the "Stop" button. When the user select an operation its status is displayed as a green, yellow, red indicator indicating respectively if the microservice operation is running correctly, if is running with some errors (reporting them) or if is stopped. Pressing now the "Start" button the single operation can be started if stopped, while pressing the "Stop" button can be stopped if started. Using the "Test a Call" button is now possible also to test the microservice operation via another interface. At any time is possible to click the "Create New" button that allow to configure a new microservice starting from scratch.

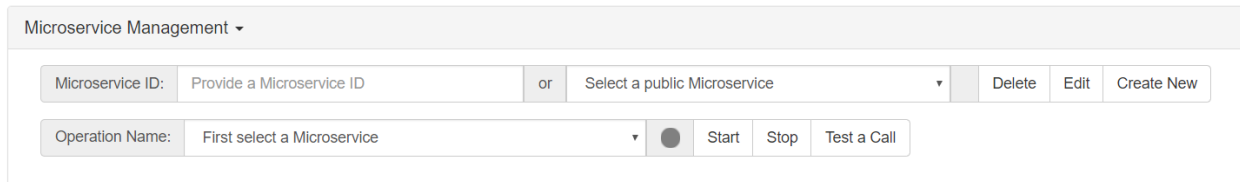


Figure 56 - Olive Microservice Controller Management UI - Main view

The testing interface that appear when the "Test a Call" button is pressed, allow to easily interact with the microservice REST endpoint, providing to the user a view of the required inputs to provide and of the produced output. All the input fields are generated automatically from the microservice definition and clicking the testing button they will be combined in the format accepted by the microservice, sent to it and the output visualized in the relative view.

¹² <https://git.boc-group.eu/adoxx/microservice-controller-rest>

The interface so (1) visualize the full REST endpoint to call and the JSON of input data to post in order to use the microservice, (2) allow to test the microservice with the provided inputs and (3) visualize the output in raw JSON with possibility to test a JavaScript algorithm to process it.

Call Microservice with ID: 1738cb62-cc55-4abf-8560-feafdb83260c Operation: default

Microservice Required Inputs

Append Text world

Test a Call

Microservice ID: 1738cb62-cc55-4abf-8560-feafdb83260c
Microservice Operation: default

POST Endpoint
https://www.adoxx.org/micro-service-controller-rest/rest/msc/callMicroserviceForced?microserviceId=1738cb62-cc55-4abf-8560-feafdb83260c&operationId=default

POST Input Data

```
{
  "Append Text": {
    "value": "world"
  }
}
```

Output description:
Output

```
{
  "dataMIME": "text/plain",
  "dataText": "Hello world",
  "moreInfo": {
    "retrievalTime": "2020-05-19 14:53:18"
  },
  "newField": "test"
}
```

Custom Rendering Algorithm

```
1 /*
2 Javascript algorithm that "return" a DOM object.
3 The algorithm can access the microservice output content
4 using the variable "output"
5 */
```

Service Output Post-Rendering Preview

```
{
  "dataMIME": "text/plain",
  "dataText": "Hello world",
  "moreInfo": {
    "retrievalTime": "2020-05-19 14:53:18"
  },
  "newField": "test"
}
```

Cancel Continue

Figure 57 - Olive Microservice Controller Management UI - Test view

The interface visualized when the "Create New" button is pressed on the main view is exactly the same of the one visualized when the "Edit" button on the main view is pressed. The only difference is the content visualized, that in the first case is empty, while in the second case is pre-filled with the data of the existing microservice definition.

This interface allows to specify first the details of the microservice with its name and description and if it must be visualized in the list of the public microservices available in the main view or if is a private one and accessible only knowing its ID. Then allow to add operations to the microservice. The operations created can be deleted pressing the "Delete" button available on the right side of the operation header. Clicking on the header it will expand allowing to configure all the details of the operation.

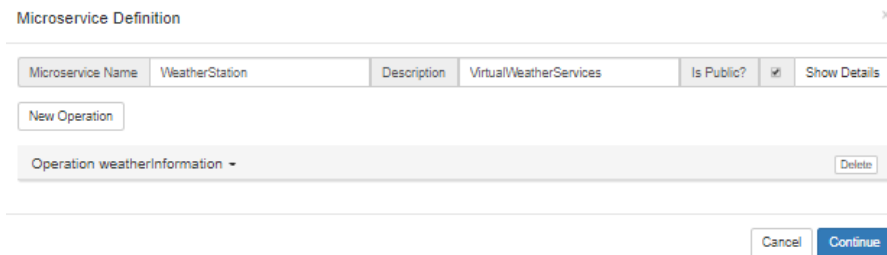


Figure 58 - Olive Microservice Controller Management UI - Edit/New Microservice view collapsed

Expanding each operation section the user can modify the operation id, its visualized named and description and if the operation should start automatically with the Olive platform. If not selected the user must start it manually before being able to use it. If the default checkbox is checked means that this operation is the entry point of the microservice and can be executed without providing the operation id but only the microservice id. Only one operation per microservice can be set as default. After this general configuration the user must select the connector to use for this operation. The select box provide a list of all the connectors available and on its selection change, all the subsequent configuration will change accordingly.

The "Start Configuration" view will visualize all the inputs required by the connector during its initialization. These vary from connectors to connectors and may also be empty. In the case of the REST Connector this section will contain the definition of the REST method to use, the expected MIME content type and optionally a list of additional HTTP headers to setup. The "Call Configuration" view will display all the inputs required by the connector during the execution phase. These also vary from connectors to connectors and may also be empty. In the case of the REST Connector this view will display the configuration of the REST endpoint with optionally its query string and in case of POST or PUT methods optionally the data to send. This section is the only one that can be affected by the microservice input from the user, using the placeholders defined in the microservice inputs section. The "Call Configuration Inputs" view allow to define the microservice inputs to ask to the final users and the placeholder to replace. Every row in this section define a microservice input. New inputs can be added using the "Add New Call Configuration Input" button and removed pressing the "X" button relative to the row. Every input require an unique ID, the name of the placeholder to match for the replacements with the inputs provided by the final users, a description and an example of working input value. This value will be used also during the microservice status check to evaluate the correctness of the microservice output.

The next section to configure is the output description of your microservice that may be the same of the connector output if no adaptation algorithm is defined. In case the output have to be adapted, a JavaScript code can be provided in the “Output Adaptation Algorithm” section. This JavaScript code can access the original output of the connector through the variable `output` and the microservice inputs through the variable `input`. Both of the values are in form of JSON objects. Additionally a function `out ({...})` is available and must be called as last instruction of the algorithm in order to return the new value provided as parameter to the `out` function in JSON object format. In this code also the `callMicroservice(microserviceId, operationId, microserviceInputs)` function can be used in order to call another microservice operation and use its output. The last section allow to provide an algorithm to process the output during the microservice status check. The last instruction in this case must be a Boolean value identifying if the output is correct or not.

Microservice Definition

Microservice Name	WeatherStation	Description	VirtualWeatherServices	Is Public?	<input checked="" type="checkbox"/>	Show Details
-------------------	----------------	-------------	------------------------	------------	-------------------------------------	--------------

New Operation

Operation weatherInformation Delete

Operation ID	weatherInformation	Name	weatherInformation	Description	Retrieve base weather information !	Is Default?	<input checked="" type="checkbox"/>	Autostart?	<input checked="" type="checkbox"/>
--------------	--------------------	------	--------------------	-------------	-------------------------------------	-------------	-------------------------------------	------------	-------------------------------------

Connector REST Connector Get data from a REST service

Start configuration

Method GET

Content Type application/json

Additional Headers

Call configuration

Endpoint `http://api.openweathermap.org/data/2.5/weather?q=%CITY%&APPID=ee97a2fe49de48925d0a6b78dc95d838&units=metric`

QueryString

POST Data

Call Configuration Inputs

Add new call configuration input

Input ID:	cityname	Matching Name:	%CITY%	Description:	City name	Working Sample:	Vienna	X
-----------	----------	----------------	--------	--------------	-----------	-----------------	--------	----------------

Connector Output Description

A JSON object in the following format:

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

Output Description

Output Adaptation Algorithm

```
1 out({
2   temperature: output.data3son.main.temp
3 });
```

Status Check Algorithm

```
1
```

Cancel Continue

Figure 59 - Olive Microservice Controller Management UI - Edit/New Microservice view expanded

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 following instruction 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 provided 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 freely accessible using the BOC cloud deployed instance at https://bimerr.boc-group.eu/ADONISNP10_0/ after registration.

Username and password required to enter the platform are created for single users on request to the e-mail address: faq@adoxx.org.

6.1.3 Fast deployment of the cloud Renovation process design tool

The fast deployment package contain 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> contain 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. After the extraction then is sufficient to perform the following operations:

1. Run 1-start.bat to start the design environment
2. Run 2-ADONISNP10_0 localhost to open the web portal
3. Run 3-stop.bat to stop the design environment

In the package an initialization script named 0-init_db.bat is provided that must be executed on very first run in order to initialize the SQL database.

6.2 WORKFLOW EXECUTION TOOLS

The workflow execution engine is available at <https://i3d.econtent.lu/bimerr/>.

If you are a new user, you need to register via "Ask for registration". In project name, do not forget to enter "bimerr".

Client registration form

User name

Project name(only if apply for known project)

Login (email)

Password

New password

Figure 60 - Workflow execution registration form

6.3 MONITORING AND EVALUATION TOOLS

In this section the deployment instruction of the monitoring and evaluation tools are provided. First the KPIs dashboard deployment instructions are described, then the building and deployment procedure of the simulation tool. The whole set of tools can be tested using the fast deployment package provided at <https://git.boc-group.eu/bimerr/fast-deploy-package/> that include all the dependencies and applications in one standalone package.

6.3.1 Fast deployment package

The GIT project at <https://git.boc-group.eu/bimerr/fast-deploy-package/> is used to store as releases all the created fast deployments packages based on Olive relative to the BIMERR EU Project.

Requirements:

1. ADOxx installed: <https://www.adoxx.org/live/download-guided>
2. KPI library and models imported: available inside the folder MODELS/KPI (check https://www.adoxx.org/live/import_new_application_library and https://www.adoxx.org/live/import_models_adl for detailed instructions)
3. BPMN library and models imported: available inside the folder MODELS/BPMN (check https://www.adoxx.org/live/import_new_application_library and https://www.adoxx.org/live/import_models_adl for detailed instructions)

Instructions to starting the portal:

1. Execute the "1-Start Tomcat.bat" file
2. Execute the "2-Start ADOxx SOAP Server.bat" using the created username and password for the ADOxx Modelling toolkit

3. Execute the "3-Start xWiki.bat" file
4. Open the link "2-Open Olive for BIMERR Portal"
5. Read the documentation in the doc folder to start using the simulator
6. Terminate the Olive execution closing the command console opened in points 1, 3 (Ctrl+c).

Inside the folder MODELS you can find, divided by categories, all the required models to run the demo. In particular inside the SIMULATION folder is provided the BPMN export of the renovation process as well as different Excel inputs for the simulation. In the KPI folder there will be the ADOxx library with the export of KPI models. In the DATA MINING there are samples of the workflow execution log to provide to Celonis for analysis. Finally in the BPMN folder you can find the library and the model to import in the ADOxx Community edition.

6.4 REFLECTION AND INNOVATION TOOLS

In this section the deployment instruction of the reflection and innovation tools are provided. Also in this case the whole set of tools can be tested using the fast deployment package provided at <https://git.boc-group.eu/bimerr/fast-deploy-package/> and described in section 6.3.1.

6.4.1 *Process Mining with Celonis*

Celonis is a process mining platform that allow to analyze log files and construct custom analytical dashboards. Its free version Celonis Snap can be used previous registration to their portal and the whole platform is available as a cloud application. In order to use the service perform a registration at <https://www.celonis.com/snap-signup/>. After successfully registered you will have access to your own Celonis Snap analytics space. Details of your specific space URL are sent by e-mail at the end of the registration process.

6.4.2 *Collaboration with Model Wiki*

The source and deployment instruction of the model wiki scenario are available in the GIT space <https://git.boc-group.eu/olive/model2wiki>. In order to run the application apply the followings steps.

Olive Microservice Controller configuration:

1. Deploy locally on port 8080 the last version of the microservice-controller from this url: <https://git.boc-group.eu/adoxx/microservice-controller-rest/-/tags>
2. Copy the microservice JSON configuration file from the CONFIG folder to your microservice-controller service repository folder.
3. If everything is correct, refreshing your microservice controller dashboard you will see a new microservice named "Wiki Scenario Services".

xWiki configuration:

1. Download the xWiki from <https://www.xwiki.org/xwiki/bin/view/Download> selecting the "Standard Flavor Preinstalled" for an easy deploy
2. Deploy xWiki on port 8081: Extract the downloaded zip file in a folder and open that folder in command console cmd.exe, then execute "start_xwiki.bat 8081" Note: In order to add comments to a page you must be logged in. The default username is "Admin" and password "admin"

ADOxx configuration:

1. Download and install the ADOxx toolkit from <https://www.adoxx.org/live/download-guided>
2. Create a modelling library (follow the tutorials at <https://www.adoxx.org/live/getstarted-helloworld>) or import an existing one (https://www.adoxx.org/live/import_new_application_library)
3. Execute the ADOxxServerStart.bat file in the CONFIG folder in order to run the ADOxx Modelling toolkit with enabled SOAP server on localhost port 80.

6.5 OPEN INTEGRATION FRAMEWORK OLIVE

This section will contain the instruction 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 in order to execute them. Three different deployment modalities has been provided:

- **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.

- **Fast deployment setup:** 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.

In the following each modality is explained.

6.5.1 Source code compilation

The Olive Microservice Framework is composed of two Java projects based on Maven. The core project is represented by the Microservice Controller that expose all the framework functionalities as Java APIs. Dependent on this project there is the Microservice Controller REST that expose all the functionalities through REST protocol.

The following software requirements are needed:

- Java 8
- Maven
- Git

In order to compile the framework you must build the dependencies:

- Obtain the Microservice Controller source code:
Download the zip from <https://git.boc-group.eu/adoxx/microservice-controller> or git clone <https://git.boc-group.eu/adoxx/microservice-controller.git> in a command console.
- Open the source folder in a command console.
- Build executing mvn install in the console.

Now you can build the war file to deploy in an application server:

1. Obtain the Microservice Controller REST interface source code:
Download the zip from <https://git.boc-group.eu/adoxx/microservice-controller-rest> or git clone <https://git.boc-group.eu/adoxx/microservice-controller-rest.git> in a command console.
2. Open the source folder in a command console.
3. Build executing mvn install in the console.

The microservice-controller-rest.war file is now ready to be deployed in an application server of your choice.

6.5.2 Manual setup

In order to manually setup the Olive Microservice Controller you have to do the following steps:

1. Download the last version of the compiled microservice-controller-rest.war file from <https://git.boc-group.eu/adoxx/microservice-controller-rest/-/tags>
2. Deploy the WAR file in a Tomcat webapp folder.
3. Update the paths in the configuration file config.json under micro-service-controller-rest\src\main\resources\org\adoxx\microservice\api\rest

The required configuration keys are:

- microservicesDefinitionFolder: containing the folder where to store the microservices Json configuration files.
- uploadFolder: containing the folder where to store the uploaded files.
- logFileName : containing the path of the log file.
- autostartEnabled: when true all the microservice that have an autostart property enabled will start at runtime.

Once deployed the management web interface will be available at http://your_domain/micro-service-controller-rest/.

6.5.3 Fast deployment package setup

The fast deployment installation package is a fast way to deploy the Olive Microservice Controller. It is provided in the form of a standalone zip file containing all the components and dependencies preconfigured to work out-of-the-box on Window OS. This is useful mainly for testing and development purpose and should not be used on production environment.

In order to proceed perform the following steps:

1. Download the last version of the Olive Microservice Controller fast deployment installation package from this URL: <https://git.boc-group.eu/olive/microservice-controller-fast-deployment-package/-/tags> <https://git.boc-group.eu/olive/model2wiki/-/tags>.
2. Extract the content of the zip archive in a folder.
3. Read the README file in order to have more information on how to proceed.
4. Execute the "1-Start Microservice Controller.bat" file.
5. Open the link "2-Open Olive Microservice Controller Management UI".
6. Terminate the execution closing every command console opened in the points 4 (Ctrl+c).

6.5.4 Docker setup

The “dockerized” version of the Olive Microservice Controller allow to simplify the deployment on production servers and Linux environment that do not support the fast deployment package. The repository <https://git.boc-group.eu/adoxx/microservice-controller-docker> contain the Docker file that generate a Docker image ready to be executed containing the last version of the Olive Microservice Controller.

In order to start a Docker container you have to perform the following steps:

1. Download the msc-Dockerfile from the repository <https://git.boc-group.eu/adoxx/microservice-controller-docker>
2. Build the image using the command:
`sudo docker build -f msc-Dockerfile .`
3. Run the container using a folder as volume for persistence:
`mkdir ./msc-data`
`sudo docker run -it -p 8080:8080 -v ${PWD}/msc-data:/opt/msc-data/ _id_from_build_`

The management web interface of the Microservice Controller will be now available at <http://127.0.0.1:8080/micro-service-controller-rest/>.

7. CONCLUSION AND OUTLOOK

This deliverable introduces the first set of functional capabilities of the ecosystem of renovation process management. The technology that is described in this deliverable corresponds to the approach that is described in D6.2 “Adaptive Renovation Process & Workflow Models 1”. This approach is updated in a second iteration and hence accordingly this deliverable and the corresponding tool set will be updated.

This document explains the renovation process management tool ecosystem by:

- First, using the meta-modelling platform ADOxx, configure 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.
- Second, the Microservice Framework Olive is used to provide a set of functional capabilities for the models. The framework is provided as a download package at ADOxx.org following the link to Olive download.
- Third, 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 select the latest BIMERR download package to get the latest version. The package indicating D6.4 corresponds with this document; a later version of the package indicates updates according this document.
- Forth, in case of accessing and integration 3rd party application like the workflow-engine, 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.

The update of this tool set will include:

- Integration in form of additional microservices to other tools in BIMERR to integrate the BIMERR ecosystem for renovation process management.
- Introducing additional modelling assistant services to simplify the modelling and integrate data that are already available into model, like using parts of the project plan to partly create the process model.
- Simplify the simulation interface used to provide simulation parameters-

- Introduce services for better reflection and improvement of the renovation process management such as the wiki pages for co-creative reflection. A tighter combination with legacy systems or mobile apps has the potential to establish a learning environment.
- The execution using workflow engines may be enriched with the use of mobile apps to not only integrate a workflow engine on the process management but to integrate relevant applications for a user friendly and flexible renovation process management ecosystem.

BIBLIOGRAPHY

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

BIMERR Consortium (2020). D4.4 BIMERR Building Semantic Modelling tool 1

BIMERR Consortium (2020). D4.6 BIMERR Information Collection & Enrichment Tool 1

BIMERR Consortium (2020). D4.8 Integrated Interoperability Framework 1

BIMERR Consortium (2020). D6.2 – Adaptive Renovation Process & Workflow Models 1.

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