Virtual Learning Factory Toolkit

Output 1

Report

Project Reference: 2018-1-EE01-KA203-047094

Co-funded by the
Erasmus+ Programme
of the European Union

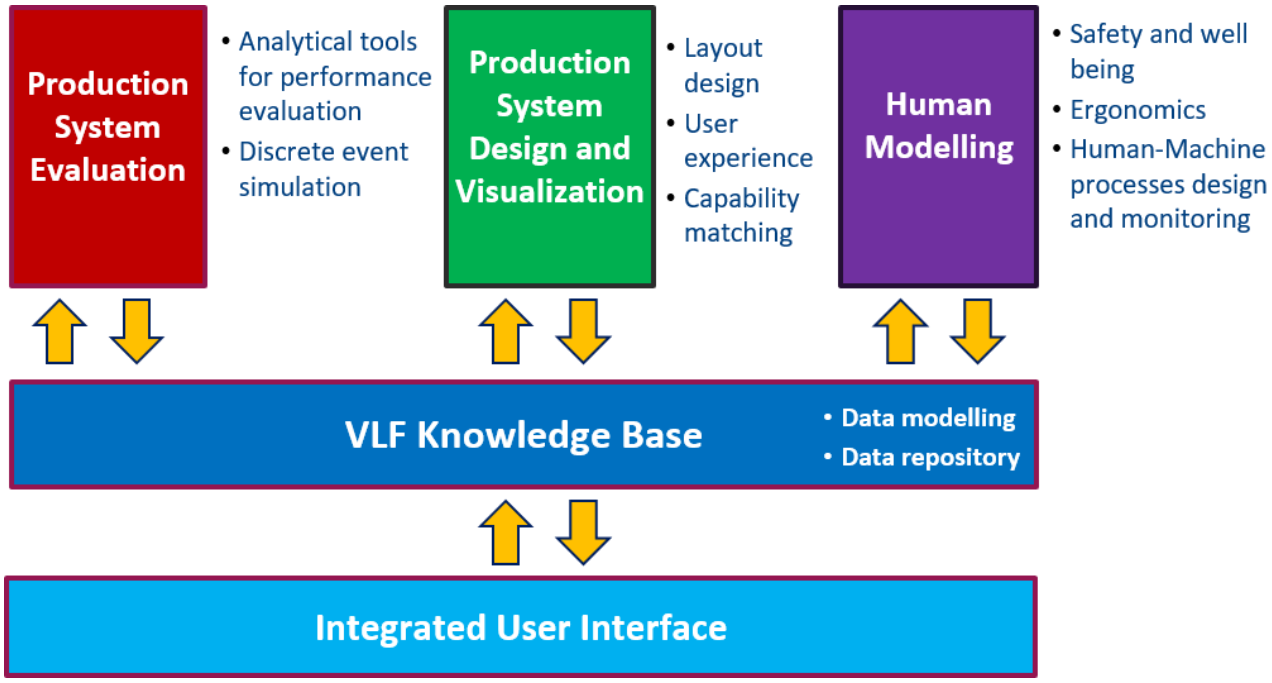
Table of Contents

Co-funded by the
Erasmus+ Programme
of the European Union

1. Virtual Learning Factory Toolkit Framework

The Virtual Learning Factory Toolkit (VLFT) is a set of existing digital tools to support advanced engineering education in manufacturing. The aim of the VLFT is to bring back to the engineering students the results of research activities in the field of digital manufacturing related to the modeling and analysis of a manufacturing system, virtual and augmented reality, as well as the role of the human workers in a factory.

VLFT grounds on a common knowledge base that consists of data model and data repository relying on semantic web technology. In principle, any digital tool can be integrated in the VLFT framework if it is possible to access and modify its internal data structures (e.g. via exchange files or an API), thus creating data flows with the knowledge base, possibly in an automated way.



VLFT Framework

The Virtual Learning Factory Toolkit (VLFT) documentation consists of the following main sections:

- VLF Knowledge Base
- VLF Tools and Libraries

2. VLF Knowledge Base

Co-funded by the
Erasmus+ Programme
of the European Union

The VLF Knowledge Base of VLFT is grounding on a standard, extensible, and common data model for the representation of factory objects related to production systems, resources, processes and products. This factory data model is developed as an OWL ontology, since this language provides a way to generate a flexible data model integrating different knowledge domains, enabling knowledge sharing between several applications and a fluent flow of data between different entities. In particular, the VLF Knowledge Base exploits already existing technical standards (e.g. Industry Foundation Classes, UML Statechart, W3C SSN/SOSA) and research results by CNR-STIIMA and Politecnico di Milano-Mechanical Engineering Department.

The VLF Knowledge Base will be continuously extended by adding concepts to the data model that is used to instantiate models representing academic use cases and industrial case studies. The results of the modelling activities will be stored in the knowledge base to be used for future teaching and training purposes. The knowledge base can be implemented adopting different technologies ranging from file-based solutions to relational databases and to native triple stores.

2.1 Factory Data Model

A suitable factory data model must be able to cover and integrate heterogeneous knowledge domains, while guaranteeing extensibility. Herein, a modular ontology-based Factory Data Model [7] is adopted to formalize the information that is in particular relevant to the design and management of factories and manufacturing systems. The ontology reuse, even if often applied in a limited fashion, represents a key best practice that was followed in this work. Indeed, already existing ontologies have been reused, integrated, and extended.

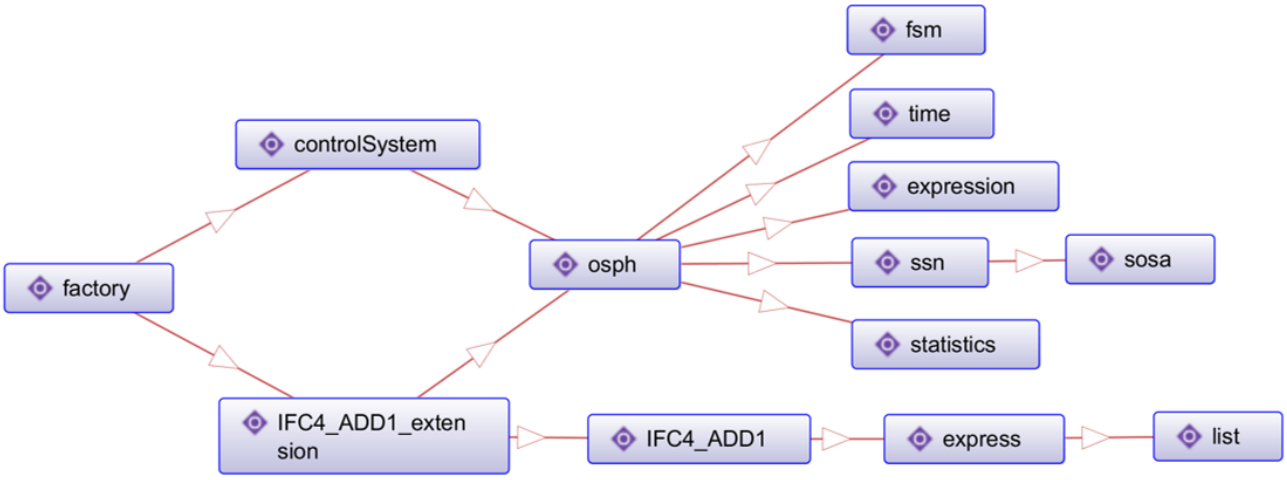
OWL Ontology - Modular Architecture

The modular data model architecture adopted by VLFT consists of OWL ontology modules. An OWL ontology can be serialized in various formats (e.g. Turtle, RDF/XML, OWL/XML) and can be opened with different ontology editors (e.g. Protégé).

The ontology architecture consists of he following modules:

- **list**, an ontology defining the set of entities used to describe the OWL list pattern [1]
- **express**, ontology mapping the concepts of EXPRESS language (International Organization for Standardization, 2004) to OWL [1]
- **IFC4_ADD1**, the ifcOWL ontology that is converted from the IFC standard defined in EXPRESS language [2]
- **time**, an ontology defining concepts related to time [3]
- **fsm**, an ontology defining the concepts required for modelling finite state machines [4]
- **sosa**, the Sensor, Observation, Sample, and Actuator (SOSA) Core Ontology [5]
- **ssn**, the Semantic Sensor Network Ontology [5]
- **statistics**, an ontology that defines probability distributions and descriptive statistics concepts [6]
- **expression**, an ontology modelling algebraic and logical expressions [6]
- **osph**, an ontology modelling Object States and Performance History, while integrating the ontology modules fsm, statistics, ssn, sosa, expression [6]
- **IFC4_ADD1_extension**, an ontology module integrating osph and IFC_ADD1_rules modules, while adding general purpose extensions to IFC_ADD1 [7]

- **factory**, a specialization of IFC_ADD1 with definitions related to production processes, part types, manufacturing systems and machine tools [7]



Ontology modules in the Factory Data Model

Modules and Prefixes

The following table reports the list of **prefixes that have been defined with reference to vocabularies and ontology modules.** All modules are available online at the same address, except *fsm* that can be found at http://people.cs.aau.dk/~dolog/fsm/fsm.owl.

The whole set of ontology modules can be found also in the "repository" folder of the OntoGui installation.

Each ontology module contains the definition of OWL classes and properties and some examples are given in the next subsection.

Prefix	Value
xsd	http://www.w3.org/2001/XMLSchema#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
owl	http://www.w3.org/2002/07/owl#

expr	https://w3id.org/express#
list	https://w3id.org/list#
fsm	http://www.learninglab.de/~dolog/fsm/fsm.owl#
stat	http://www.ontoeng.com/statistics#
time	http://www.w3.org/2006/time#
ex	http://www.ontoeng.com/expression#
sosa	http://www.w3.org/ns/sosa/
ssn	http://www.w3.org/ns/ssn/
osph	http://www.ontoeng.com/osph#
ifc	http://ifcowl.openbimstandards.org/IFC4_ADD1#
ifcext	http://www.ontoeng.com/IFC4_ADD1_extension#
fa	http://www.ontoeng.com/factory#

Object Typing pattern

Ontology modules that are based on the IFC standard (i.e. ifcOWL) take advantage of pattern of object typing thanks to the definition of two main classes: **ifc:IfcObject** and **ifc:IfcTypeObject**.

Class ifc:IfcTypeObject (and its subclasses, e.g. fa:MachineToolType) can be used to define “specific information about a type, being common to all occurrences of this type”. Instances of ifc:IfcTypeObject are represented by sets of properties which apply to all the associated instances of ifc:IfcObject and its subclasses. Object types can be however directly instantiated without being assigned to instances of ifc:IfcObject [8][9]. This approach is useful whenever the user needs to be generic or non-committal while defining the resources needed or used to execute a process. The link between instances of ifc:IfcObject and ifc:IfcTypeObject is realized via properties ifcext:typesObject and ifcext:isDefinedByType.

In practical applications, subclasses of ifc:IfcTypeObject are instatiated to populate a catalog of models/types/templates (e.g. machine models in the catalog of a machine tool builder), whereas subclasses of ifc:IfcObject are instatiated to define the occurrences composing a specific factory configuration.

References

1. Pauwels P, Terkaj W (2016) EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. Automation in Construction, 63:100–133. ISSN: 0926-5805. doi:10.1016/j.autcon.2015.12.003
2. Pauwels P, Krijnen T, Terkaj W, Beetz J (2017) Enhancing the ifcOWL ontology with an alternative representation for geometric data. Automation in Construction, 80:77-94. ISSN: 0926-5805. doi:10.1016/j.autcon.2017.03.001
3. Time Ontology in OWL, https://www.w3.org/TR/owl-time/
4. Dolog P (2004) Model-Driven Navigation Design for Semantic Web Applications with the UML-Guide. In Proc. ICWE, pages 75–86, 2004.
5. Semantic Sensor Network Ontology, https://www.w3.org/TR/vocab-ssn/
6. Terkaj W, Schneider GF, Pauwels P (2017) Reusing Domain Ontologies in Linked Building Data: the Case of Building Automation and Control. Proceedings of the 8th Workshop Formal Ontologies Meet Industry, Joint Ontology Workshops 2017, CEUR Workshop Proceedings, vol. 2050.
7. Terkaj W, Gaboardi P, Trevisan C, Tolio T, Urgo M (2019) A digital factory platform for the design of roll shop plants. CIRP Journal of Manufacturing Science and Technology, 26:88-93. ISSN: 1755-5817. doi:10.1016/j.cirpj.2019.04.007
8. Liebich, T., Adachi, Y., Forester, J., Hyvarinen, J., Richter, S., Chipman, T., Weise, M. & Wix, J. (2013). Industry foundation classes IFC4 official release. https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/

9. Borgo S, Sanfilippo EM, Sojic A, Terkaj W (2015) Ontological Analysis and Engineering Standards: An Initial Study of IFC. In: Ebrahimipour V, Yacout S (eds) Ontology Modeling in Physical Asset Integrity Management. Springer: 17-43. https://doi.org/10.1007/978-3-319-15326-1_2

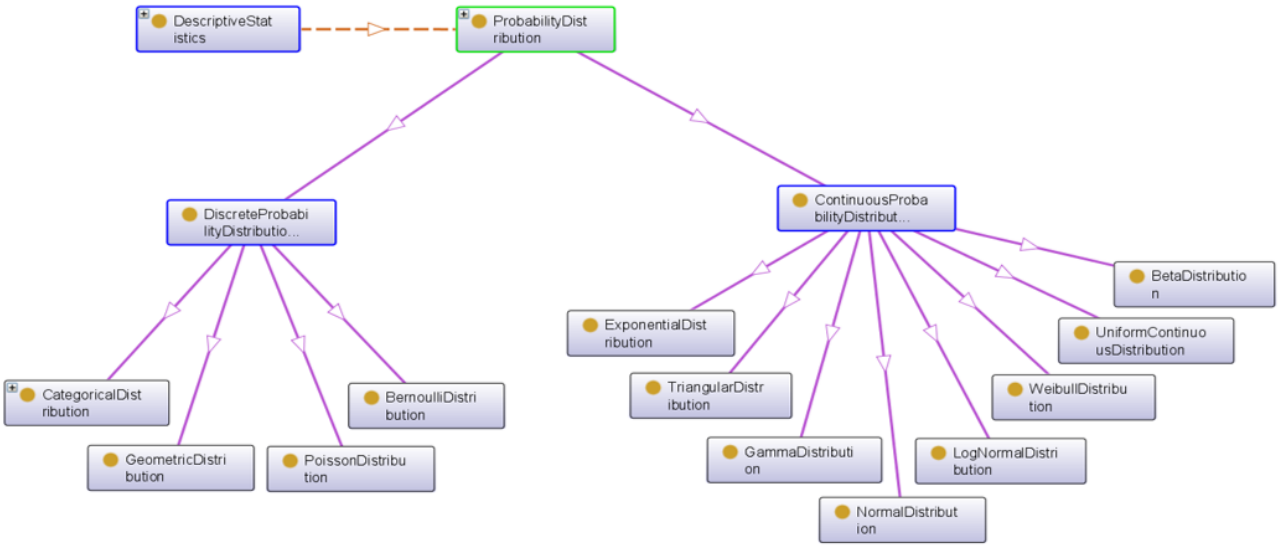
OWL Classes

This section presents more details of the Factory Data Model in terms of OWL classes that are useful to instantiate factory models while going through the ontology modules.

---

statistics

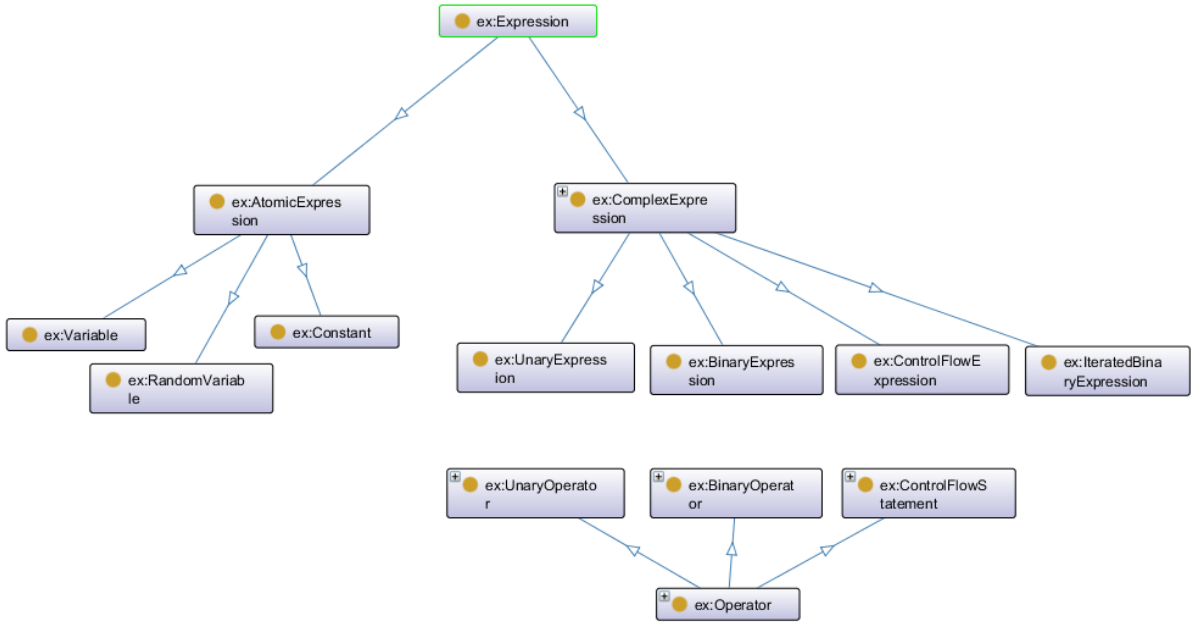
The **statistics module** represents basic probability distributions via semantic characterization of key parameters**.** Representation of descriptive statistics.



---

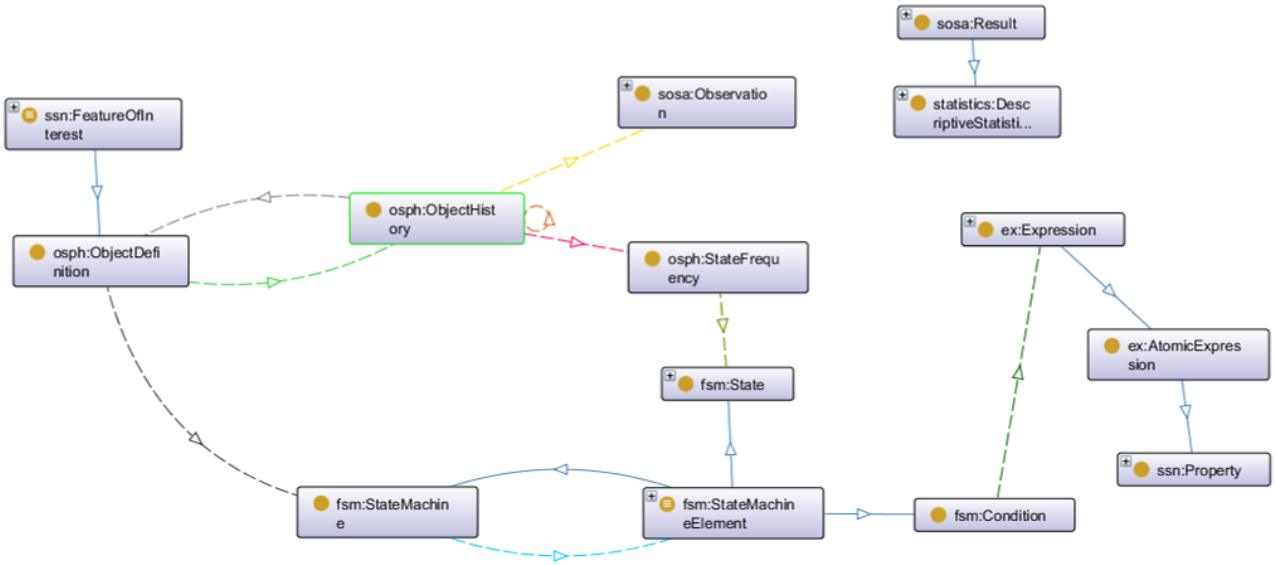
expression

The **expression module** formalizes Algebraic and Logical expressions. An expression can be decomposed in atomic, unary, binary, iterated binary, control flow expressions.



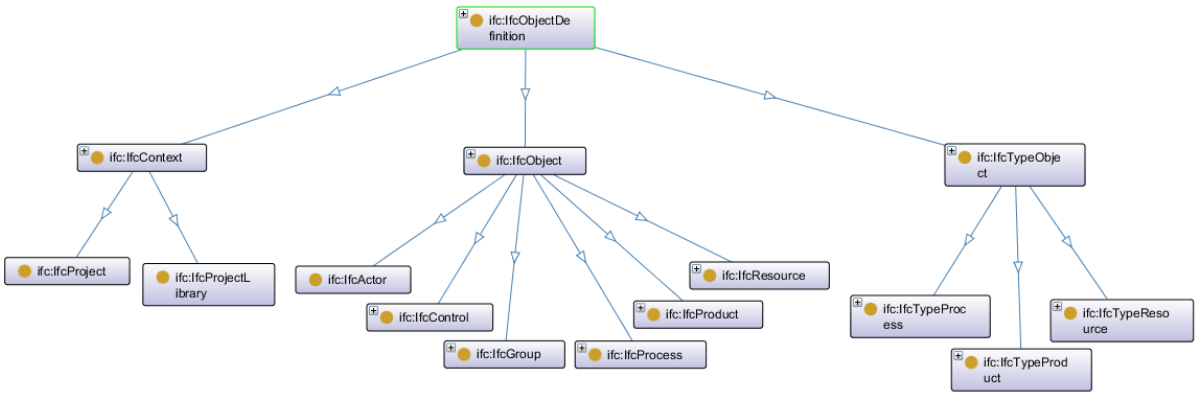
osph

The **osph module** models Object States and Performance History, while integrating the ontology modules fsm, statistics, ssn, sosa, expression.



IFC4_ADD1 and IFC4_ADD1_extension

The **ifcOWL ontology** consists of three main classes: ifc:IfcContext, ifc:IfcObject, ifc:IfcTypeObject.



The following table lists some subclasses of ifc:IfctObject that can be exploited to model a factory. If available, a subclass of ifc:IfcTypeObject is paired with the corresponding sublcass of ifc:IfctObject, as it can be exploited according to the object typing pattern.

ifc: prefix stands for http://ifcowl.openbimstandards.org/IFC4_ADD1#

Subclass of ifc:IfcObject	Description	Subclass of ifc:IfctTypeObject
ifc:IfcBuilding	Building	
ifc:IfcDoor	Door	ifc:IfcDoorType
ifc:IfcColumn	Column of a building	ifc:IfcColumnType
ifc:IfcRoof	Roof of a building	ifc:IfcRoofType
ifc:IfcWall	Wall of a building	ifc:IfcWallType
ifc:IfcWindow	Window of a building	ifc:IfcWindowType
ifc:IfcSensor	Sensor subclass also of sosa:Sensor	ifc:IfcSensorType
ifc:IfcActuator	Actuator subclass also of sosa:Actuator	ifc:IfcActuatorType

ifc:IfcElementAssembly	Complex element assembly	ifc:IfcElementAssemblyType
ifc:IfcIfcTransportElement	Generalization of all transport related objects	ifc:IfcTransportElementType
ifc:IfcTask	Unit of work	ifc:IfcTaskType

factory

The **factory module** specializes IFC_ADD1 and controlSystem modules with definitions related to production processes, part types, manufacturing systems and machine tools.

The following tables list some subclasses of ifc:IfctObject that can be exploited to model a factory. If available, a subclass of ifc:IfcTypeObject is paired with the corresponding sublcass of ifc:IfctObject, as it can be exploited according to the object typing pattern.

fa: prefix stands for http://www.ontoeng.com/factory#

Subclass of ifc:IfcProduct	Description	Subclass of ifc:IfctTypeProduct
fa:Artifact	Part that is the result of a production activity	fa:ArtifactType
fa:BufferElement	Object or space dedicated to hosting objects (e.g. artifacts, pallets)	fa:BufferElementType
fa:MachineTool	Generic machine tool	fa:MachineToolType
fa:Robot	Robot	fa:RobotType
fa:Pallet	Object dedicated to hosting artifacts that are transported in a system (e.g. production system).	fa:PalletType

fa:Tool	Tool	fa:ToolType

Subclass of ifc:IfcTask	Description	Subclass of ifc:IfctTaskType
fa:AssemblyTask	Task executing an assemby operation.	fa:AssemblyTaskType
fa:DisassemblyTask	Task executing an disassemby operation.	fa:DisassemblyTaskType
fa:MaintenanceTask	Task executing a maintenance operation.	fa:MaintenanceTaskType
fa:ManufacturingTask	Task executing a manufacturing operation.	fa:ManufacturingTaskType
fa:MachiningTask	Task executing a manufacturing operation.	fa:MachiningTaskType
fa:QualityControlTask	Task executing a quality control.	fa:QualityControlTaskType
fa:TransportTask	Task executing a transportation.	fa:TransportTaskType

Subclass of ifc:IfcControl	Description
fa:ProductionPlan	Plan of activities to be executed by a production system in terms demanded volume of artifacts, available resources, assignements, adopted control policies
fa:ProductionSchedule	Component of a fa:ProductionPlan specifying the demanded volumes for a type of artifact

SPARQL Queries

SPARQL is a W3C query language standard to retrieve and manipulate data stored in Resource Description Framework (RDF) format, thus including OWL ontologies.

The use of SPARQL queries asks for a SPARQL Endpoint, i.e. a server that can handle HTTP requests. SPARQL Endpoints are typically provided by RDF stores.

This section presents examples of SPARQL Queries tailored for the Factory Data Model. These examples must be intended as templates that can be customized by specifying RDF Datasets.

- Get part types
- Get process plans

Get Part Types

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX factory: <http://www.ontoeng.com/factory#>
select distinct ?parttype
WHERE {
    ?parttype rdf:type/rdfs:subClassOf* factory:ArtifactType .
}
```

Get Process Plans

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ifc: <http://ifcowl.openbimstandards.org/IFC4_ADD1#>
PREFIX ifcext: <http://www.ontoeng.com/IFC4_ADD1_extension#>
PREFIX factory: <http://www.ontoeng.com/factory#>
select distinct ?parttype ?pplan
WHERE {
  ?parttype rdf:type/rdfs:subClassOf* factory:ArtifactType .
  ?pplan rdf:type/rdfs:subClassOf* ifc:IfcTaskType .
  ?parttype ifcext:hasAssignedObject|^ifcext:hasAssignmentTo ?pplan .
```

```
11  }
```

SPARQL Updates

SPARQL 1.1 Update is a W3C language to create and modify RDF graphs, thus including OWL ontologies.

The use of SPARQL updates asks for a SPARQL Endpoint, i.e. a server that can handle HTTP requests. SPARQL Endpoints are typically provided by RDF stores.

This section presents examples of SPARQL Updates tailored for the Factory Data Model.

2.2 Instantiation of Factory Models

This section explains how a factory model can be instantiated in terms of:

- Assets composing the model.
- 3D models representing assets, with possible upgrades for virtual reality.
- Animations associated with assets using .json files.

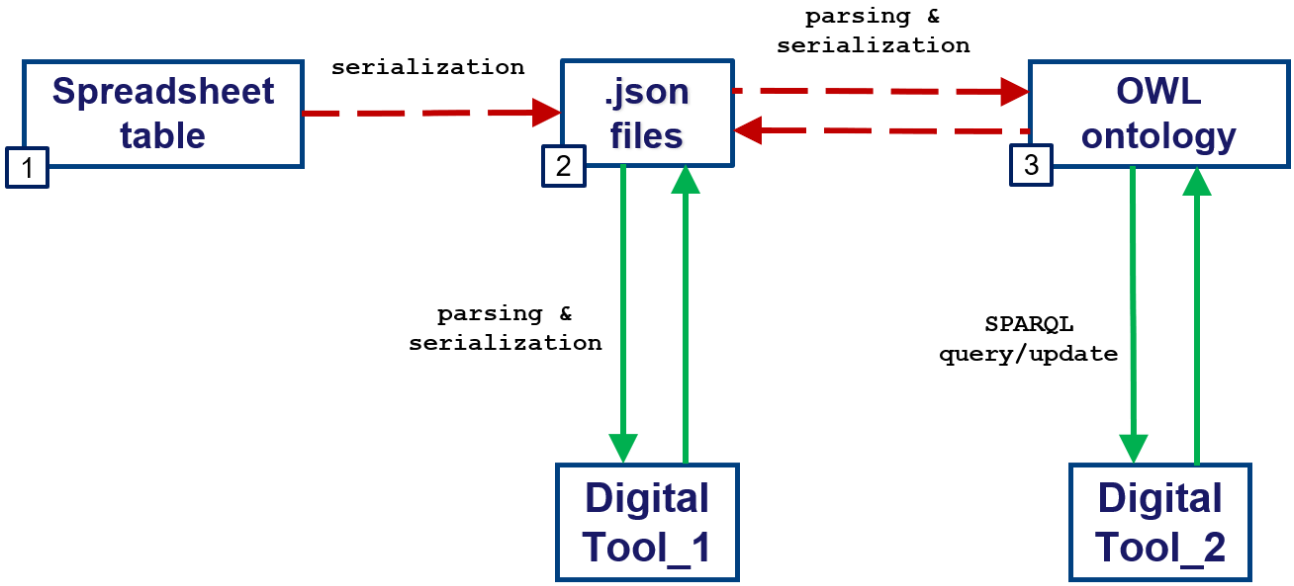
Instantiated factory models can be stored in dedicated data repositories.

Assets

Assets are basic elements composing a model, e.g. physical objects like machine tools, parts, conveyors, buffers, but also processes and plans.

Assets can be defined according to an instantiation workflow consisting of the following steps:

- spreadsheets
- .json files
- OWL ontology



Instantiation workflow

Assets in Spreadsheet

A virtual factory model, including the 3D scene, can be defined in terms of assets by taking advantage of a spreadsheet organized in two sheets:

- "**Context**": definition of context setup.
- "**Assets**": detailed definition of assets that are included (or not) in the scene. Assets not included in the scene are models/templates that are referenced or could be later instantiated in the scene.

The template (updated on 27/04/2021) can be downloaded. It is recommended **not to delete rows** in the "**Assets**" sheet.

VF_assets_template.xlsx VF_assets_template.xlsx - 48KB

Two examples are available in the dedicated section.

Context

The **Context** sheet contains the following properties (all required):

- "**UnitOfMeasureScale**": The default unit of measurement scale for distances (e.g. 0.01 stands for centimeter, whereas 1 stands for meter)
- "**Zup**": is a boolean parameter specifying the convention for the 3-D Cartesian coordinate system. The parameter is set to true if Z-axis is the vertical axis pointing up from the ground (i.e. Zup convention), or set to false if the Y-axis is the vertical axis pointing up from the ground (i.e. Yup convention). In both cases the right-handed system convention is assumed.
- "**RepoPath**": Path of the repository where the 3D models can be found. RepoPath can be defined as absolute or as relative to the application path. The file path of 3D models is defined relatively to RepoPath (e.g. '/repository/', 'https://example.com/datarepository/').

Cell “B1” in sheet "Context" returns the text as a .json file.

Assets

The **Assets** sheet contains items characterized by the following fields in columns:

- "**id**": unique identifier of the asset [required]
- “**inScene**”: equal to 1 if the asset is included in the scene, 0 otherwise [required]
- "**descr**": textual static description of the asset [optional]
- "**type**": the type of the asset, i.e. OWL class of the Factory Data Model it belongs to [required]
- "**model**": ID of the model of the asset (if existing), e.g. the model of a machine tool that is described in a catalog. In turn, the model can have a model. Please refer to the Object Typing pattern.
- "**file**": file path of the 2D/3D model representation, as relative to the RepoPath [optional]. The file can be available on a local or remote file system (see example), as any online repository accessible via HTTPS (see example). The “**file**” property can be used also as a reference to a specific component inside the hierarchy of a 3D model by adding a hashtag and the ID of the component to the file path (e.g. #componentID'). For instance, the “**file**” property will have the value "FileName.glb#nodeId" if it refers to a node with unique ID "nodeId" inside a GLTF file named "FileName.glb".
- "**unit**": Unit of measure to interpret a 3D representation (e.g. 0.01 stands for centimeter, whereas 1 stands for meter) [required if “file” is defined]
- "**position**": The Position of the asset in terms of x, y, z values. If missing, the default value is [0.0,0.0,0.0] or the value of the corresponding node in a GLTF hierarchy, if available.
- "**rotation**": The rotation of the asset as Euler angles YXZ defined in radians. If missing, the default value is the rotation of the corresponding node in a GLTF hierarchy (if available), otherwise [0.0,0.0,0.0].
- "**placementRelTo**": ID of the asset with respect to which the placement (position and rotation) is defined in relative terms. This means that a rototranslation must be applied with respect to the placement of the asset identified by the value of **placementRelTo**. This relation happens between nodes directly connected in a scene graph. For instance, the placement of a pallet can be defined as relative to the placement of a conveyor.

- "**parentObject**": ID of the asset that is decomposed (it may be empty or missing) when an aggregated asset is represented. If a **parentObject** is defined, then **placementRelTo** is defined as equal to parentObject. However, if a **placementRelTo** value is defined, then it is not necessarily also the **parentObject**. For instance, machine components decompose a workstation, whereas a pallet doesn't decompose a conveyor.

In addition, the following optional fields can be used to further characterize the scene in terms of relations and properties.

- "**connectedTo**": list of assets ID that are connected downstream to the asset. The list may be empty or missing [optional]
- "**assignmentTo**": list of assignments to other assets. The list may be empty or missing [optional]
- "**successors**": successor of a task [optional]
- "**taskTime**": execution time of a task [optional]
- "**bufferCap**": buffer capacity [optional]
- "**TTF**": time to failure (for devices like machines) in terms of probability distribution and value [optional]
- "**TTR**": time to repair (for devices like machines) in terms of probability distribution and value [optional]
- "**duration**": duration of a production plan [optional]
- "**quantity**": quantity to be produced in a production plan [optional]

Examples

Examples of a scene with assets instantiated in a spreadsheet are available together with the corresponding to the JSON files for simple cases, an assembled product, and a workstation.

Assets in JSON

The assets composing a factory model, including the 3D scene, can be defined according to the JSON schema described in this section. **JSON** (JavaScript Object Notation) is a lightweight text-based data-interchange format that is easy to read/parse and write/generate both for humans and machines. A .json file can be opened with any basic or advanced text editor (e.g. NotePad++, Visual Studio Code).

The resulting .json files can be also used to instantiate the ontology Factory Data Model thanks to the import functionality of OntoGui-Utlilities module.

Two examples are available in the dedicated section.

Schema

The .json schema consists of three root properties:

1. "**context**": definition of context setup.
2. "**scene**": definition of the 3D scene.
3. "**assets**": detailed definition of assets that are included or not in the scene. Assets not included in the scene are models/templates that are referenced or could be later instantiated in the scene.

Context

The **context** contains the following properties (all required):

- "**UnitOfMeasureScale**": The unit of measure scale (e.g. 0.01 stands for centimeter, whereas 1 stands for meter)
- "**Zup**": is a boolean parameter specifying the convention for the 3-D Cartesian coordinate system. The parameter is set to true if Z-axis is the vertical axis pointing up from the ground (i.e. Zup convention), or set to false if the Y-axis is the vertical axis pointing up from the ground (i.e. Yup convention). In both cases the right-handed system convention is assumed.

- "**RepoPath**": Path of the repository where the 3D models can be found. RepoPath can be defined as absolute or as relative to the application path. The file path of 3D models is defined relatively to RepoPath (e.g. '/repository/', 'https://example.com/datarepository/').

Scene

The scene is an array consisting of asset IDs that are included in the 3D scene.

Assets

The **assets** array contains items characterized by the following properties:

- "**id**": unique identifier of the asset [required];
- "**type**": The type of the asset, i.e. OWL class of the Factory Data Model it belongs to [required]
- "**model**": ID of the model of the asset (if existing), e.g. the model of a machine tool that is described in a catalog. In turn, the model can have a model. Please refer to the Object Typing pattern.
- "**representations**": Array of 2D/3D representations of the asset. Each item of the array may have the following properties:
  - "**file**": file path of the 2D/3D model representation, as relative to the RepoPath [optional]. The file can be available on a local or remote file system (see example), as any online repository accessible via HTTPS (see example). The "**file**" property can be used also as a reference to a specific component inside the hierarchy of a 3D model by adding a hashtag and the ID of the component to the file path (e.g. #componentID'). For instance, the "**file**" property will have the value "FileName.glb#nodeId" if it refers to a node with unique id "nodeId" inside a GLTF file named "FileName.glb".
  - "**unit**": Unit of measure to interpret a 3D representation (e.g. 0.01 stands for centimeter, whereas 1 stands for meter) [required if "file" is defined]
- "**position**": The Position of the asset. If missing, the default value is [0.0,0.0,0.0];
- "**scale**": The scaling of the asset. If missing, the default value is [1.0,1.0,1.0]. Scaling is defined independently of the unit of measurement of the 3D representation (cf. "unit" in "representations");

- "**rotation**": The rotation of the asset as Euler angles YXZ defined in radians. If missing, the default value is the rotation of the corresponding node in the GLTF hierarchy (if available), otherwise [0.0,0.0,0.0].
  . Alternatively the rotation can be defined as a quternion.
- "**placementRelTo**": ID of the asset with respect to which the placement (position and rotation) is defined in relative terms. This means that a rototranslation must be applied with respect to the placement of the asset identified by the value of **placementRelTo**. This relation happens between nodes directly connected in a scene graph. For instance, the placement of a pallet can be defined as relative to the placement of a conveyor.
- "**parentObject**": ID of the asset that is decomposed (it may be empty or missing) when an aggregated asset is represented. If a **parentObject** is defined, then **placementRelTo** is defined as equal to parentObject. However, if a **placementRelTo** value is defined, then it is not necessarily also the **parentObject**. For instance, machine components decompose a workstation, whereas a pallet doesn't decompose a conveyor.

In addition, the following optional properties can be used to further characterize the factory model in terms of relations and properties:

- "**connectedTo**": array of assets ID that are connected downstream to the asset. The list may be empty or missing [optional]
- “**assignmentTo**": array of assignments to other assets. The list may be empty or missing [optional]
- “**successors**": array of successors of a task [optional]
- “**taskTime**": execution time of a task [optional]
- “**bufferCap**": buffer capacity [optional]
- “**TTF**": time to failure (for devices like machines) in terms of probability distribution and value [optional]
- “**TTR**": time to repair (for devices like machines) in terms of probability distribution and value [optional]
- “**duration**": duration of a production plan [optional]
- “**quantity**": quantity to be produced in a production plan [optional]

Examples

Examples of a scene with assets instantiated in a JSON file are available together with the corresponding spreadsheets for simple cases, an assembled product, and a workstation. These JSON files can be used to visualize a 3D scene in VEB.js.

Assets in Ontology

Assets can be defined by instantiating the Factory Data Model as an ontology Abox.

The VLF tool OntoGui can support the instantiation in the following ways:

- direct instantiation using Individual Manager module
- instantiation of production systems using System Design module
- import and parsing of JSON files (assets and animations) using Utilities module

3D Models of Assets

Assets composing a model of a factory also need to be modeled in terms of their 3D representation. To go in this direction, 3D models must be available. They can be obtained according to two main options:

1. **Use existing 3D models**. 3D models of industrial objects are often available in online databases (e.g., GrabCAD) or directly provided by OEMs in official catalogs. These models are usually very detailed, thus an aspect to be considered is being able to end up with manageable models in terms of dimension and complexity.
   Furthermore, 3D models must be available in a neutral format (e.g. .STEP or .IGES), so that they can be easily imported in general software environments for further processing.
   IMPORTANT: 3D models downloaded from external sources are often constrained by privacy policies and rights preventing some or all their possible uses. This can prevent the possibility to publish this material on this website, even if not commercial use is foreseen. Please, before proceeding, check information about licenses and favor open licensing schemes.
2. **Generate 3D models**. A second option is to generate 3D geometries using CAD modeling tools (e.g. Solidworks, Inventor). Although more difficult and time-consuming, consider this option to avoid possible issues related to licensing and rights.

Scaling and orientation

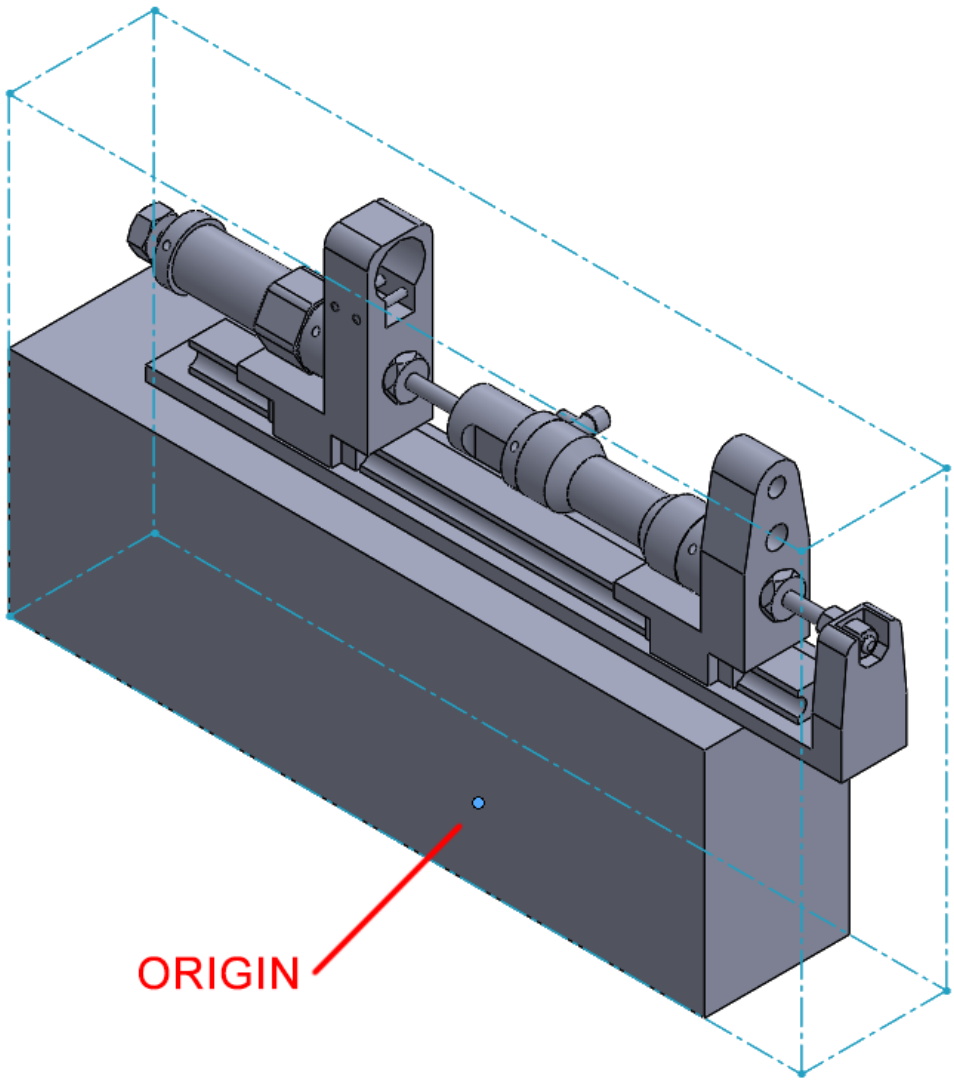
3D models in the scene have to be properly scaled. The hypothesis, unless differently specified, is that **all the measures are expressed in millimeters (mm)**.

Furthermore, the orientation of each single component must be coherent with the other objects in the scene. In many CAD environments, it is possible to specify the convention used for the orientation of the z-axis (z-up option).

Grouping and referencing

Assets in a factory are often complex objects consisting of many components assembled together (e.g., a workstation). It is important to build 3D models in order to be easily and clearly composed to form complex assets.

It's important to have a univocal reference point for each asset in the scene. Namely, this point is the origin of the corresponding 3D model. Origins may often have unusual positions, especially when 3D models have been downloaded from online databases. To make the use and re-use of 3D models in the scene coherent, a reasonable solution is setting the origin in the central point of the bottom face of the bounding box of the component.



Standard positioning of the origin for a 3D model.

The positions of the origins can be used to define a complex asset by specifying the relative position of its components according to the following steps:

1. Each complex asset will have a parent empty object to which its static components will be referenced and positioned.
2. Subcomponents have to be grouped matching the hierarchy of the assembly, i.e., assets forming a sub-assembly must be grouped together.
3. Parenthood relationships between the 3D models have to be defined specifying the associate relative positioning. CAD environments use this approach to manage multipart assemblies and dependencies.

Setting up the origin

This operation can be accomplished in CAD software such as Solidworks, or downstream when editing the exported GLTF/OBJ in Blender.

1. Generate the bounding box of the 3D model.
2. Make the central point of its bottom face explicit.
3. Move the origin to make it coincident with the point.

Formalize information of assets using 3D models

The described procedure and conventions are used to derive information on 3D models to support the instantiation of assets by specifying their representation. It must be stressed that not all components in an aggregated object must be necessarily mapped to assets instantiated in the digital model (spreadsheet, JSON or ontology). Indeed, this mapping should be limited to components that are associated with needs in the following (non-exhaustive) list:

- the placement (either position or rotation) is customized with respect to the default values defined in the 3D model file (e.g. GLTF file)
- a description or any other property (type, model, connection, assignment) must be defined, as documented for spreadsheets and JSON files.
- the component must animated independently of its parent. Static components typically are not associated with this need.

Please note that if a component is explicitly defined as an instantiated asset, then also its parent must be explicitly defined for the sake of consistency. In addition, a unique reference to the component inside the 3D model must be provided (e.g. adding '#componentID' to the file name, cf. "file" property in spreadsheet and JSON file). This means that also the ID of components in the 3D model must be uniquely identifiable.

Examples are provided in the use case section for an assembled product and a workstation.

3D Models for Virtual Reality

3D models, defined according to what described in the section 3D Models of Assets, are not ready to be used in a VR environment. In fact, specifications related to materials and the behavior of light for rendering are required.

Several file formats exist to support VR. The recommended option is GLTF, an open standard developed and maintained by the Khronos Group. It supports many features, i.e., 3D model geometry, appearance, scene graph hierarchy, and animation. Many CAD environments are capable of directly export models in the GLTF format.

By default, **the unit of measure in the GLTF standard is the meter**.

In order to be ready for the use in a VR environment, 3D models of assets must be further elaborated to define materials and properties for the rendering. Depending on the expected outcome and the level of realism to achieve, two options are possible:

1. GLTF + standard materials for basic 3D representation.
2. GLTF + PBS textures for high realism.

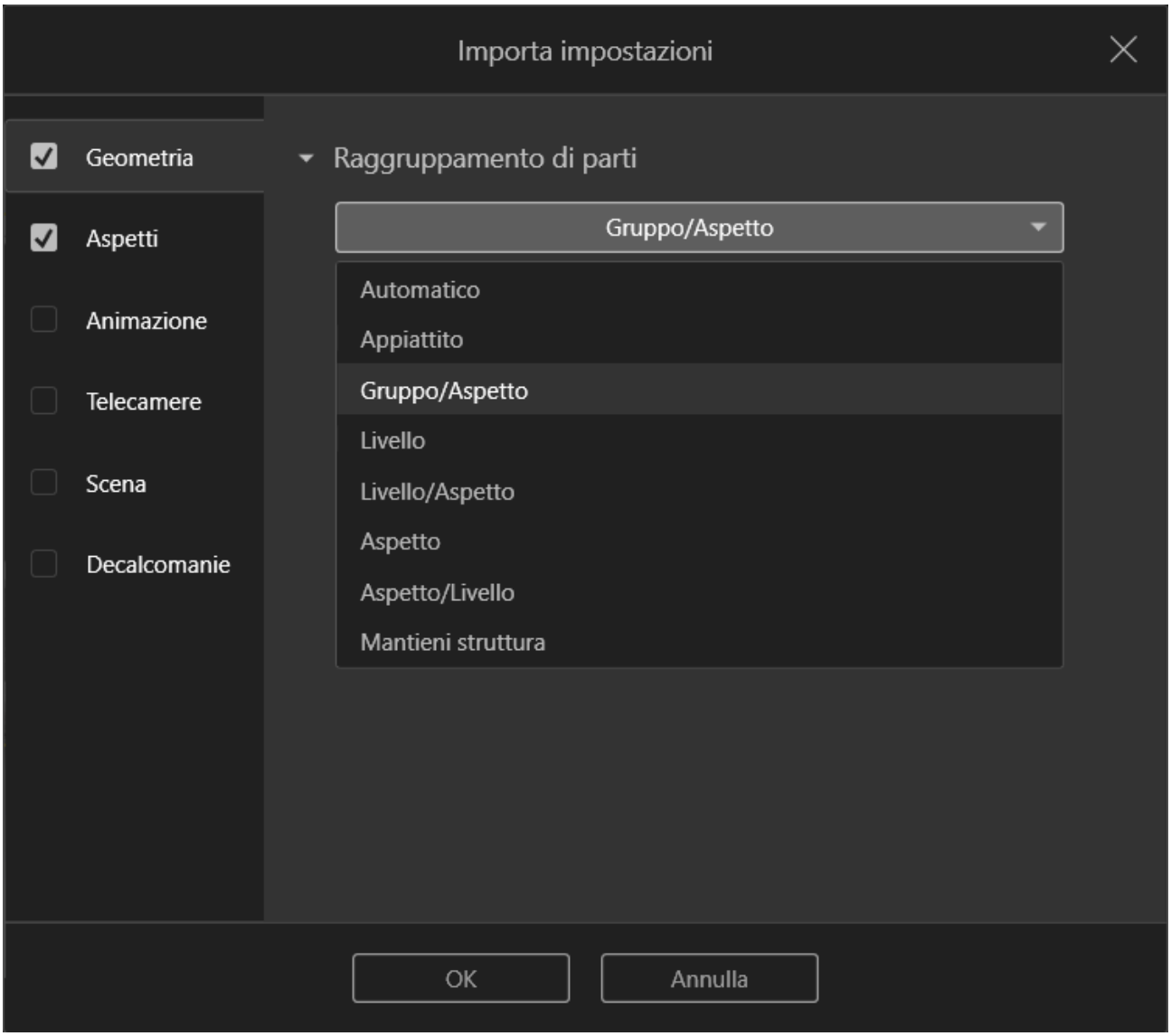
GLTF + Standard Materials

Basic materials can be applied with regular rendering software such as KeyShot and Solidworks Visualize and later exported to GLTF.

GLTF + Standard Materials using SolidWorks Visualize

SolidWorks Visualize is a software supporting the definition of VR-ready models. When importing a Solidworks Assembly file (SLDASM) to Solidworks Visualize, a dialog pops up to select the desired grouping method to instantiate the hierarchy of the sub-components. This hierarchy must match the specifications described in the section related to 3D models of assets, while providing the user the capability of applying different materials to the different sub-components.

For this purpose, the most efficient import setting option is by “Group/Aspect”.



Import settings in SolidWorks Visualize

Thus, drag and drop commands in the software environment can be used to assign materials to the components in the 3D viewer/hierarchy tree. Further information related to how assign materials can be found in the official Solidworks Visualize Manual.

It is recommendable, if possible, to use basic materials (e.g. opaque plastic materials), since more complex appearances (paint materials, translucent, etc...) may cause unwanted artifacts when the model is exported to the GLTF format.

Finally, the user can export the resulting models in either the GLTF or GLB formats.

GLTF + PBR Textures

Physically based rendering is a technique enabling an increased level of detail by adopting high resolution textures applied to the GLTF models.

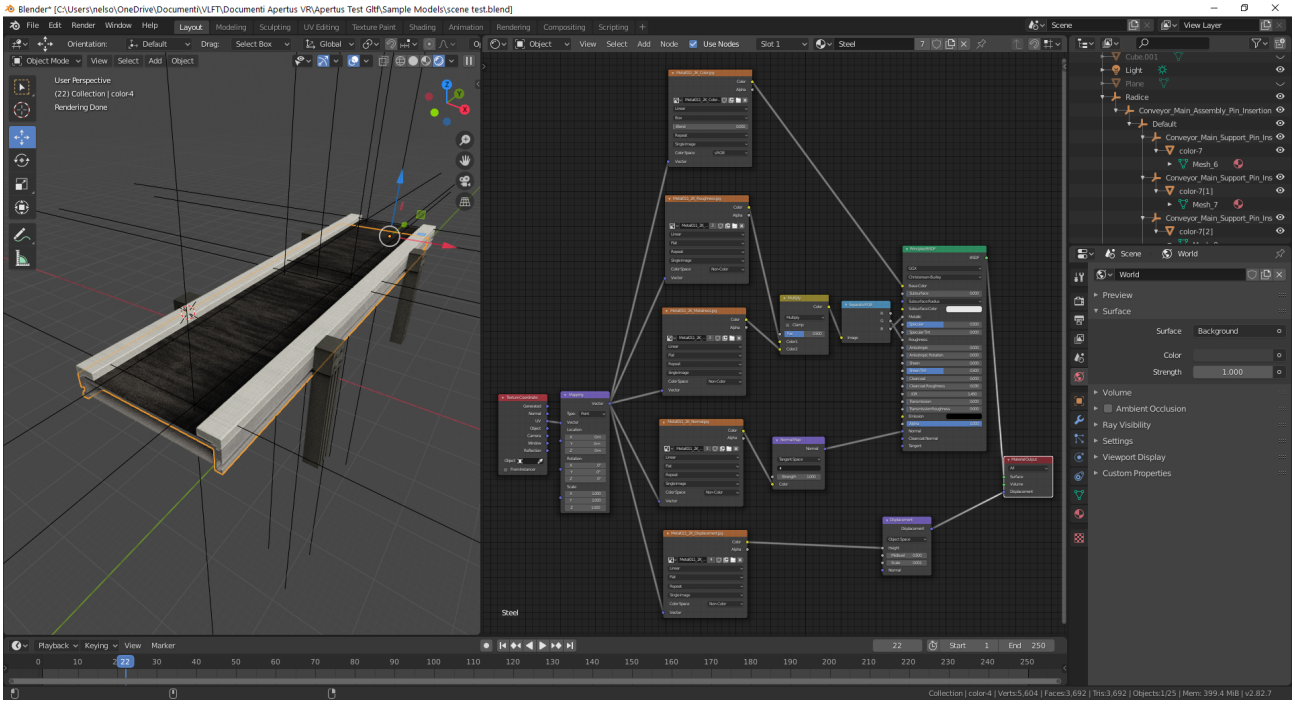
GLTF + PBR Textures using Blender

Blender is a free and open-source 3D computer graphics software tool set.

Differently from other rendering tools, Blender does not natively include materials to be applied. Thus, materials have to be collected exploiting online sources such as CC0 Textures, providing an extensive catalog of realistic looking materials that can be freely downloaded under a Creative Commons license. Each material can be downloaded either in JPG (lighter) or PNG format, with different texture resolution (from 1K to 8K). These settings will have a huge impact on the overall size of the final GLTF model, so it is important to make a trade-off between quality and size of the models.
As a general rule of thumb, opaque materials with overall regular textures like metals, paints, and rubber can be downloaded with minimum resolution without showing clearly visible defects once applied to the model: the JPG/1K setting is valid in most cases. Complex materials ready for the PBR method are composed of a multilayer array of different textures (namely images) associated to different visual properties that will be eventually embedded in the model: Color, Displacement, Metalness, Normal, Roughness.

The steps to obtain realistic VR models are the following:

1. The setup of the material is performed through the Shader Editor interface, where each level of the texture is assigned to their specific nodes. It may be necessary to fine adjust some parameters if textures appear to be off scale.



Blender's interface for shader editor.

The user can decide to export the resulting models in the GLTF, GLB, or GLTF with separate directories for texture resources. By right-clicking the root node of the model and choosing “Select Hierarchy” all the subcomponents downstream will be selected. The "Compression" option might be needed to reduce the size of the resulting models, though it can also create some unwanted result, so it's recommendable to check the exported models afterwards.

More details can be found in the use case 3D Modelling of a Workstation for Virtual Reality.

Further resources

Further information can be found in dedicated online resources:

Khronos Art Pipeline for glTF

THE PBR GUIDE

Blender 2.8 PBR Texturing for Beginners

Animations

The animation of assets in a factory model can be defined according to the JSON schema described in this section. **JSON** (JavaScript Object Notation) is a lightweight text-based data-interchange format that is easy to read/parse and write/generate both for humans and machines. A .json file can be opened with any basic or advanced text editor (e.g. NotePad++, Visual Studio Code).

Schema

The schema is composed of three root properties:

- “**context**”: set of properties of initial setup for the whole animation
- "**nodes**": an array defining what happens during the animation
- "**bookmarks**": an array defining bookmarks

The “**context**” is characterized by the following set of properties:

- "**UnitOfMeasureScale**": (optional) unit of measure scale (e.g. 0.01 stands for centimeter, whereas 1 stands for meter). If not defined, then the unit of measure of the associated scene is adopted.
- "**assetTrail**": value set to true if trails (red lines) must be shown to track the movements of moving assets, false otherwise to hide all trails.

Each item in the “**nodes**” array contains the following properties (all required):

- "**id**": unique identifier of the asset that is characterized in terms of animation [required]
- "**actions**": array of actions that are relevant for the animation

Each item in the “**actions**” array contains the following properties (all required?):

- "**trigger**": specifying when the action takes place with properties “**type**” and “**data**”
- "**event**": specifying what action takes place with properties “**type**” and “**data**”

Possible values for “**type**” of “**trigger**”:

- "**timestamp**": the value in “**data**” is interpreted as time in milliseconds after the start of animation

Possible values for “**type**” of “**event**”:

- "**animation**": animation of the asset starts. The value of “**data**” property is interpreted as the path of the file (.bin or .txt) defining the animation sequence. The positions and rotations are defined in absolute terms, but if the additional property “**placementRelTo**” is defined then it must be intended in relative terms.
- "**animationAdditive**": animation of the asset starts. The value of “**data**” property is interpreted as the path of the file (.bin or .txt) defining the animation sequence. The animation is intended as additive (incremental) with respect to the current placement (position, rotation) of the asset.
- “**show**”: the asset is shown in the scene. The place where the asset appears is defined with additional properties (as for the scene definition): "**position**" (default value [0.0,0.0,0.0]), “**scale**" (default value [1.0,1.0,1.0]), "**rotation**" (default value [0.0,0.0,0.0] Euler angles YXZ in radians), "**placementRelTo**". For instance, if only "**placementRelTo**" is defined, then position, scale and rotation have default values.
- “**attach**”: the asset keeps its absolute position, rotation and scale, but its location in the scene graph is updated according to the value of property "**placementRelTo**". If "**placementRelTo**" is empty or not defined, then the asset is attached to the root node in the scene graph.
- “**hide**”: the asset is hidden from the scene
- “**state**”: the asset changes its state in terms of 3D representation and description. The state is specified by the following properties, all optional:
  - “**data**” property is a string with the file path of the 3D representation file (e.g. .obj). The new 3D representation completely replace the (possibly) existing mesh while inheriting its attributes (e.g. position, rotation, scaling).
  - "**descr**" is a string with the description of the current state. This description is incremental and does not replace the static description of the asset.
- “**link**”: link to a (text) file that is specified by the following properties:
  - “**data**” is a string with the local file path
  - "**descr**" is a string with the description of the content that is found at the link
  - “**URL**” is the address where the file can be downloaded from
- “**trail**”: setting the visualization of the trail (red line) to track movements of the specific asset with the following property.

- “**value**” is a boolean set to **true** if the trail generation is started, **false** if the trail generation is stopped.

Animation sequence

The animation (“animation” or “animationAdditive”) sequence is defined in a **.txt** file as a list of numerical values to be interpreted as follows:

- The first line defines the total number of frames (N) in the file.
- The second line defines the FPS (frame per second).
- Lines from 3 till N+2 define the position in the 3D space for each frame.
- Lines from N+3 till 2*N+2 define the rotation (as Euler angles YXZ in radians or alternatively as quaternion for "animation" and axis-angle for "animationAdditive") in the 3D space for each frame.

Examples

Examples of animation instantiated in a JSON file are available together with the corresponding JSON file instantiating the scene and assets. Animations can be played in VEB.js.

Data Repositories

Repositories are needed to store Factory Models in terms of assets, 3D files, animations.

Data repositories can be either installed locally or made available remotely via communication protocols (e.g. HTTP, FTP).

Local Repository

The instantiated factory models can be serialized as text/binary files and saved on a local file system that is directly accessible by digital tools:

- Assets can be saved in .xlsx (Spreadsheet), .json, or .owl/.rdf (ontology) files
- 3D models can be saved in various formats like .OBJ and .GLTF/.GLB
- Animations can be saved in .json and .txt files

Localhost

If a digital tool needs http(s) communications to retrieve factory models, then a localhost server can be launched, e.g. using JavaScript or Python scripts.

Remote Repository

Remote repositories are typically installed on servers. In most cases, a remote repository can be made operational also locally by running a localhost.

RDF store

Commercial and non-commercial RDF stores, also named Triplestores, are available, e.g. Stardog, Apache Jena Fuseki, Virtuoso.

RDF stores are typically structured into Datasets/Databases.

Apache Jena Fuseki

Apache Jena Fuseki is a SPARQL server that can run as an operating system service, as a Java web application (WAR file), or as a standalone server.

After downloading and unzipping Apache Jena Fuseki .zip file, the simplest way to run Fuseki as a Standalone Server by executing `fuseki-server` from the command prompt (or launching `fuseki-server.bat` in Windows).

The default location of a Fuseki server installation is http://localhost:3030

A demo Fuseki SPARQL Endpoint with a Dataset named "VLFT" is available at `http://mi-eva-d001.stiima.cnr.it/fuseki`

Online repository for binary/text files

GitHub repositories can be exploited to store binary files (e.g. 3D models, animation sequences) and make them available via secure HTTP connections.

1. After your GitHub registration, you can create a new repository as "Public".

2. Upload binary files to the repository ("Add file", "Upload files") and "Commit changes", creating the repository directories that you prefer.

A couple of examples:

- repository https://github.com/KhronosGroup/glTF-Sample-Models with file 2.0/WaterBottle/glTF-Binary/WaterBottle.glb
- repository https://github.com/wterkaj/RepoExample with file example_1.json

Option 1

Each file in the repository can be directly retrieved with a URL structured as `https://raw.githubusercontent.com/$User/$Repository/$Branch/$LocalFilePath` where `$User` is the registered GitHub user, `$Repository` is the chosen name of the repository, `$Branch` is the selected versioning branch (e.g. "main"), and `$LocalFilePath` is the local of the file in the repository. The example files are available at these URLs:

- https://raw.githubusercontent.com/KhronosGroup/glTF-Sample-Models/master/2.0/WaterBottle/glTF-Binary/WaterBottle.glb
- https://raw.githubusercontent.com/wterkaj/RepoExample/main/example_1/example_1.json

Option 2

For a given GitHub repository it is possible to activate the GitHub Pages option ("Project site") that turns the repository into a website. The first example file is available at this URL:

- https://wterkaj.github.io/RepoExample/example_1/example_1.json

3. VLF Tools and Libraries

The Virtual Learning Factory (VLF) Tools are digital tools that can be integrated thanks to the interoperability provided by interacting with the common factory data model. In principle, any digital tool can be included in the VLFT if the following requirements are met:

- The digital tool offers a way to access and modify (if needed) its internal data structures, by means of exchange files or through an application programming interface (API).
- A customized software component (a.k.a. plugin, connector) is developed to import/export data from/to the VLF Knowledge taking in due consideration both the reference factory data model and the specific data model adopted by the digital tool.

The development of the connector can be eased by the use of programming libraries and pre-defined SPARQL queries that are part of the VLFT.

The integrated digital tools support various phases along the factory lifecycle, such as production system design, layout design, system performance evaluation and visualization of production system performance. These phases can be sequential or more frequently iterated in loops. VLFT include the following groups of VLF Tools:

- Graphical User Interfaces (GUIs) to access the knowledge base, explore and generate new factory models. GUIs and queries can support the definition and retrieval of production system configurations (i.e. part types, process plans, operations, capabilities of the production resources like machine tools, transporters, storage systems) and also planning and monitoring of manufacturing execution (scheduled production volumes, available resources, key performance indicators (KPI) like throughput, average inventory, average lead time). The prototype tool OntoGui developed by CNR-STIIMA can be used for this purpose.
- Performance evaluation via Discrete Event Simulation. Commercial off the shelf tools and academic tools (e.g. Java Modeling Tools) can be integrated in VLFT by developing plugins that support the automatic generation of simulation models and

the automatic retrieval and storage of results to be exploited by other users and applications.

- 3D visualization and interaction with production systems and production resources by means of Virtual and Augmented Reality. These tools support the design, reconfiguration, training and maintenance of production systems and resources. Digital tools based on the libraries babylon.js and ApertusVR are currently available in VLFT.
- Human Modeling in manufacturing. Tools and methodologies supporting safety and well-being, ergonomics (e.g. RULA, OCRA), human-machine process design and monitoring (e.g. MTM, MOST).

3.1 OntoGui

OntoGui is a software tool providing a graphical user interface to support:

- The fast evaluation of a T-box under development by concurrently instantiating a corresponding A-box, thus implementing a kind of test-driven development approach.
- The generation of RDF data sets to be used as input for other ontology-based applications, without needing customized graphical user interfaces or data converters.

The main window of OntoGui is a **Control Panel** that can manage (networks of) ontologies The Control Panel gives also access to ontology tools like **OWL Individual Manager**.

OntoGui is developed as a desktop application in C++ making use of wxWidgets Cross-Platform GUI Library [2] for the creation of graphical elements, and the RdfCpp library. RdfCpp is a C++ library, based on Boost Library [3] and Redland RDF [4] (enabling the parsing and generation of RDF triples), that provides classes and functions to manage a network of RDF graphs, parse and generate OWL individuals, parse OWL axioms of a T-box (i.e., equivalent classes; subclasses; restrictions of any degree involving universal quantifier, existential quantifier, or cardinality constraints; domain and range of properties). Moreover, RdfCpp supports the connection with different RDF store solutions: (a) file-based; (b) Stardog triplestore [5].

OntoGui is currently available as a Windows application. The academic version of OntoGui can be freely downloaded and used for non-commercial applications:

OntoGui application OntoGui.zip - 10MB

The Root folder (*$ROOT*) is where the OntoGui executable (i.e. OntoGui32.exe or OntoGui32_academic.exe) is located. Several .dll are included in the root folder and must be available to run OntoGui. Depending on the configuration of the Windows operating system, it may be necessary to install the MVC++2010 Redistributable Package (x86 or x64).

The XML file «*$ROOT/ConnConfig.xml*» can be used to set configuration options. The folder «*$ROOT/doc*» contains documents. The folder «*$ROOT/log*» contains log file that

may be generated during the execution of OntoGui. The folder «*$ROOT/repository*» is the default local repository.

References

1. Terkaj W (2017) OntoGui: a Graphical User Interface for Rapid Instantiation of OWL Ontologies. Proceedings of the Workshop Data Meets Applied Ontologies, Joint Ontology Workshops 2017, CEUR Workshop Proceedings, vol. 2050
2. https://www.wxwidgets.org/
3. http://www.boost.org/
4. http://librdf.org/
5. http://www.stardog.com/

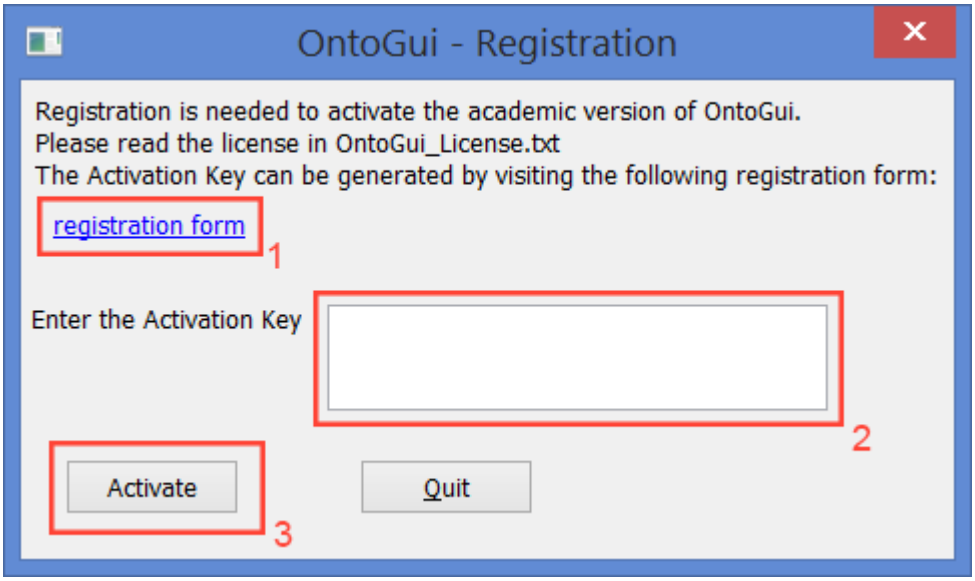
Applications

- Colledani M, Pedrielli G, Terkaj W, Urgo M (2013) Integrated Virtual Platform for Manufacturing Systems Design. Procedia CIRP 7:425-430. doi:10.1016/j.procir.2013.06.010
- Terkaj W, Sojic A (2015) Ontology-based Representation of IFC EXPRESS rules: an enhancement of the ifcOWL ontology. Automation in Construction, 57:188-201. ISSN: 0007-8506. doi:10.1016/j.autcon.2015.04.010
- Sojic A, Terkaj W, Contini G, Sacco M (2016) Modularising ontology and designing inference patterns to personalise health condition assessment: the case of obesity. Journal of Biomedical Semantics, 7:12. ISSN: 2041-1480. doi:10.1186/s13326-016-0049-1
- Terkaj W, Gaboardi P, Trevisan C, Tolio T, Urgo M (2019) A digital factory platform for the design of roll shop plants. CIRP Journal of Manufacturing Science and Technology, 26:88-93. ISSN: 1755-5817. doi:10.1016/j.cirpj.2019.04.007

How to start

Registration

Please read the license in *OntoGui_License.txt*. In case of commercial application you can contact the author at wterkaj@gmail.com or walter.terkaj@stiima.cnr.it



Registration form

When launching the executable the first time, a registration window appears:

1. Link to the web page for the OntoGui registration. This registration must be repeated for each workstation/laptop where OntoGui is used.
2. Enter the Activation Key generated after the registration
3. Check the Activation Key. If successful, the registration and activation will not be repeated as long as the software tool is used on the same computer.

OntoGui Modules

OntoGui provides access to the following main modules:

- Control Panel
- Individual Manager
- System Design
- Utilities

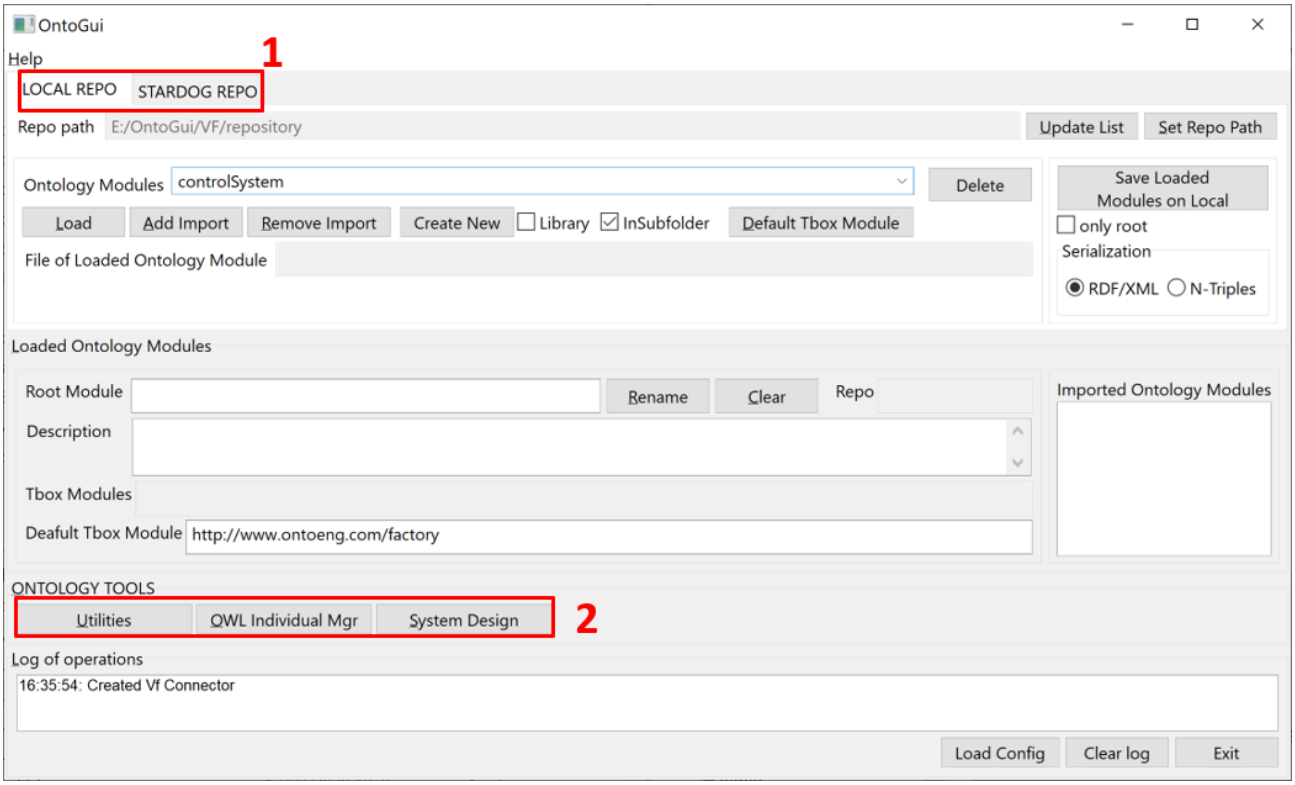
In addition, examples showing how OntoGui can be exploited are described in the Use Cases section for the Flow Shop, Job Shop, and Hybrid Flow Shop systems architectures.

Control Panel

The main window of OntoGui is a Control Panel that can manage (networks of) ontologies in a file-based repository or other more scalable RDF stores by selecting different repo tabs:

- *LOCAL REPO*, i.e. local file system
- *STARDOG REPO*, i.e. Stardog Knowledge Graph (https://www.stardog.com/)

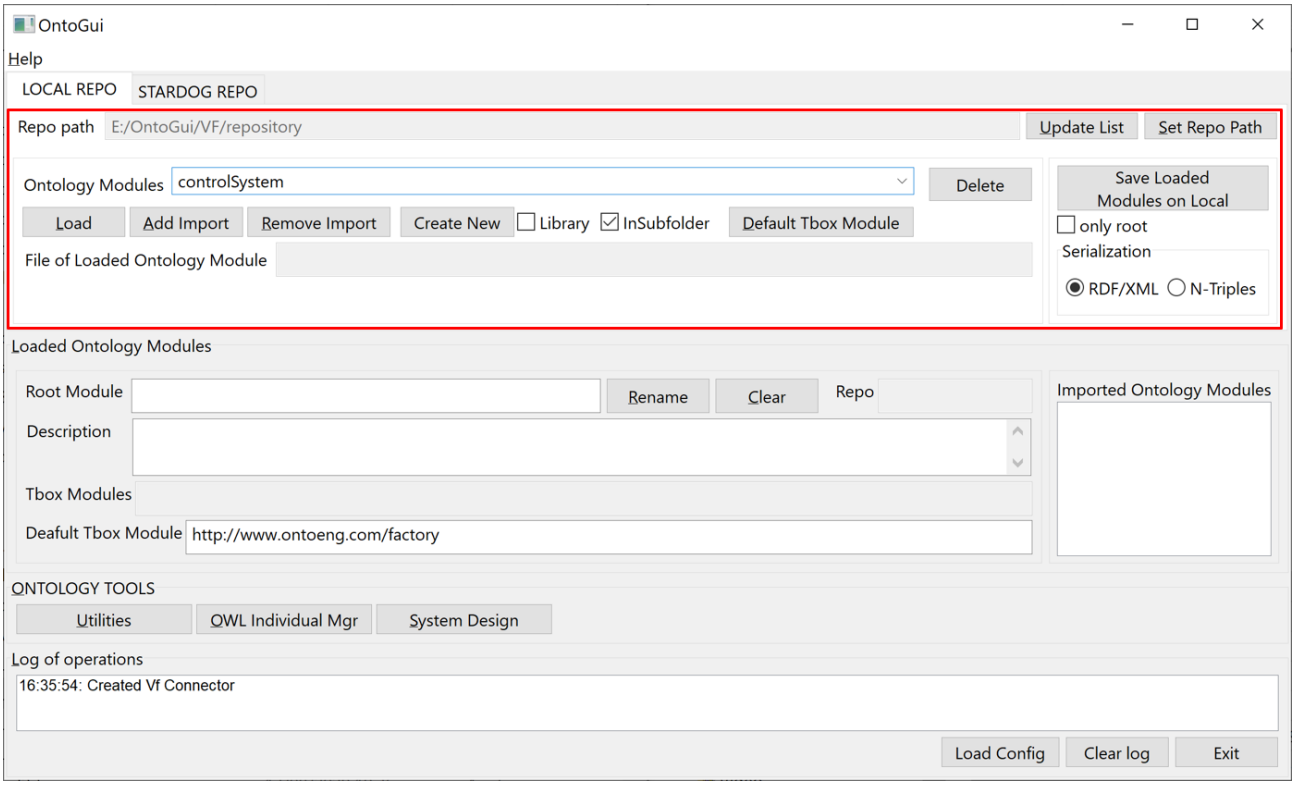
The Control Panel allows to load an existing ontology module and its dependencies, create a new A-box module, define new import relations between modules, and save the modules in any of the available repository.



OntoGui Control Panel

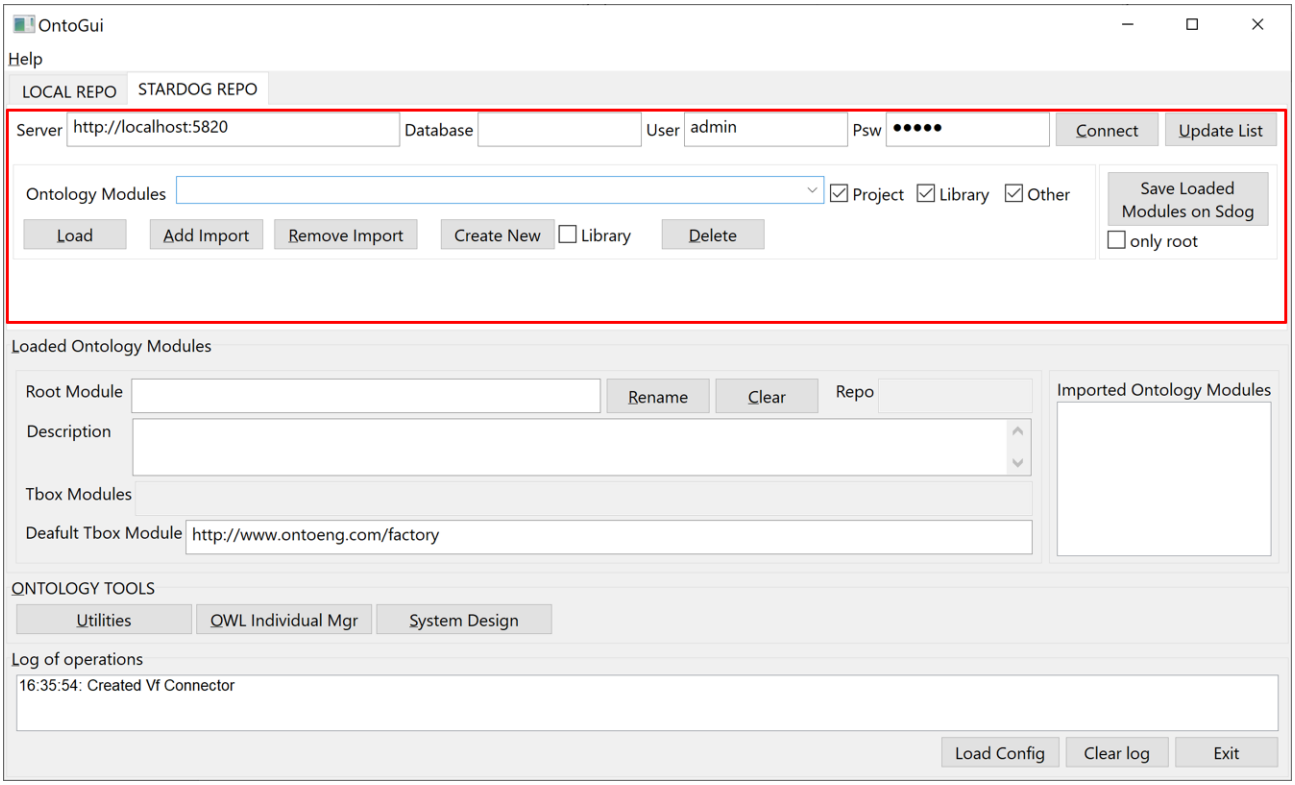
1. Selection of available repositories.
2. Control Panel: it provides also access to software tools and plugins.

LOCAL REPO



- **Repo path**: Location of the local repository in the file system.
- **Update List** button to update the list of the ontology modules available in the selected repository.
- **Set Repo Path** button to set the location of the local repository within the file system.
- **Ontology Modules** combo box lists the ontology modules available in the selected repository. This field can be modified to define the name a new module.
- **Load** button to load the ontology module selected in the list.
- **Add Import** button: the ontology module selected in the list is imported by the loaded ontology module.
- **Remove Import** button: the ontology module selected in the list is removed from the import list of the loaded ontology module.
- **Create New** button: a new ontology module is created using the name specified in the combo box. The new module imports the default Tbox module. If 'InSubfolder' is flagged then a new module is placed in a dedicated subfolder within the repository folder.
- **Delete** button to delete the selected ontology module.
- **Default Tbox Module** button to select the ontology module that is imported by default when a new module is created.
- **Save Loaded Modules on Local** button to save the loaded ontology module in the current repository.If 'only root' is flagged, only the root ontology module is saved on the repository. In 'Serialization' the serialization format can be selected.

STARDOG REPO

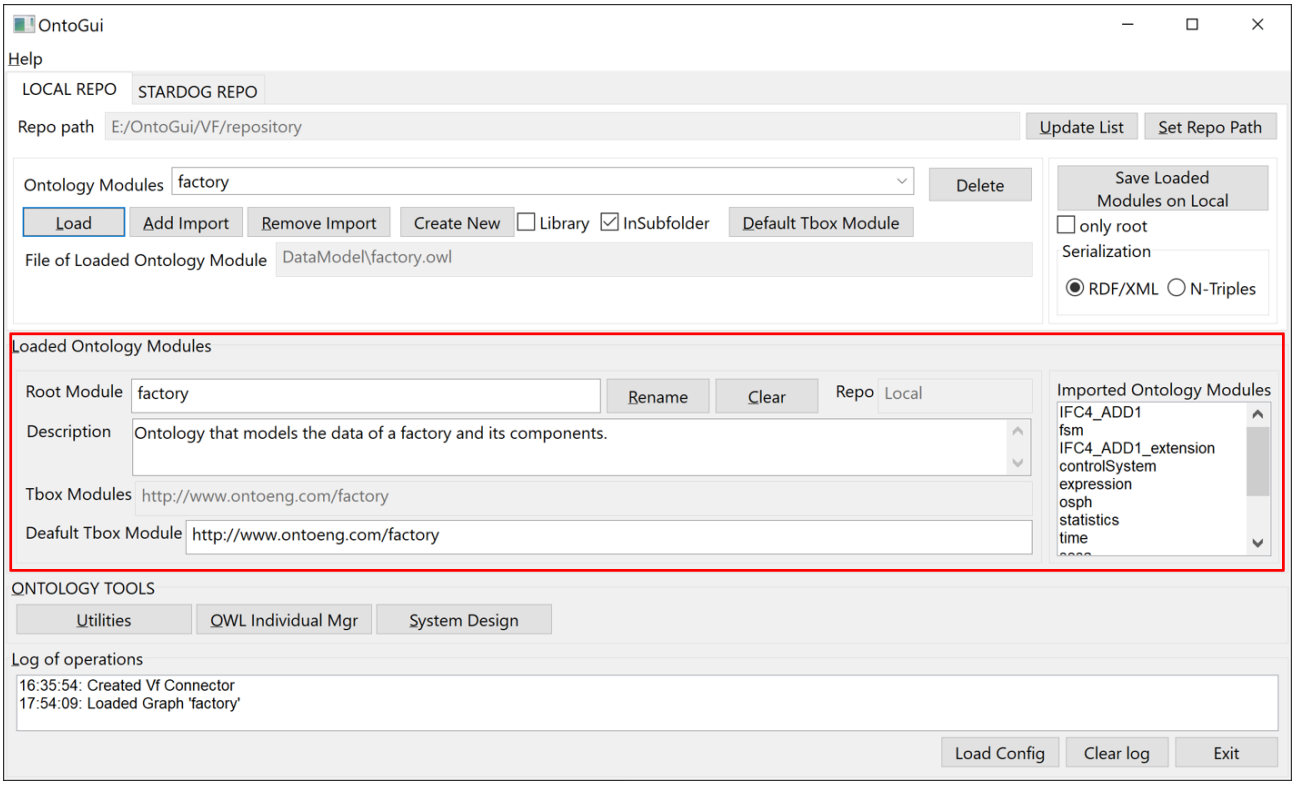


- **Server**: address of the Stardog server (port is included)
- **Database**: selected database in the Stardog server
- **User**: user credentials to access the Stardog server
- **Psw**: password credentials to access the Stardog server
- **Connect** button to connect to the repository
- **Update List** button to update the list of the ontology modules available in the selected repository.
- **Ontology Modules** combo box lists the ontology modules available in the selected repository. This field can be modified to define the name a new module.
- **Load** button to load the ontology module selected in the list.
- **Add Import** button: the ontology module selected in the list is imported by the loaded ontology module.
- **Remove Import** button: the ontology module selected in the list is removed from the import list of the loaded ontology module.
- **Create New** button: a new ontology module is created using the name specified in the combo box. The new module imports the default Tbox module.
- **Delete** button to delete the selected ontology module.

- **Save Loaded Modules on Sdog** button on Local to save the loaded ontology module in the current repository. If 'only root' is flagged, only the root ontology module is saved on the repository.

Loaded Ontology Modules

The central part of the Control Panel presents information about the loaded ontology modules, whatever is the source repository.



- **Root Module** shows the name of the root ontology module that is currently loaded. This field can be modified to change the name by pressing the **Rename** button.
- **Descriiption** of the root ontology module that is currently loaded.
- **Clear** button to clear the loaded ontology modules from memory.
- **Repo** shows which is the repository that is currently accessed.
- **Imported Ontology Modules** lists the ontology modules that are directly or indirectly imported by the root ontology module.
- **Default Tbox Module** defines which ontology module is imported by default when a new ontology module is created.

OWL Individual Manager

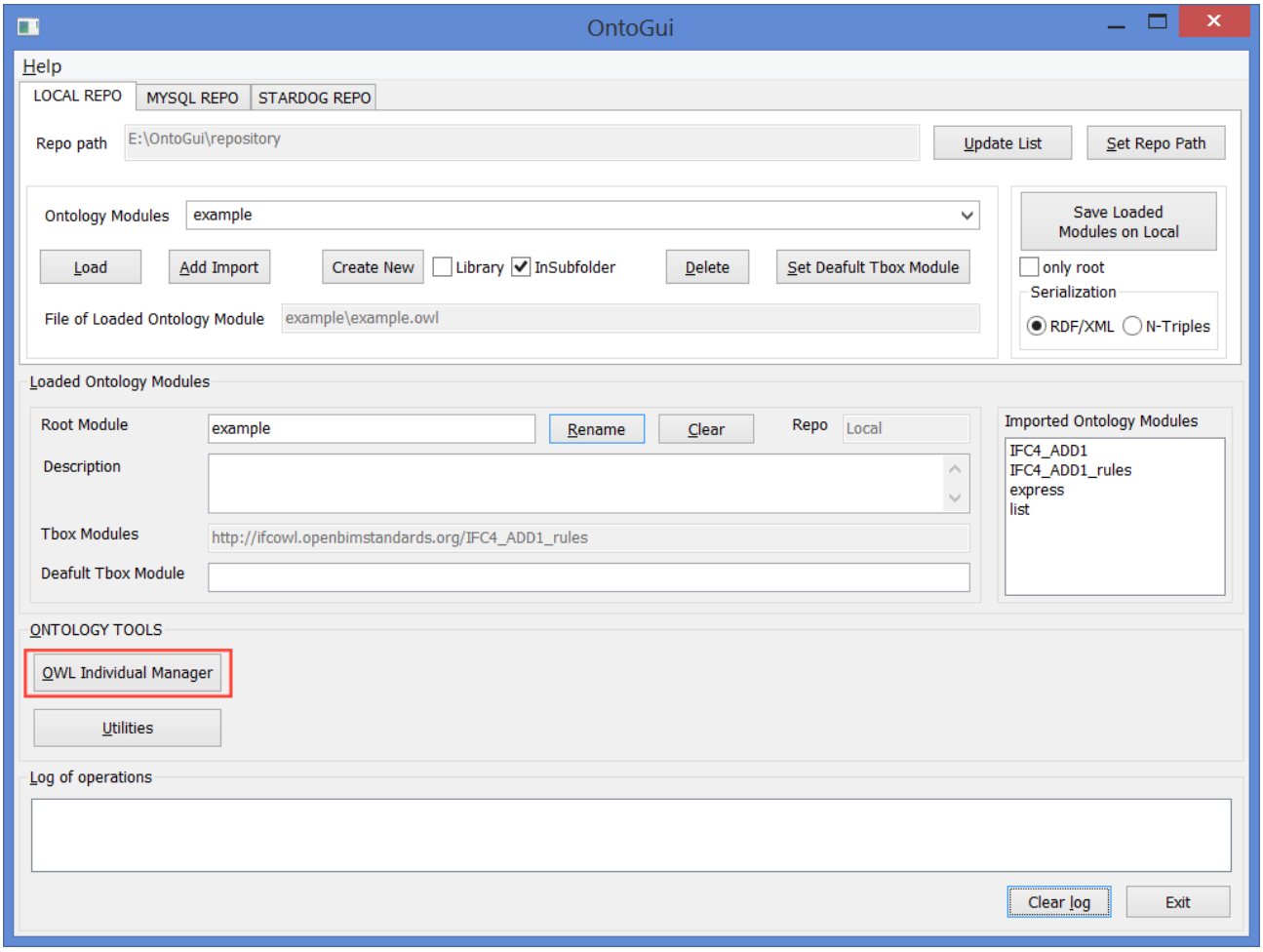
OWL Individual Manager is a general purpose tool for the management of OWL individuals. For information go to the dedicated page.

Individual Manager

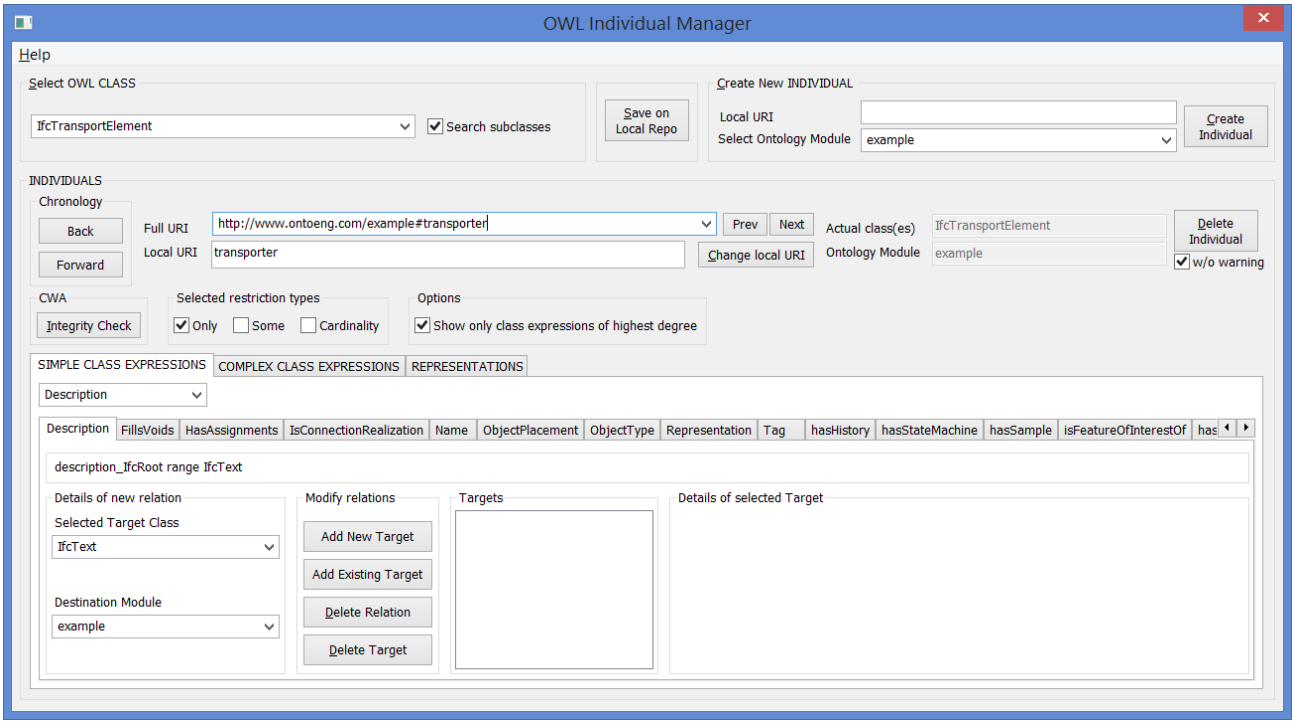
OWL Individual Manager is a general purpose tool for the management of OWL individuals. The main window of the tool is dynamically reconfigured every time an OWL class belonging to the available T-box is selected (in the top left corner). After loading an ontology module in the control panel and selecting an OWL class in OWL Individual Manager, the characterization of the OWL classes provided by the RdfCpp library enables the following functionalities:

- Generation and listing of individuals belonging to the selected class.
- Exploring one individual or generate a new individual belonging to the selected class.
- Listing the properties that can have the selected individual as a subject, based on the T-box axioms. For each property it is possible to visualize the target value that is found at the end of the property chain.
- Exploring and setting the target value of a property for the selected individual.
- Checking the integrity of the selected individual by interpreting the OWL axioms as Integrity Constraints according to the Closed World Assumption (CWA).

Launch OWL Individual Manager:

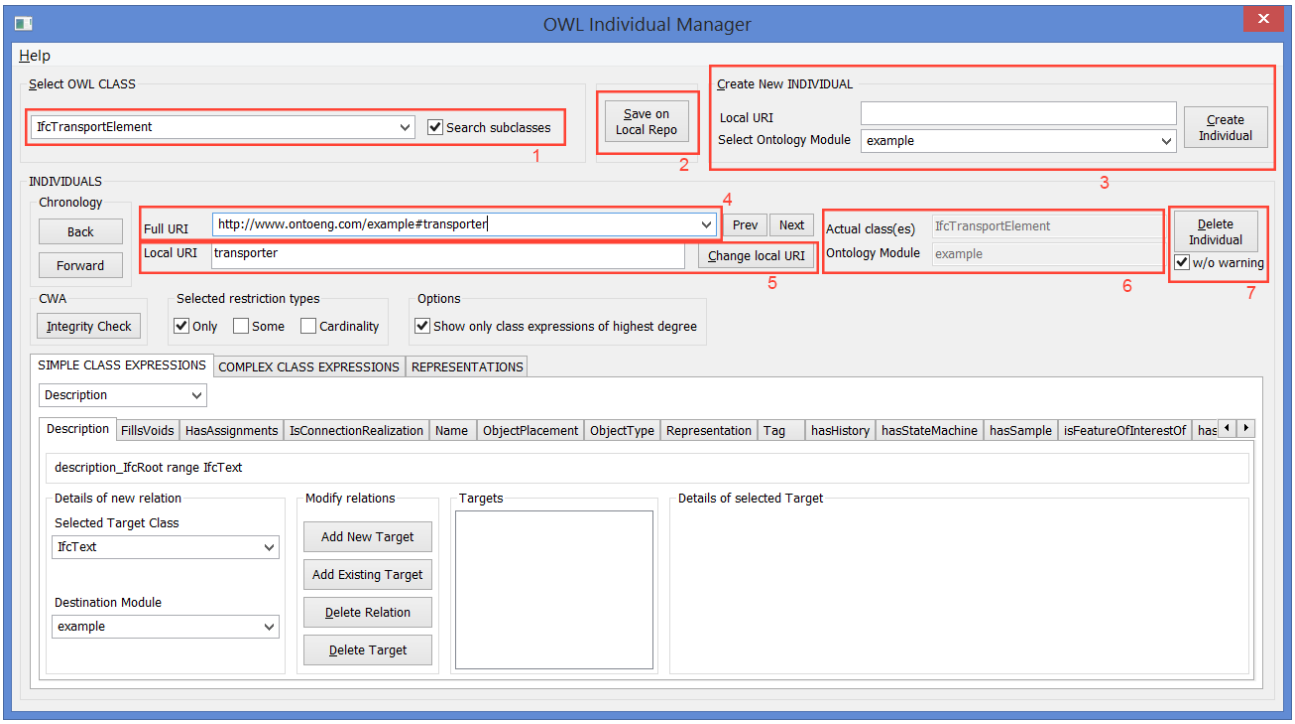


Launch OWL Individual Manager



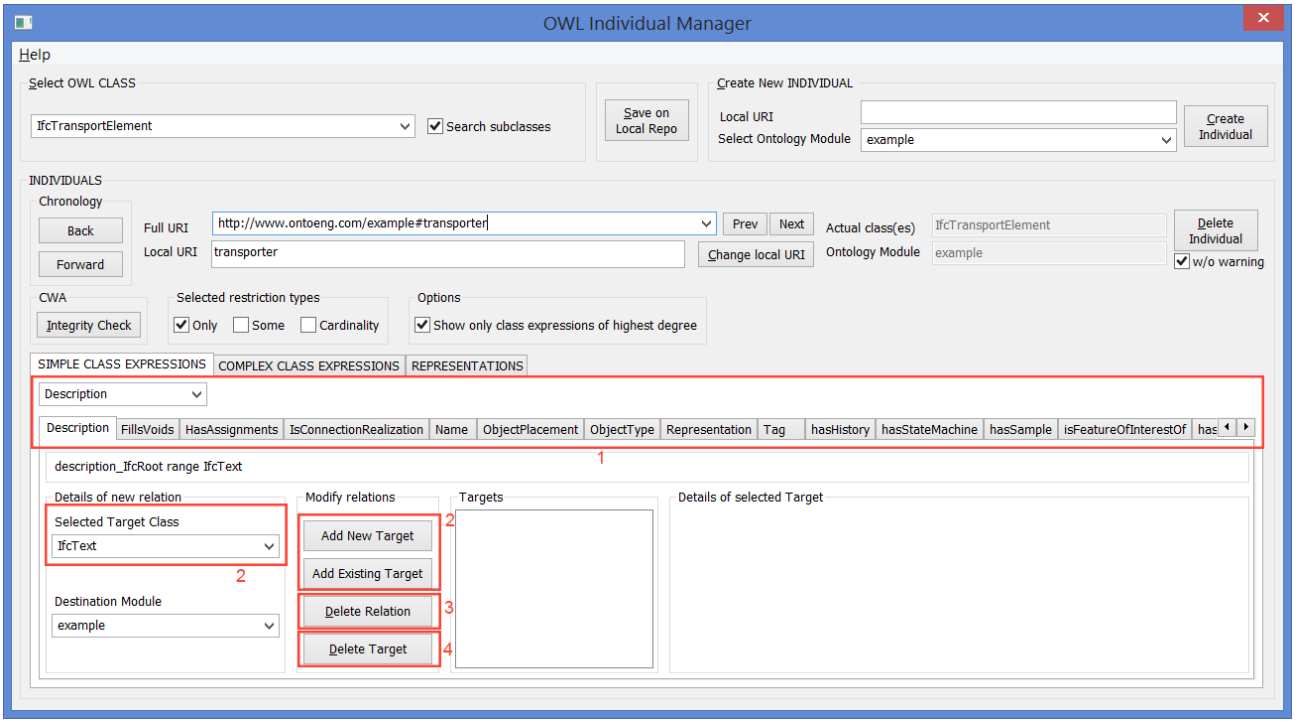
OWL Individual Manager overview

Here's a description of the features and usage of the Individual Manager:



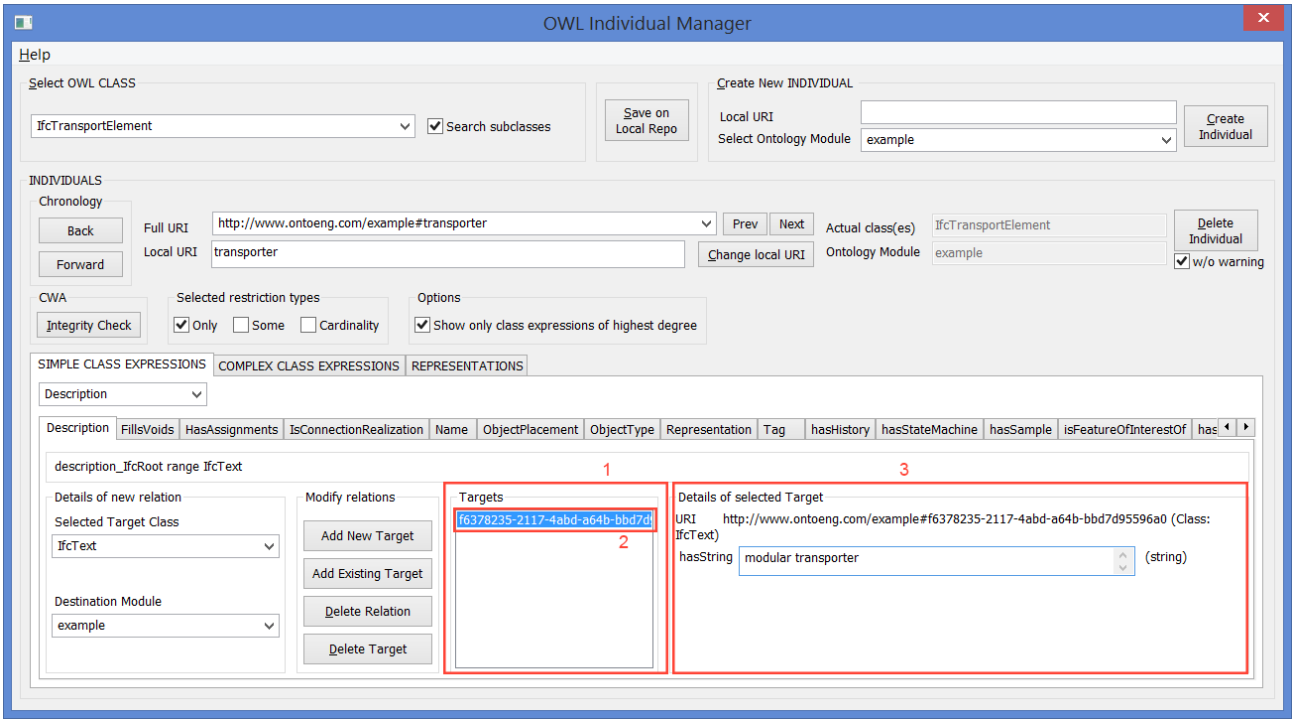
OWL Individual Manager interface - 1

1. List of Selectable OWL classes. The individuals belonging to the selected class (and subclasses if the corresponding box is checked) are searched.
2. If working with a **local repository,** it is necessary to explicitly save if the modifications must be mantained. If working with **Stardog repository** any modification is immediately saved in the repository.
3. Create an individual of the selected OWL class with the specified Local URI in the selected ontology module.
4. List of selectable Individuals belonging to the selected OWL class.
5. Local URI of the selected Individual. This value can be changed.
6. Informations about the selected individual.
7. Delete the selected Individual from the ontology module.



OWL Individual Manager interface - 2

1. Listing the properties that can have the selected individual as a subject, based on the T-box axioms.
2. Add a triple to the ontology module having the selected individual as subject, the currently explored property as predicate and a new or existing individual as target belonging to the selected target class.
3. Delete the relation between the selected individual and the target individual via the currently explored property.
4. Delete the target individual from the ontology module (therefore also the relation with the selected individual is deleted).

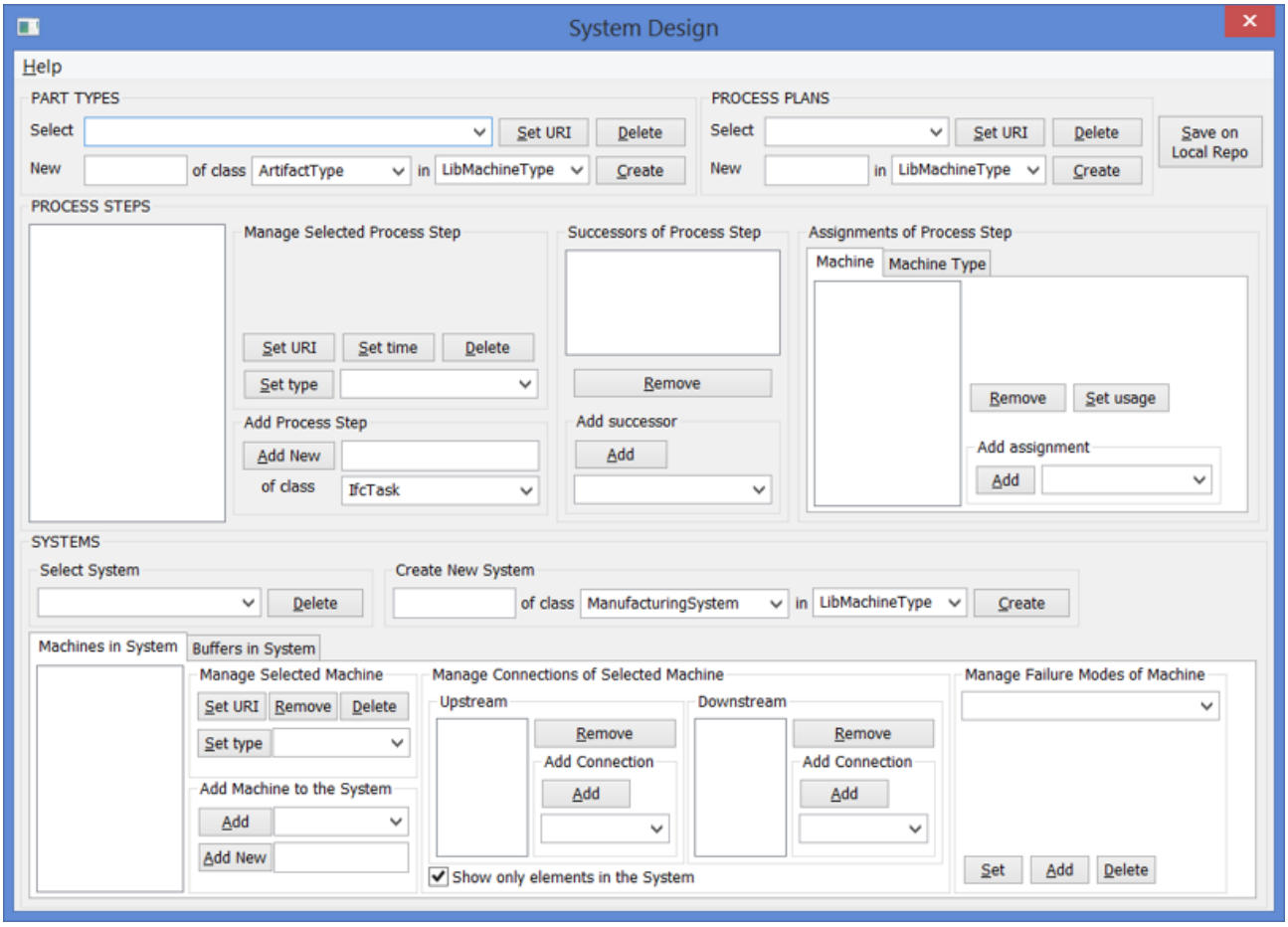


OWL Individual Manager interface - 3

1. List of targets of the selected individual and property.
2. Selected target individual. It can be double-clicked to be explored as new main selected individual.
3. Details of the selected target individual. It is possible to set values if the target individual can be the subject of a relation involving a datatype property.

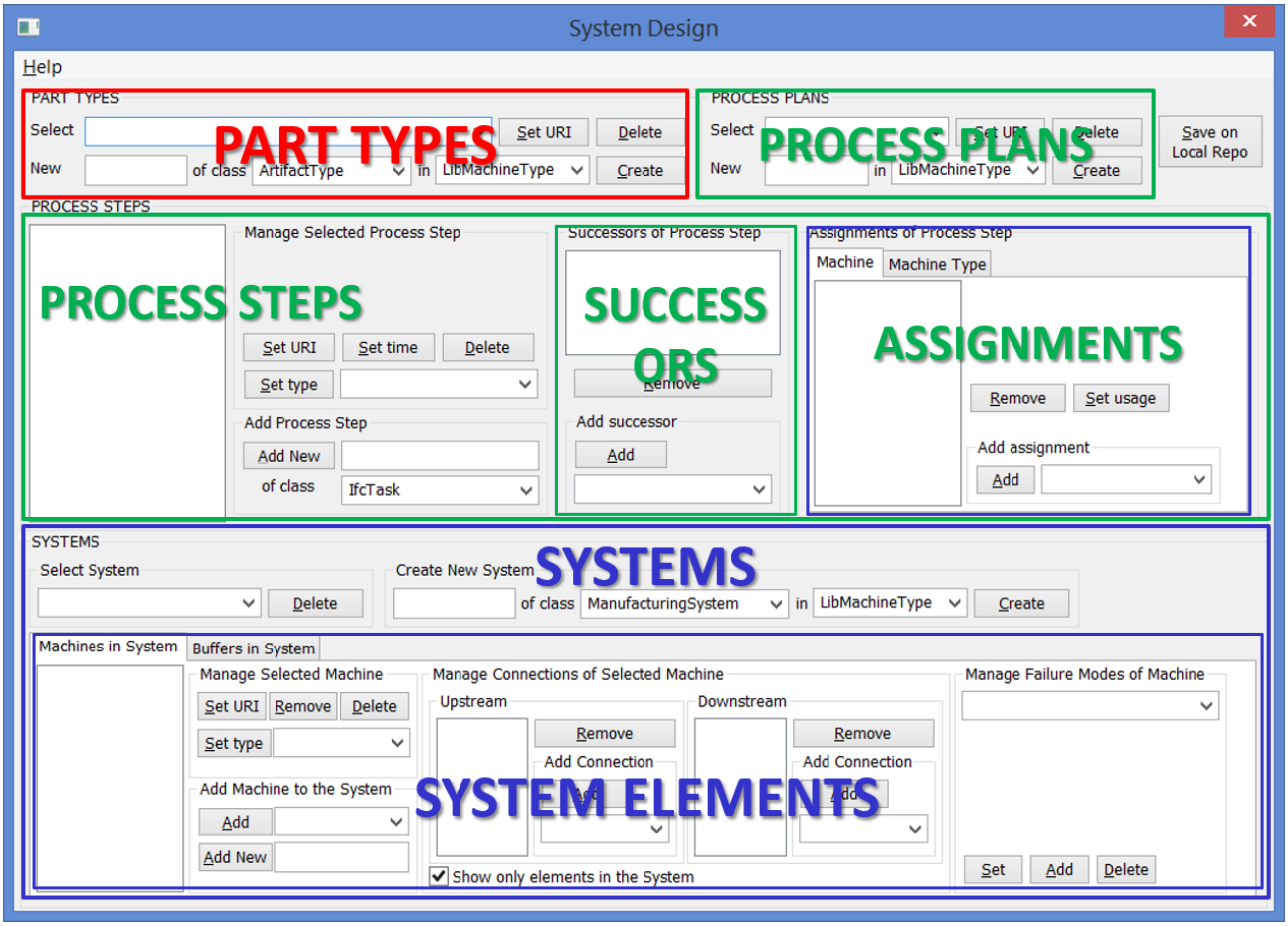
System Design

System Design tool is an OntoGui module that supports the design of a production system and in particular the definition of part types, process plans, and the elements of the system like machines and buffers. The process plans can be decomposed into process step characterized by precedence relations. Moreover, the process steps can be assigned to the elements of the production systems.



System Design graphical user interface

The graphical user interface of System Design can be divided into areas associated with different phases of the configuration of a production system, as shown in the next figure.



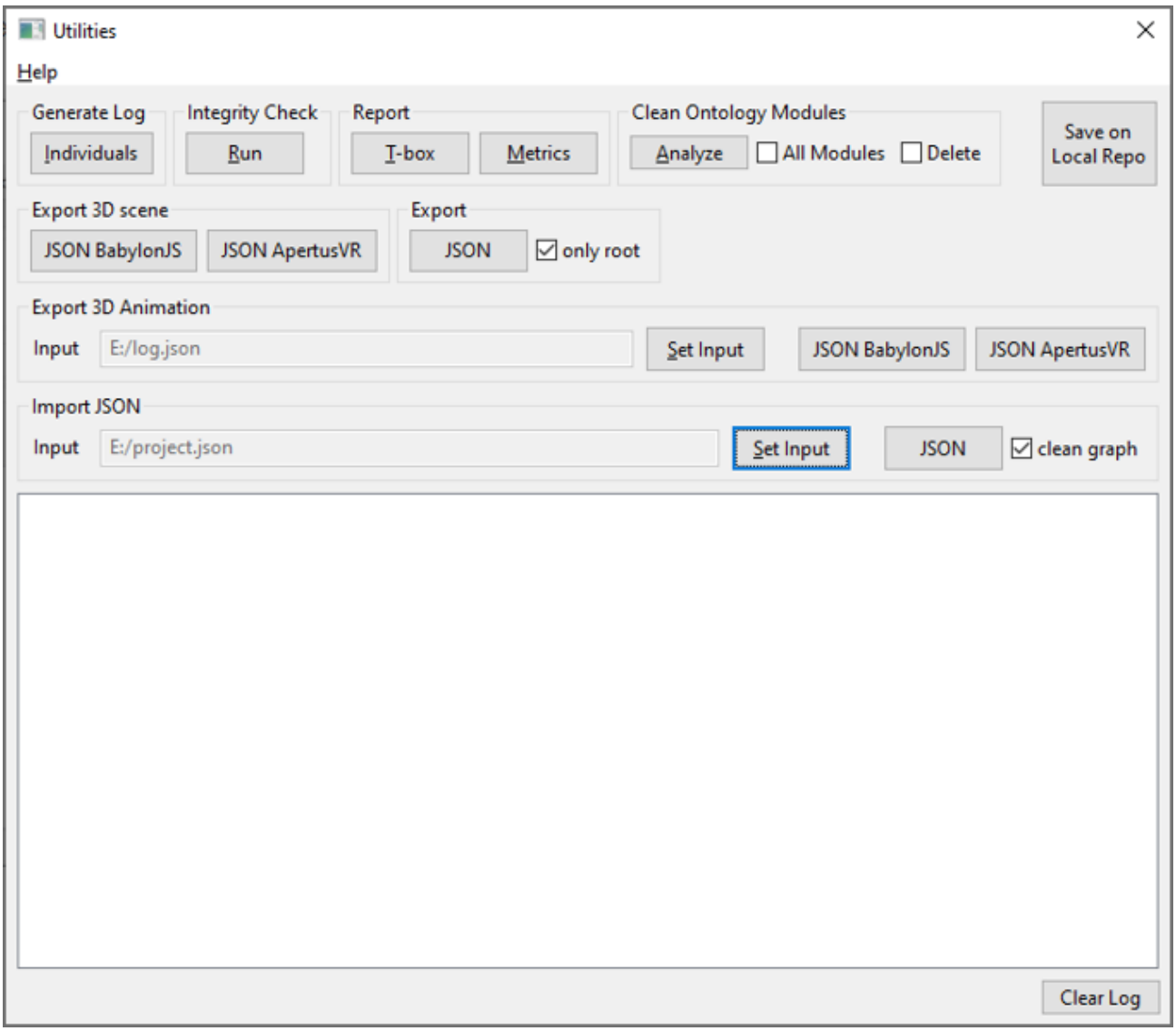
Functionalities of System Design

- **PART TYPES**. Part types can be created, selected or deleted.
- **PROCESS PLANS**. Process plans to produce the currently selected part type can be created, selected or deleted.
- **PROCESS STEPS**. The selected process plan can be decomposed into process steps characterized by processing times.
    - **SUCCESORS**. It is possible to specify which is the successor of the selected process steps, thus defining precedence relations.
    - **ASSIGNMENTS**. The selected process step can be assigned to resources/resource types that are needed to complete/execute the process step.
- **SYSTEMS**. Production systems can be created, selected or deleted.
    - **SYSTEM ELEMENTS**. Production system are defined as an aggregation of elements like machines and buffers. It is also possible to specify the connections between machine tools and buffers. Moreover, machine tools can be characterized by their failure modes and buffers in terms of capacity.

Utilities

Utilities tool provides functionalities to generate reports and exchange files based on the contents of the currently loaded ontology modules:

- **Generate Log**, generating a log about the individuals and related properties.
- **Integrity Check**, checking the integrity of all individuals by interpreting the OWL axioms as Integrity Constraints according to the Closed World Assumption (CWA).
- **Report**:
  - T-box report in terms of classes, object properties, datatype properties.
  - Metrics of ontology modules.
- **Clean Ontology Modules**, cleaning the ontology modules by deleting dangling individuals
- **Export 3D scene**, generating exchange .json file representing a 3D scene for an application based on
  - Babylon.js
  - ApertusVR
- **Export** a .json file representing the contents of the loaded ontology module.
- **Export 3D Animation**, generating exchange .json file representing a 3D animation for an application based on:
  - Babylon.js
  - ApertusVR
- **Import JSON**, importing the contents of a .json file to loaded ontology module.


Utilities
Help
Generate Log
Individuals
Integrity Check
Run
Report
T-box
Metrics
Clean Ontology Modules
Analyze
All Modules
Delete
Save on Local Repo
Export 3D scene
JSON BabylonJS
JSON ApertusVR
Export
JSON
only root
Export 3D Animation
Input
E:/log.json
Set Input
JSON BabylonJS
JSON ApertusVR
Import JSON
Input
E:/project.json
Set Input
JSON
clean graph
Clear Log

Personalization

Configuration file «ConnConfig.xml»

The configuration file ConnConfig.xml can be modified to set the following properties:

- **LocalRepo** to characterize the local repository in the file system with attributes:
  - **path**, i.e. the path of the local repository defined as relative to the root folder of OntoGui
  - **serFormat**, serialization format that can be rdfxml for RDF/XML or ntriples for N-triples
  - **inSubfolder**, value set to «true» if the A-box that are create must be placed in a dedicated subfolder of the local repository, «false» otherwise.
  - **TboxFolder**, subfolder of the local repository dedicated to store T-box ontology modules
- **StardogRepo** to characterize the Stardog repository with attributes:
  - **server**, i.e. address of the Stardog server
  - **database**, i.e. selected database provided by the Stardog server
  - **user**, i.e. user credentials to access the Stardog server
  - **psw**, i.e. password credentials to access the Stardog server
- **RootDataModel**
  - **path**, i.e. default T-box module that is imported by the A-box modules that are created
- **LocalRepoBinary**
  - **path**, i.e. path defined as relative to the root folder of OntoGui that is dedicated to store binary files
- **SceneThreeD:** settings to generate exchange file representing a 3D scene
  - **ModelFiles**, format of 3D models
  - **Quaternion**, true if quaternions are used to define rotations, false if Euler Angles are used
  - **Zup**, true if the Zup convention is adopted, false if Yup is adopted
  - **LengthUnitOfMeas**, unit of measure used for positions in space ("0.01" stands for centimeters)
- **ImportExport**: setting for import/export .json files
  - **InputFile**, input file
  - **OutputDir**, output directory

- **ApertusVR**: settings for export .json file representing a 3D scene for an ApertusVR-based application
  - **ModelFiles**, formats of 3D models that are accepted
  - **OutputFile**, output file
  - **LengthUnitOfMeas**, unit of measure used for positions in space ("0.01" stands for centimeters)
  - **AbsolutePlace**, set to true if absolute placements are used, false if relative placements are used
  - **BlankNode**, set to true if objects without a mesh can be added to the 3D scene
  - **OutputDir**, output directory
  - **AbsPath3Dmodel**, set to true if the path of 3D models is defined in absolute terms
  - **AddModelTypes**, set to true if the model of objects is included in the asset list
  - **ArtifactHierarchy**, set to true if the hierachy of artifact objects is represented in the 3D scene
- **BabylonJS**: settings for export .json file representing a 3D scene for a babylon.js-based application
  - **ModelFiles**, formats of 3D models that are accepted
  - **OutputFile**, output file
  - **LengthUnitOfMeas**, unit of measure used for positions in space ("0.01" stands for centimeters)
  - OutputDir="BabylonJs/Scenes/"
  - **AbsolutePlace**, set to true if absolute placements are used, false if relative placements are used
  - **BlankNode**, set to true if objects without a mesh can be added to the 3D scene
  - **AddModelTypes**, set to true if the model of objects is included in the asset list
  - **ArtifactHierarchy**, set to true if the hierachy of artifact objects is represented in the 3D scene
- **OntoGuiOptions**:
  - **safeDelete**, true if confirmation is asked before deleting an ontology module
- **OntoGuiSysDesignOptions**: settings to customize the interface of System Design module
  - **AssignmentClasses**, list of resource types that can be assigned to process steps
  - **AssignmentLabels**, list of labels for the resource types that can be assigned
  - **ElementClasses**, list of element types that can be included in a production system
  - **ElementLabels**, list of labels for the element types that can be included in a production system
- **Animation**: settings for the generation of 3D animations

- **InputLogFile**, input log file to generate the animation.
- **InputClipFile**, input .json file defining animation clips.
- **ReplicatesN**, number of replicates of the animation sequence.
- **ReplicatesLag**, lag between replicates.

3.2 jsimIO

jsimIO is a platform-independent Python library, compatible with **Python > 3.x**, which can be used to generate JSIM simulation models of a manufacturing system and run the simulation from a Python environment using the Java Modelling Tools (JMT) simulation engine.

GitHub repository: https://github.com/frabera/jsimIO

Features

jsimIO is able to generate simulation models for manufacturing systems with the following characterstics:

- **Multiple part-types**
- **Open** and **closed part-type classes**
- **Assembly operations** (sub-assembly part types can be joined to a assembled part type)
- **Multiple drop** and scheduling strategies
- **Multiple statistical distribution** for modelling service times and arrival times.

> For a detailed description, refer to the jsimIO Manual in the dedicated GitHub repository.

The available nodes to model the system are:

- **Source**
- **Sink**
- **Station** composed by:
  - **Queue** of finite or infinite size
  - **Server** with single or finite number of concurrent jobs
- **Infinite Server** (Delay)

- **Fork**
- **Join**
- **Logger**

Measures

Many different measures can be selected as simulations result, such as the throughput or the average number of customers, both referred to a single node or to the whole system. *For further information please refer to the JMT manual.*

Simulation Log

Using the Logger station, it is possible to save a log of each event relate to part flowing by the station.

jsimIO can automatically add a preceding Logger station to each node of the system. In this way, at the end of the simulation a log file is saved as a *.csv* file containing all the events and part flows in the system for further analysis.

Installation and Requirements

For installation instructions and requirements, visit the dedicated page:

→ How to start /tools/jsimio/how-to-start

How to start

Hardware, OS, browser compatibility

jsimIO is a platform-independent Python library, compatible with Python > 3.x.

Java Modelling Tools is platform-independent and requires only the Java Oracle JDK.

jsimIO installation instruction are also reported in the README.md file in the GitHub repository:

https://github.com/frabera/jsimIO/blob/master/README.md

Testing Platforms

The library has been succesfully tested with:

- Microsoft Windows XP/VISTA/7/8/10
- Mac OS X 10.4.6+
- Linux Ubuntu 16.10+

Requirements

JMT - Java Modelling Tools require the installation of Oracle/Sun Java JDK. To download it, visit https://www.oracle.com/java/technologies/javase-downloads.html. The required package is called "Oracle JDK".

Select your O.S., accept the Terms and Conditions and download and install the application.

Download jsimIO

To download jsimIO, download or clone the jsimIO repository:

https://github.com/frabera/jsimIO

Download JMT

> The required file to download is the Java .jar executable, not the JMT installer.

Current Stable JMT Version: 1.1.1

JAR Version Download it here

After downloading the executable, rename it to `JMT.jar` and place it in the jsimIO subfolder, where `JMT.jar_GOES_HERE` file is located.

License

See license informations in the License page.

JMT Overview

Introduction

Java Modelling Tools (JMT) is a suite of applications developed by Politecnico di Milano and Imperial College London and released under GPL license.

The project aims at offering a comprehensive framework for performance evaluation, system modeling with analytical and simulation techniques, capacity planning and workload characterization studies.

The suite includes six Java applications:

1. ~~JSIM*graph*~~ - Queueing network and Petri net simulator with graphical user interface
2. ~~JSIM*wiz*~~ - Queueing network and Petri net simulator with wizard-based user interface
3. ~~JMVA~~ - Mean Value Analysis and Approximate solution algorithms for queueing network models
4. ~~JABA~~ - Asymptotic Analysis and bottlenecks identification of queueing network models
5. ~~JWAT~~ - Workload characterization from log data
6. ~~JMCH~~ - Markov chain simulator

Versions

Download: Latest Release

Manual: Download

References

M.Bertoli, G.Casale, G.Serazzi.
JMT: performance engineering tools for system modeling.
ACM SIGMETRICS Performance Evaluation Review, Volume 36 Issue 4, New York, US, March 2009, 10-15, ACM press. (Article) (BibTex)

JSim

In the JMT suite a discrete-event simulator for the analysis of queueing network models is provided. Two user interfaces are available: alphanumerical (**JSIMwiz**) and graphical (**JSIMgraph**).

JSIMgraph is the GUI front-end to JMT simulation engine. It helps the users to perform an evaluation study in two ways. Firstly, critical statistical decisions, such as transient detection and removal, variance estimation, and simulation length control, have been completely automated, thus freeing the users from taking decisions about parameters s/he may not be familiar with. The simulation is automatically stopped when all performance indexes can be estimated with the required accuracy. Secondly, a user-friendly graphical interface allows the user to describe both the network layout and the input parameters. Furthermore, the graphical interface also provides support for the use of advanced features (several of them are for networks with very general characteristics, usually referred to as non-product-form networks) like fork and join of customers, blocking mechanisms, regions with capacity constraints on population, state-dependent routing strategies, user-defined general distributions, import and reuse of log data. A module for What-If Analysis, where a sequence of simulations is run for different values of control parameters, particularly useful in capacity planning, tuning and optimization studies, is also provided.

The simulation engine performs on-line the statistical analysis of measured performance indices, plots the collected values, discards the initial transient periods and computes the confidence intervals. Network topologies implemented and solved using JSIMgraph can be exported in vector (e.g., eps, pdf) or raster (e.g., jpg, png) image formats.

From: http://jmt.sourceforge.net/Papers/JMT_users_Manual.pdf

Following, the workflow for modelling and simulating a production system in Jsim is presented.

Sections:

→ Model generation
/tools/jsimio/jmt-overview/jsim/model-generation

→ Launch of the simulation
/tools/jsimio/jmt-overview/jsim/launch-of-the-simulation

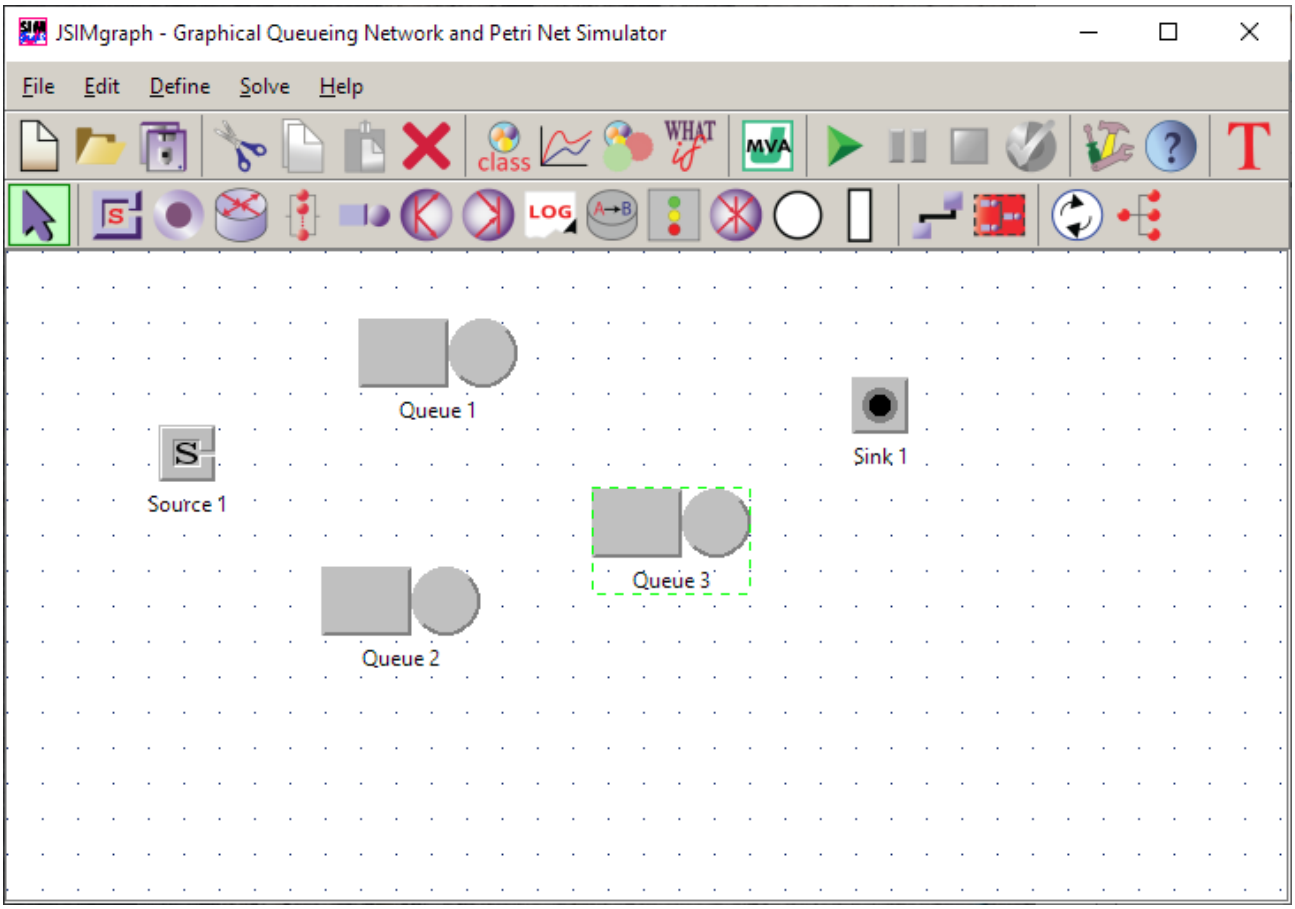
→ Reporting
/tools/jsimio/jmt-overview/jsim/reporting

Model generation

The required steps to generate the JSim model are the following:

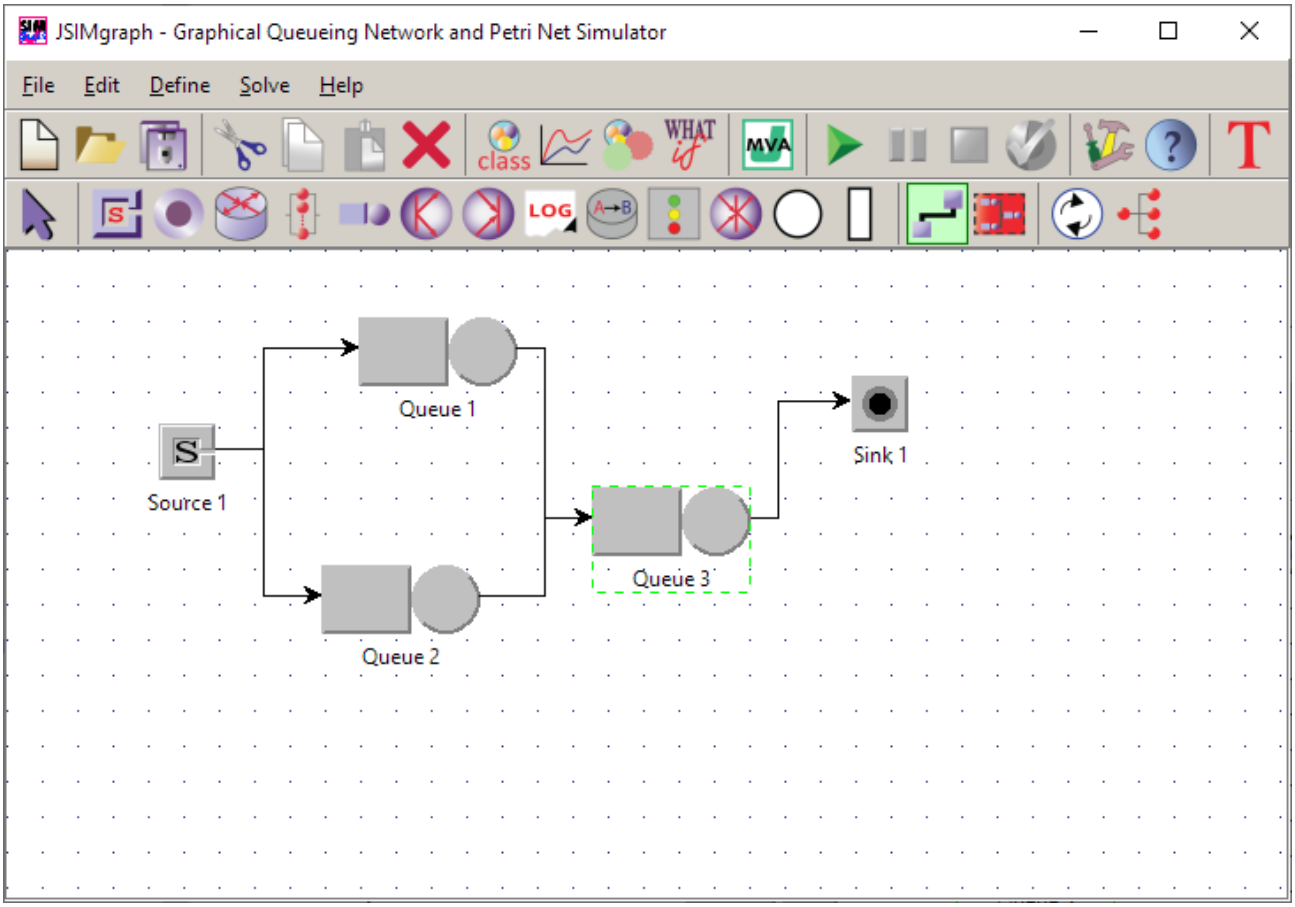
Draw the network

To draw the network, use the cursor to choose the node types from the toolbar and insert them by clicking in the canvas in any area. It is then possible to simply drag the icon to move the node.



Nodes insertion

After all the required nodes are included, it is necessary to draw the connection between them. The drawing connections mode is activate by clicking on the highlighted icon (see figure). To connect two nodes simply click on the first node and drag it. An arrow will appear. Move to the following node and release, the two nodes are now connected.



Nodes connection

Customer Classes

After the model draw, it is necessary to define the customer classes that will be in the system and to define a *Reference Station* for each one of them.

Bibliography

Manuals & Books

- Java Modelling Tools - *User manual* v1.0.4, 227 pages, 4 September 2019. (Manual)
- G.Serazzi Ed. *Performance Evaluation Modelling with JMT: learning by examples* Politecnico di Milano - DEI, TR 2008.09, 366 pp., June 2008 (Book)

Tutorials

- Video lectures: introduction to queueing theory, JSIMgraph and JMVA.
- Video demos: short videos illustrating key features of JSIMgraph and JMVA.
- G.Serazzi. *Advanced features for Multi-formalism modelling with Java Modelling Tools*. EPEW Keynote, November 2019, Milan, Italy. (Slides - PDF)
- G.Casale, G.Serazzi, L.Zhu. *Performance Evaluation with Java Modelling Tools: a Hands-On Introduction*. IFIP Performance Conference, November 2017, NYC, US. (Slides - PPT) (Slides - PDF) (Examples)
- G.Casale, G.Serazzi. *Quantitative System Evaluation with Java Modelling Tools*. 2nd ACM/SPEC International Conference on Performance Engineering (ICPE), March 2011, Karlsruhe, Germany. (Slides - PPT) (Slides - PDF)

Papers

- M.Bertoli, G.Casale, G.Serazzi. *JMT: performance engineering tools for system modeling*. ACM SIGMETRICS Performance Evaluation Review, Volume 36 Issue 4, New York, US, March 2009, 10-15, ACM press. (Article) (BibTex)
- G. Casale, M. Cazzoli, S. Jiang, V. S. Lopes, G. Serazzi, L. Zhu. *Generalized Synchronizations and Capacity Constraints for Java Modelling Tools*. ACM/SPEC ICPE 2017, 169-170, ACM press. (Article) (BibTex)
- M.Bertoli, G.Casale, G.Serazzi. *User-Friendly Approach to Capacity Planning Studies with Java Modelling Tools*. Int.l ICST Conf. on Simulation Tools and Techniques, SIMUTools 2009, Rome, Italy, 2009, ACM press. (Article) (Slides) (BibTex)

- M.Bertoli, G.Casale, G.Serazzi. *The JMT Simulator for Performance Evaluation of Non-Product-Form Queueing Networks*. SCS Annual Simulation Symposium 2007, Norfolk,VA, US, 3-10, IEEE Press. (Article) (Slides) (BibTex)
- M.Bertoli, G.Casale, G.Serazzi. *An Overview of the JMT Queueing Network Simulator*. Politecnico di Milano - DEI, TR 2007.2, 2007. (Article) (BibTex)
- M.Bertoli, G.Casale, G.Serazzi. *Java Modelling Tools: an Open Source Suite for Queueing Network Modelling and Workload Analysis*. Proceedings of QEST 2006 Conference, Riverside, US, Sep 2006, 119-120, IEEE Press. (Article) (Slides) (BibTex)

Designed by Bertoli Marco

DEIB -Politecnico di Milano - Italy

Department of Computing - UK

Coordinators: G.Casale, G.Serazzi

3.3 VEB.js

VEB.js (**V**irtual **E**nvironment based on **B**abylon.**js**) is a reconfigurable model-driven virtual environment application based on Babylon.js, a complete JavaScript framework and graphics engine for building 3D applications with HTML5 and WebGL (Web Graphics Library). Babylon.js enables to load and draw 3D objects, manage these 3D objects, create and manage special effects, play and manage spatialized sounds, create gameplays and more. Babylon.js library is free to use, thus enabling also didactic use.

Babylon.js was exploited to develop **VEB.js** as a reconfigurable model-driven virtual environment application.

Hardware, OS, browser compatibility

VEB.js works on any browser that supports WebGL and no specific configuration of the hardware is needed. The same can be stated for the Operating System (OS).

Though it has to be noted that depending on the complexity of the scene, problems (e.g. lag) might arise if using a low potential graphic processing unit (GPU); indeed, even a smartphone can be used to load basic scenes, but if the complexity of the scene increases a more powerful GPU will be needed in order to obtain optimal rendering.

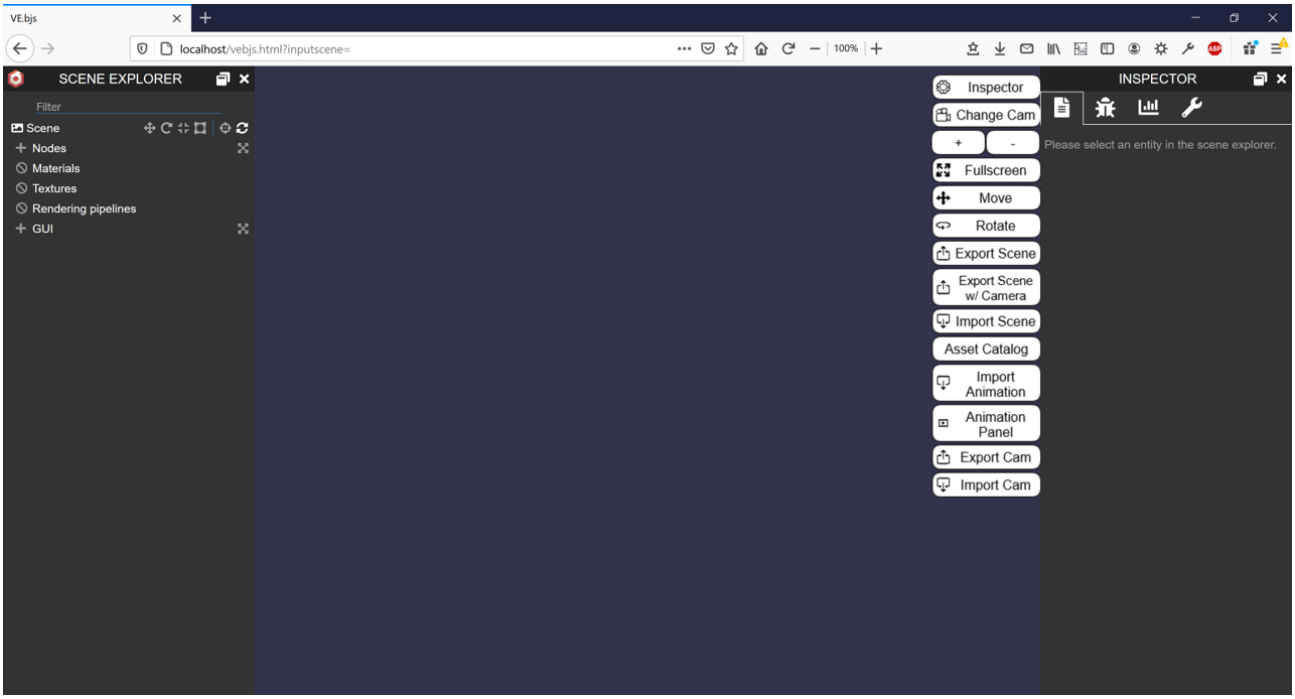
How to run VEB.js remotely

It is possible to remotely access the VEB.js application if it is installed on a server. If the server is active at the address `$ServerAddress`, then the application can be accessed via browser at `$ServerAddress/vebjs.html?inputscene=`.

A demo of VEB.js is available at: http://mi-eva-d001.stiima.cnr.it/vebjs/?inputscene=

Getting started

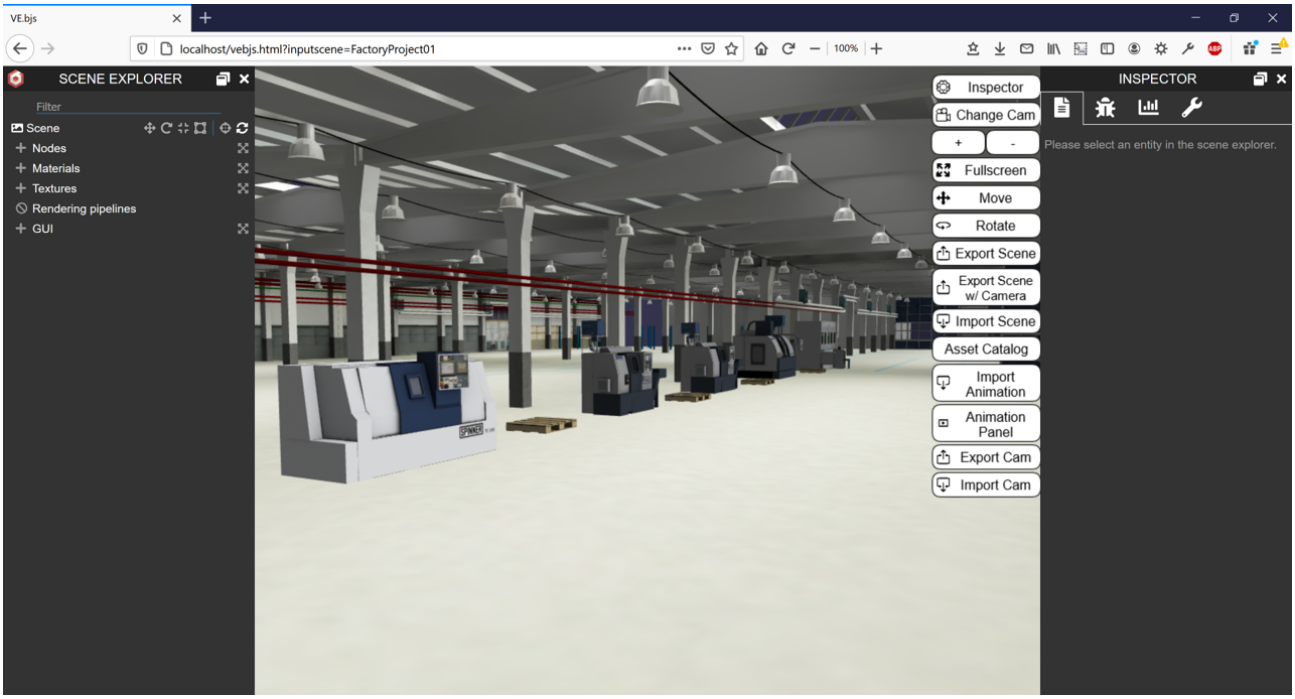
At startup the browser shows an empty scene that contains the main application functionalities.



empty scene

A **scene** consists of a set of assets (e.g. selected from a catalog of resources) that are organized in a layout; in addition, lights and cameras (i.e. navigation point of view) are attached to a scene. After opening the empty scene page, there are two options to load an already existing scene:

- import a scene .json file, clicking on the **Import** button and browsing your local file system.
- modify the URL writing after "=" the name of the scene to be opened, e.g. http://mi-eva-d001.stiima.cnr.it/vebjs/?inputscene=example_1. The corresponding .json file must be available in the `Scenes` folder of the server where VEB.js is running.
- modify the URL writing after "=" the address of scene .json file that is available on a remote repository, e.g. http://mi-eva-d001.stiima.cnr.it/vebjs/?inputscene=https://raw.githubusercontent.com/wterkaj/RepoExample/main/example_1/example_1.json



Example of a 3D scene

References

- Compatibility https://doc.babylonjs.com/

Basic functionalities

1. Moving the camera in the scene

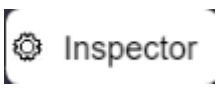
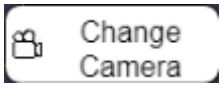
It is possible to use the keyboard to navigate the scenes with the following keys.

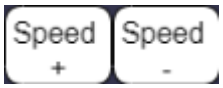
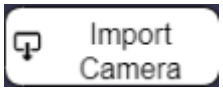
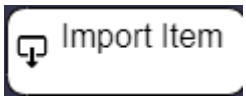
Key	Function
A	moves the point of view to the left
D	moves the point of view to the right
W	moves the point of view forward (zoom)
S	moves the point of view backward (de-zoom)
E	moves the point of view upward
Q	moves the point of view downward

Moreover, it is possible to select an asset by clicking on it; the selection will change to red the color. A selected asset can be deleted from the scene by pressing key "Del"; however, it must be stressed that a deletion command cannot be undone. The selection of components in an assembled asset is possible using the *Scene Explorer*.

2. Buttons in the Working environment: functionalities and mode of use

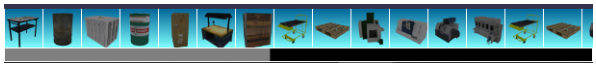
Various buttons appear as soon as the application is launched (also in case of empty scene). The following table describes each button and the associated functionality.

Button	Function	Use mode
	Open/close the side bars (BabylonJS default function)	Click
	Select a saved point of view (i.e. camera)	Change the view with one click

	Increase/decrease movement speed	Each click on the button corresponds to the increasing/decreasing of speed x2
	Full screen	Click to undo (same result with *esc* key)
	Move the selected asset	Once clicked, select the asset and move it using the translation widget that will be shown
	Rotate the selected asset	Once clicked, select the asset assembly and rotate it using the rotation widget
	Save the scene as a .json file	Click
	Save the scene as a .json file, including also the current cameras (points of view)	Click
	Save the current cameras (points of view) as a .json file	Click
	Import a scene from from a .json file	Click and browse a .json file
	Import a camera (point of view)	Click and browse browse a .json file
	Opens a catalogue GUI containing all the assets that can be added to the scene	Click

3. Catalog GUI

The catalog GUI shows the assets that can be added to the scene, as shown in the following image.

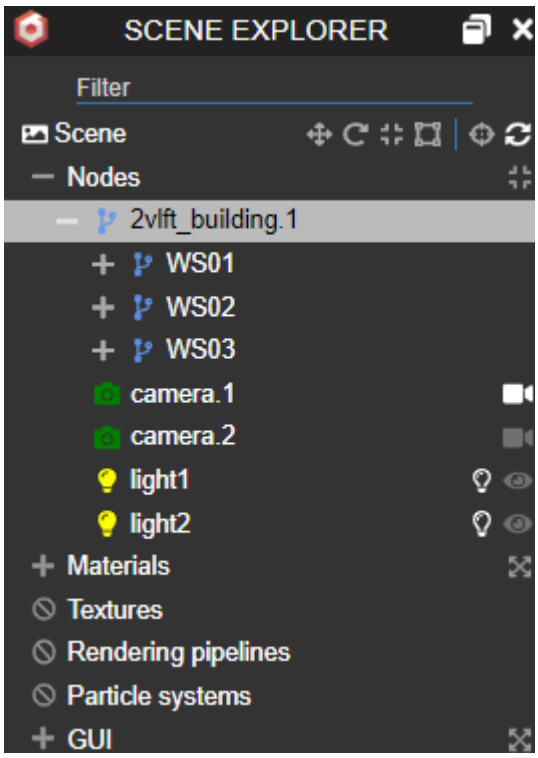
 Catalog *GUI*

An asset can be added to the scene by clicking on the corresponding icon. After the 3D representation of the asset is generated, it is possible to move/rotate the asset in the scene.

4. Scene Explorer and Inspector

The *Inspector* button show/hide two sidebars on the working environment page: Scene Explorer (left sidebar) and Inspector (right sidebar).

NOTE	On the top of both sidebars there is a command to detach the sidebar from the scene. This command might be useful in case of a large scene to visualize it on the whole screen.



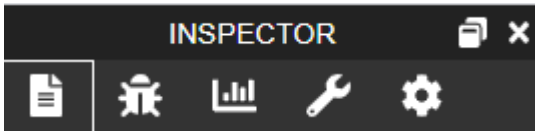
empty scene

In the *Scene Explorer* bar it is possible to click on the "+" on the side of *Nodes* to show all the items composing the scene in terms of: assets (and their components), cameras and lights.

It is also possible to select an asset group or a single asset belonging to an assembly.

Materials and *Textures* can be explored to retrieve further information about the assets and their 3D representation. Finally, *GUI* shows the list of items added to the user interface, e.g. buttons, that can be shown/hidden.

The *Inspector*" bar consists of the main tabs described in the following subsections.



empty scene

4.1 Properties

Properties are activated after selecting an asset, camera, light, or material. Some properties can be manually changed.

In the case of an *asset*, properties include position, scale and rotation.

In the case of a *camera*, properties include: position, speed, mode, etc.

In the case of a *light*, properties include: the intensity of the light, the set-up, and different setting on the shadows generated by the light.

4.2 Debugging

The second icon of the *Inspector* allows to enter the area relative to debugging operations.

4.3 Statistics

The third icon is related to the statistics (quantitative properties) of the scene:

- FPS (frame per second) at which the scene is running.
- COUNT describing the graphical properties of the scene.
- FRAME STEPS DURATION showing the dynamical properties of the scene
- SYSTEM INFO displaying information related to the hardware and software system, in therms of graphical and operating system properties.

4.4 Tools

The fourth icon opens a window showing a set of tools:

- "*Screenshot*" to capture the whole current scene in an image that is saved automatically to the *Download* folder in a *.png* file.
- "*Record video*" to start recording the current scene (animation) and then to stop recording. The video is saved in the *Download* folder in a *.webm* file.
- "*Capture*" is equivalent to the *Screenshot* functionality, though it allows to take a screenshot of the scene without any button or other GUI feature. It also allows to set the *Precision* of the capture or to set its width and height.
- "*Generate replay code*" to download the code of the scene in a *.txt* format.
- "*Export to GLB*" to save the 3D model of the scene in a *.glb* file.
- "*Export to Babylon*" to save the scene in a *.babylon* file.

Input/Output files

Scenes can be imported/exported from/to a **.json** file according to a specific schema.

The animation of the scene can be defined in another **.json** file together with additional data (e.g. reusable animation sequences) stored as **.txt** files.

Given a scene file named `scene.json`, the convention is that the corresponding animation file is named `scene_anim.json`.

Examples of input files are provided in the Use Case section.

Integration with other software tools

Third party tools can be exploited to automatically generate a scene in .json format that can be imported by the babylon.js Virtual Factoy application. For instance OntoGui provides provides such functionalities in the Utilities module.

Advanced Users

Further functionalities of VEB.js can be developed by modifying HTML and JavaScript code that is available in the folder named `$vebjs` . This folder may be provided as as a single *.zip* file to be unzipped.

Node.js and NPM are needed to run the application. If Node.js and NPM are not already installed, it is possible to download and run the latest (LTS) Node.js installer confirming the default settings . NPM (Node Package Manager) is a package manager for JavaScript language and is included in this installation. For installing the dependencies, open the command prompt, go to directory `$vebjs/js` and execute command `npm install`

It is important to keep the original structure of the `$vebjs` folder to guarantee a correct execution of the application, including the access to already available scenes in subfolder `$vebjs/Scenes` .

To easily access local files (e.g. 3D models) it is necessary to launch a local host server, as explained in the following for Windows and MacOS.

Run on Windows: launch the application by double-clicking `vebjs_launch.bat` in the `$vebjs` folder and a browser page will be opened automatically at the address: `http://localhost/vebjs.html?inputscene=`

Run on MacOS: first open the terminal and go to the folder `$vebjs.` Then run the command `sh vebjs_launch.sh` and a browser page will be opened automatically at the address: `http://localhost/vebjs.html?inputscene=`

Offline/Online use

The default setting of the package allows to run the application offline by exploiting the libraries (modules) installed in `$vebjs/node_modules` . However, if an internet connection is available, the most recent version of the libraries can be accessed by manually modifying the file `$vebjs/vebjs.html` ; in this case it is necessary to comment the whole "LOCAL" block and uncomment the "ONLINE STABLE" block in the `<head>` .

VEB.js as a Tomcat service

The contents of the folder `$vebjs` can be compressed in a .war file and the application launched as a Tomcat service.

3.4 ApertusVR

ApertusVR is an open source software library (available on GitHub) to develop AR/VR applications for science, education, and industry.

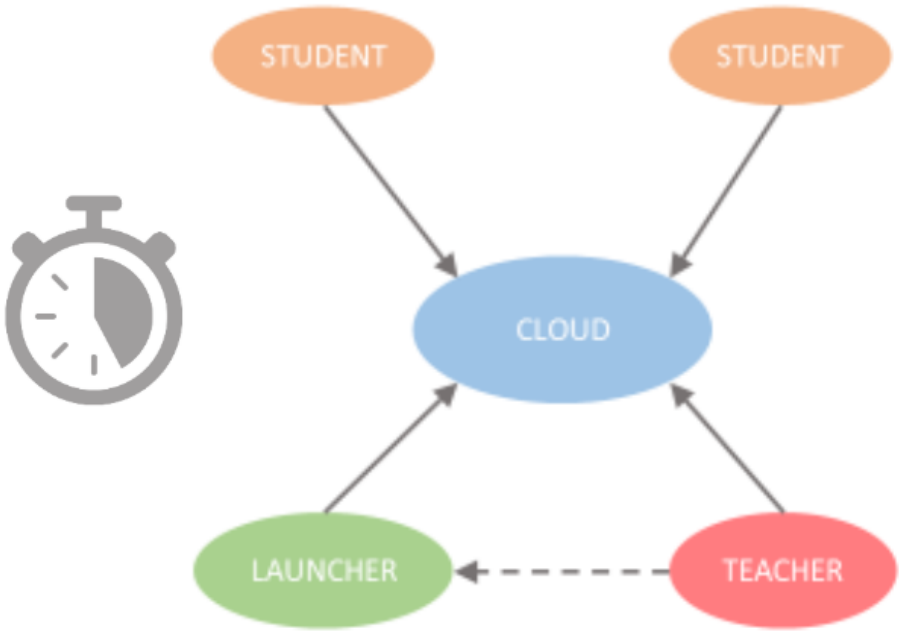
The general documentation of ApertusVR can be found here.

A Gamification application based on ApertusVR libraries was developed to support teaching and training in a VR environment. The following pages present the two versions of the the VLFT Gamification App (Version 1 and Version 2). More details can be found here.

OntoGui can be exploited to generate input files for the VLFT Gamification application.

Virtual Learning Factory Toolkit Gamification - Version 1

The aim of VLFT Gamification is exploiting the advances and technologies of modern games to provide students with a realistic representation of a real manufacturing system, making use of VR visualization experiences.
Students will learn how to analyse the configuration of a manufacturing systems in terms of:
• products manufactured and the associated process;
• machines, associated capability and processing times;
• occurrence of failures and their statistics;
• identify the bottlenecks in the system. In particular, the students could enter the VR environment of a manufacturing systems where they will find its VR representation, i.e., a production line.

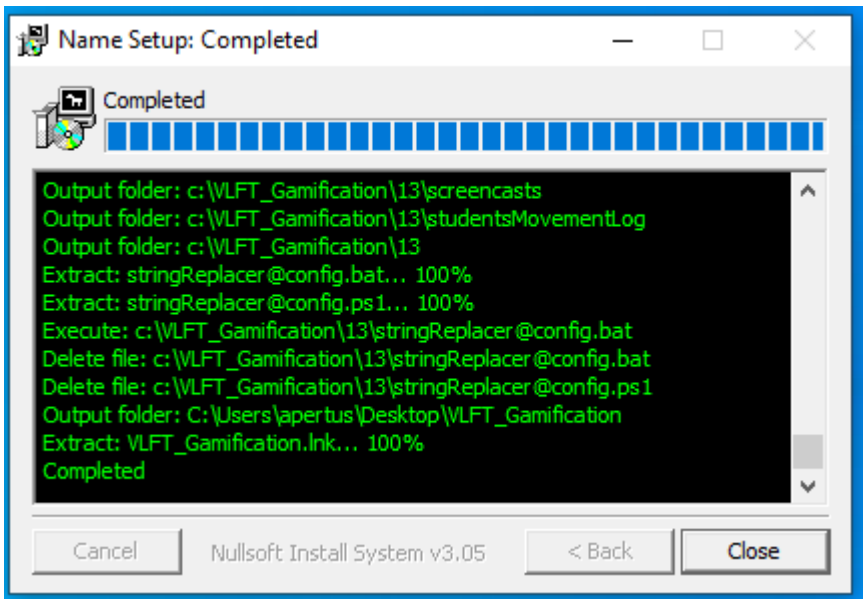


Workflow of a VLFT Gamification Session:
1: The teacher starts a Gamification Session on the cloud server
2: Students can connect to the desired Gamification Session and sharing the same VR visualization of the manufacturing system.

3: Interactions between the participants can be made via the specific Student User Interface and Teacher User Interface

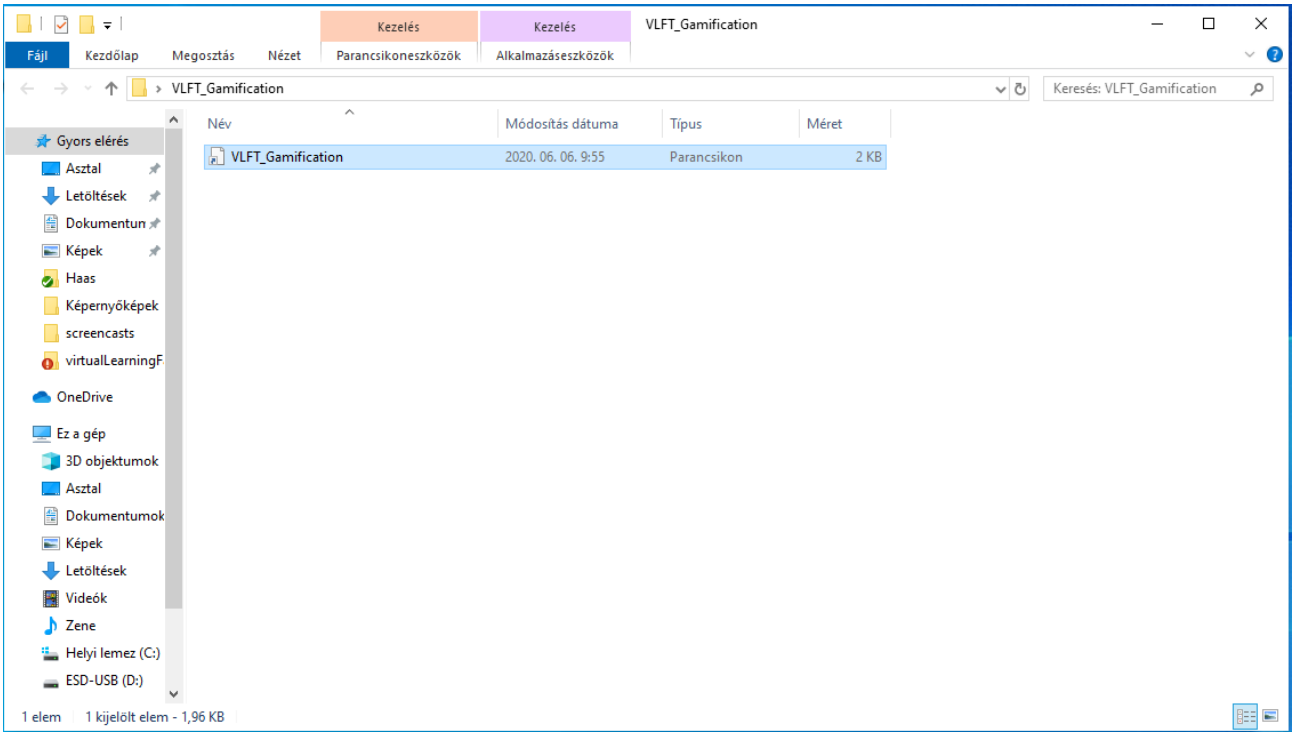
Installation

The latest version of the VLFT Gamification can be downloaded here



VLFT Gamification Installer

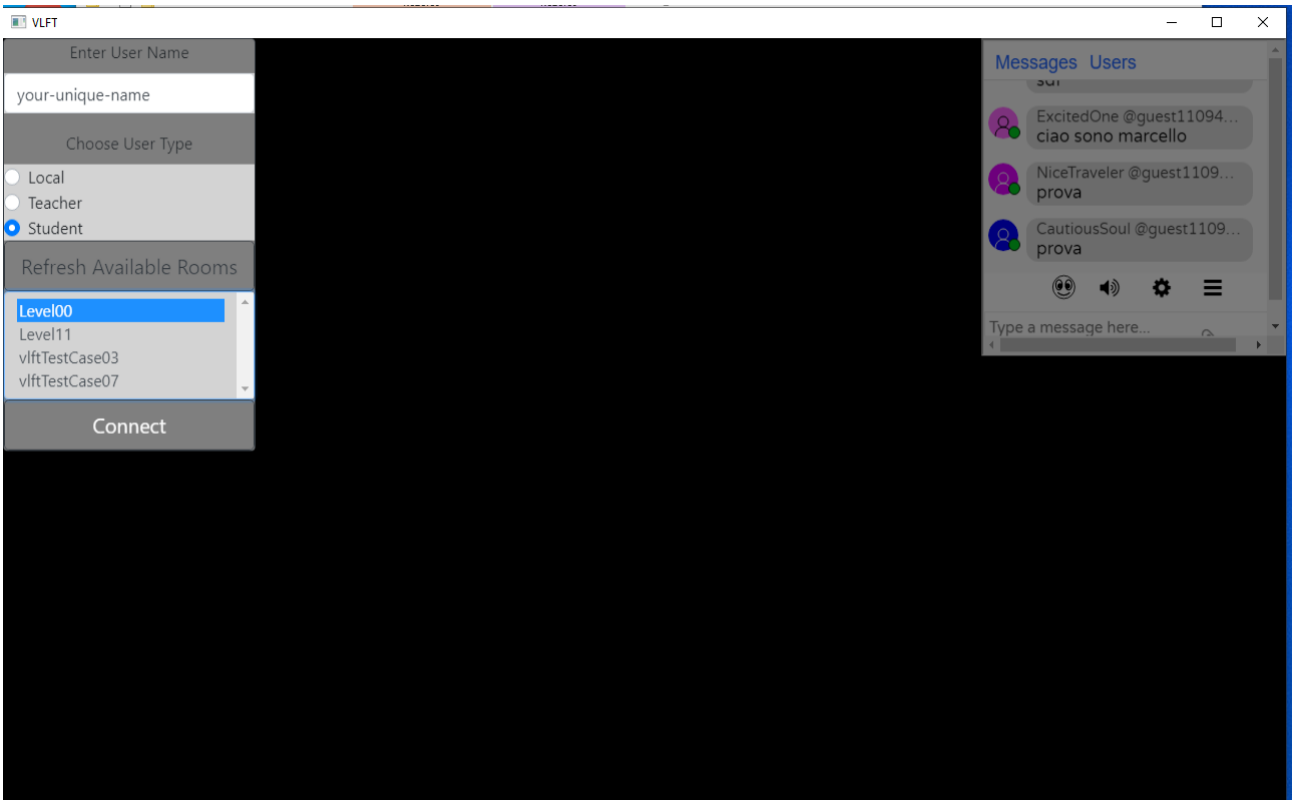
After the installation was completed a folder named VLFT_Gamification can be founded in the desktop.



A shortcut for the VLFT Gamification application can be founded. After its clicked then a Lobby User Interface should be displayed

Lobby - User Interface

After the VLFT Gamification Application started the Lobby GUI will appear.



Lobby GUI

On the right side of the screen the chat window can be used to send a massage to the other on-line participants in the Lobby.

The next steps are describe how to join a VLFT Gamification Session by using the left menu of the Lobby screen:
1: Enter a UNIQUE user name
2: Choose a user type
3: Refresh available rooms
4: Click on the desired room
5: Click on the Connect button

Based on the chosen type of the user the following specific User Interfaces can be showed up:
1: Student User Interface

Local - User Interface

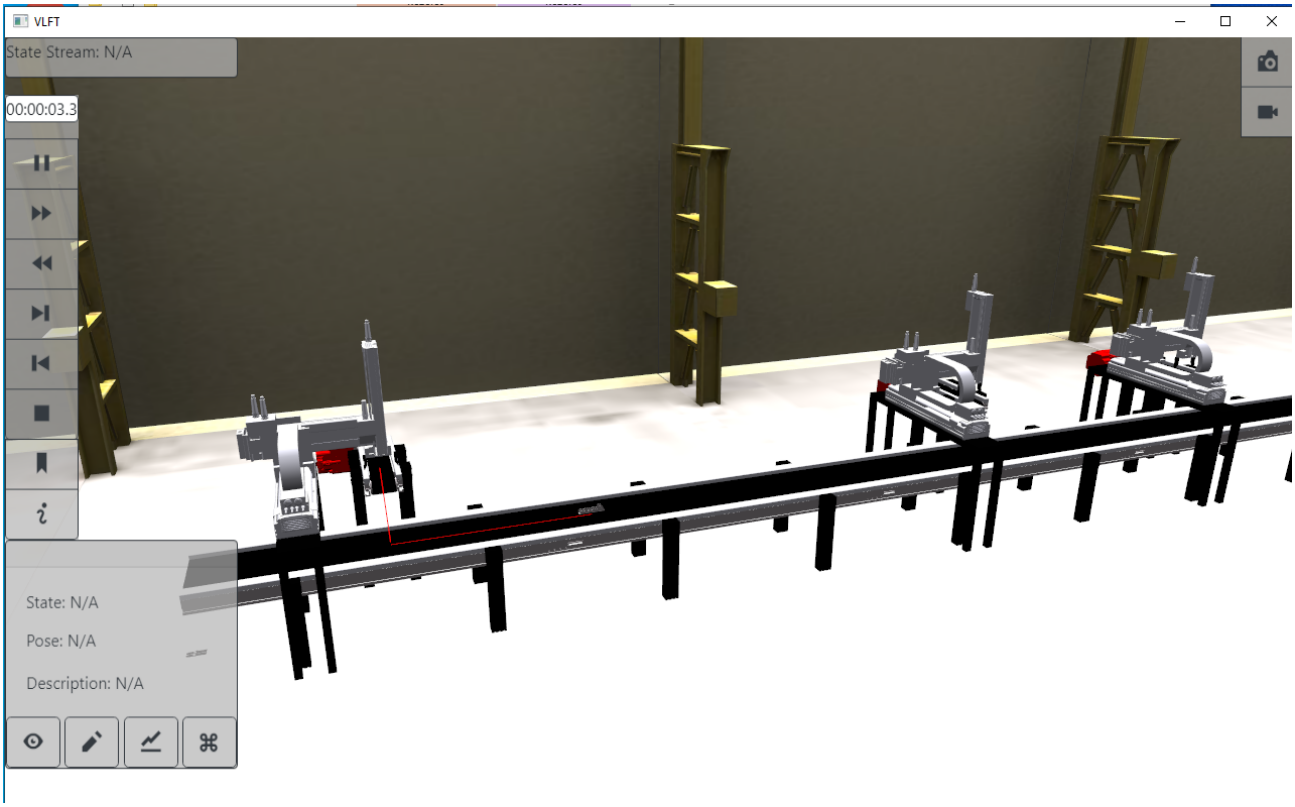
The Local User Interface can be used for testing the resources and the animation on a local machine.

Local User Interface

On the bottom-left side of the menu the Information Box contains the properties of the clicked object.
By the help of the "i" Info Button the Information Box can be toggled (hide/show).

On the left side of the menu the following buttons are available:
1: File Button for downloading all of the states of the manufacturing cell into a log file on the local machine
2: Bookmark Button for starting the animation at a bookmarked time
3: Play Button for starting the animation from the beginning

Right after the Play Button or the Bookmark Button is clicked then the Animation Player Menu will be displayed.



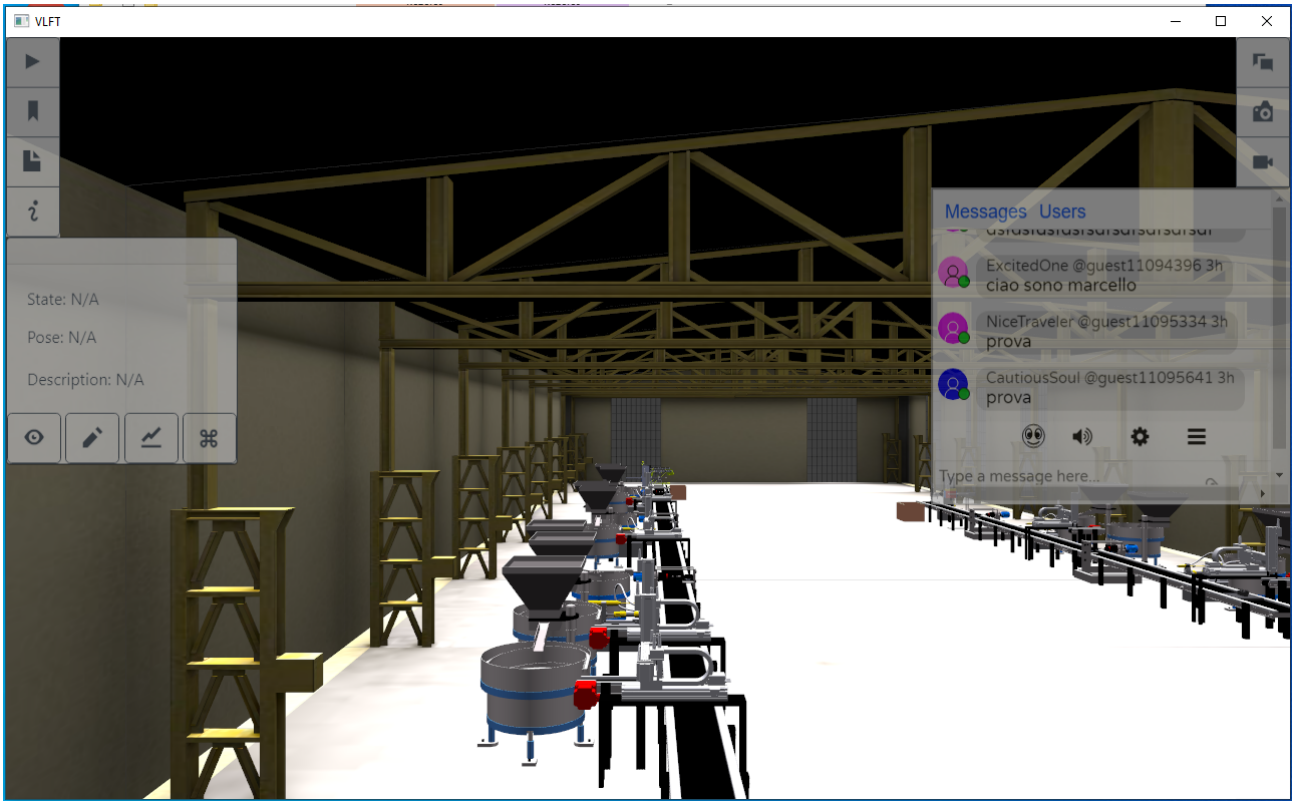
Animation Player Menu

The Animation Player Menu contains the well-known buttons to:
- Pause
- Stop
- Fast Forward
- Fast Bacward
- Jump to the End
- Jump to the Begin

On the right side of the menu the Screen Capture Button and the Scree Cast Video Button is placed. By the help of that buttons a picture or a video can be taken and saved into the installation folder.

Student - User Interface

The Student User Interface is responsible for allowing the desired interactions with the VR space and its elements and also with the other participants in the VLFT Gamification Session.



Student User Interface

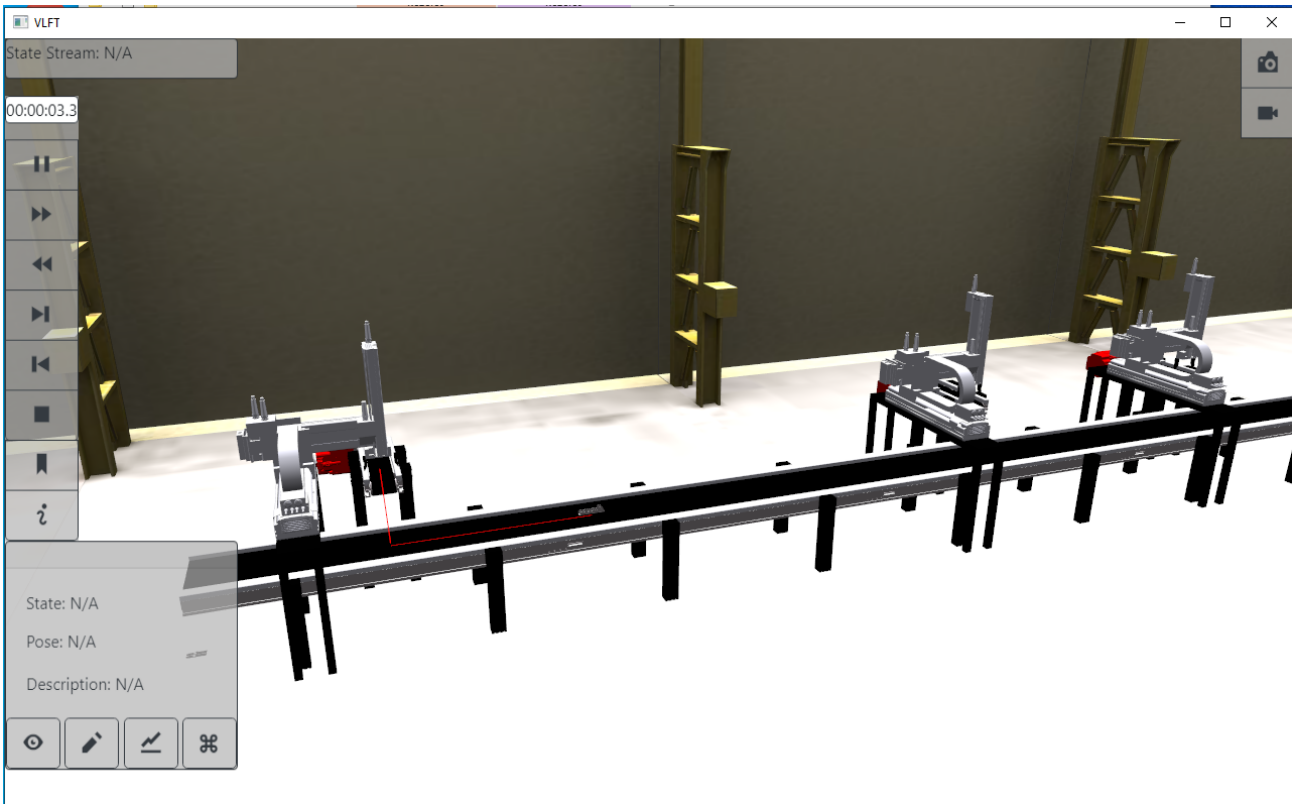
On the bottom-left side of the menu the Information Box contains the properties of the clicked object.
By the help of the "i" Info Button the Information Box can be toggled (hide/show).

On the right side of the menu the Chat Window can be used to send a massage to the other participants in the VLFT Gamification Session.
By the help of the Chat Button the Chat Windows can be toggled (hide/show).

On the left side of the menu the following buttons are available:
1: File Button for downloading all of the states of the manufacturing cell into a log file on the local machine

2: Bookmark Button for starting the animation at a bookmarked time
3: Play Button for starting the animation from the beginning

Right after the Play Button or the Bookmark Button is clicked then the Animation Player Menu will be displayed.



Animation Player Menu

The Animation Player Menu contains the well-known buttons to:
- Pause
- Stop
- Fast Forward
- Fast Bacward
- Jump to the End
- Jump to the Begin

On the right side of the menu the Screen Capture Button and the Scree Cast Video Button is placed. By the help of that buttons a picture or a video can be taken and saved into the installation folder.

Teacher - User Interface

The Teacher User Interface allows the supervisor functionality during a VLFT Gamification Session. Also responsible for allowing the desired interactions with the VR space and its elements. Moreover, the student participants can be guided during the VLFT Gamification Session via the Teacher User Interface.

Teacher User Interface

On the bottom-left side of the menu the Information Box contains the properties of the clicked object.
By the help of the "i" Info Button the Information Box can be toggled (hide/show).

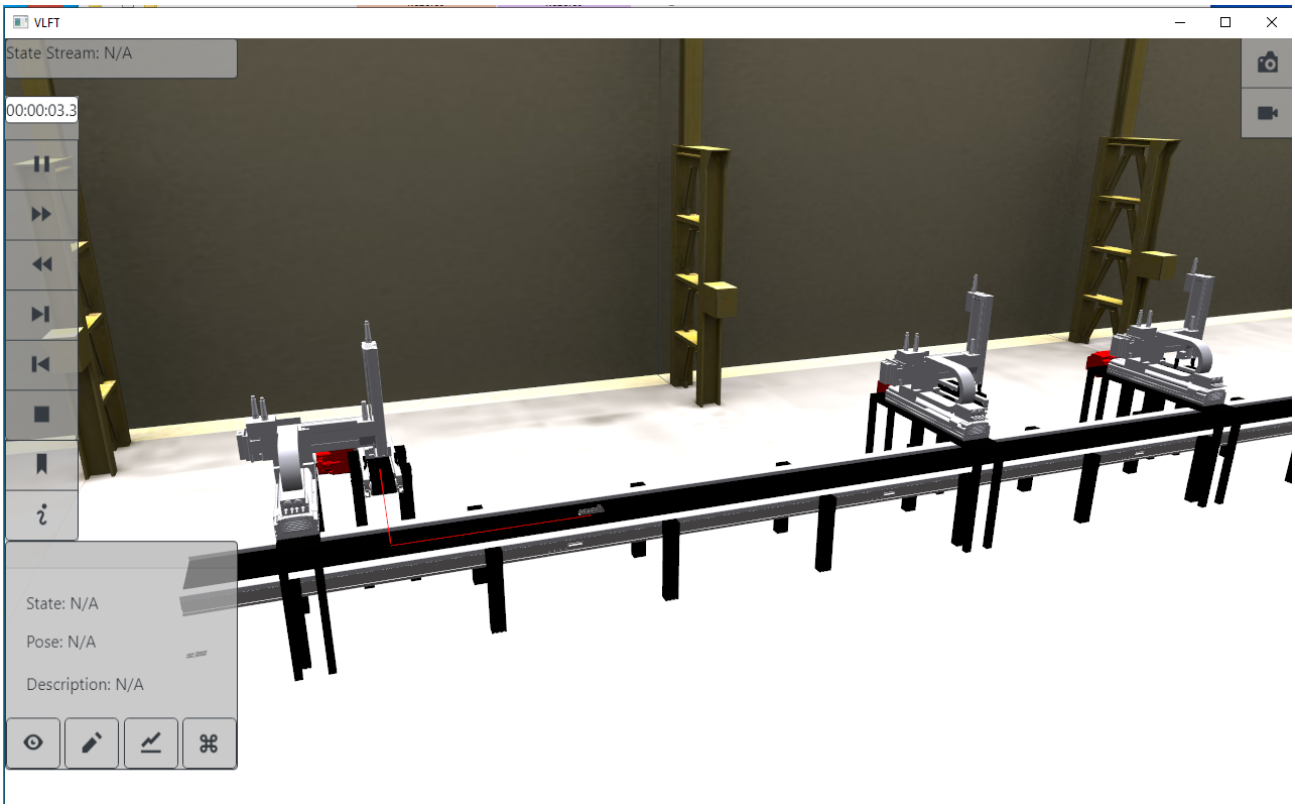
On the right side of the menu the Chat Window can be used to send a massage to the other participants in the VLFT Gamification Session.
By the help of the Chat Button the Chat Windows can be toggled (hide/show).

The supervisor functionalities are available via the following buttons:
1: People Button lists the name of the student
2: File Button logs the movement of the student into the installation folder as a log file

3: Lock Button attaches the students' viewpoint to the teacher's point of view
4: Map Button draws a 2D map and displays the positions of the students relative to the teacher position

On the left side of the menu the following buttons are available: 1: File Button for downloading all of the states of the manufacturing cell into a log file on the local machine 2: Bookmark Button for starting the animation at a bookmarked time 3: Play Button for starting the animation from the beginning

Right after the Play Button or the Bookmark Button is clicked then the Animation Player Menu will be displayed.



Animation Player Menu

The Animation Player Menu contains the well-known buttons to:
- Pause
- Stop
- Fast Forward
- Fast Bacward
- Jump to the End
- Jump to the Begin

On the right side of the menu the Screen Capture Button and the Scree Cast Video Button is placed. By the help of that buttons a picture or a video can be taken and saved into the installation folder.

VLFT Gamification Session

After the Teacher started the desired VLFT Gamification Session on the cloud server then the Students will be able to connect and sharing the same VR visualization of the manufacturing system.

The following two videos (Teacher and Student) demonstrate the functionalities of the Teacher User Interface and the Student User Interface
Start these videos at the same time in parallel to get the proper experience during the playback.

ApertusVR VLFT Gamification Teacher https://youtu.be/ennR9xIppNw?t=33

ApertusVR VLFT Gamification Student https://www.youtube.com/watch?v=HW366go15I8&list=PLMFKeF-URkJqHTYf4b8KM9GQTOuBV5X1K&index=42

VR Mode

The VLFT Gamification Application supports the following two standards for enabling the VR Mode on various VR HMDs:
1: OpenVR for HTC Vive and Oculus Rift
2: OpenXR for Windows Mixed Reality Headsets

ApertusVR VLFT Gamification VR Mode https://www.youtube.com/watch?v=3cd4thZ-Dw0&list=PLMFKeF-URkJqHTYf4b8KM9GQTOuBV5X1K&index=41

The OpenVR Plugin of ApertusVR can be used for visualizing a real manufacturing system during a VLFT Gamification Session.
The following configurations provide sample how to utilize the OpenVR Plugin of ApertusVR during a VLFT Gamification Session.
https://github.com/MTASZTAKI/ApertusVR/blob/bb69eac9cba66c5c1a8076871d6de1bc9a32893f/samples/virtualLearningFactory/studentHtcVive/apeCore.json
https://github.com/MTASZTAKI/ApertusVR/blob/bb69eac9cba66c5c1a8076871d6de1bc9a32893f/samples/virtualLearningFactory/studentHtcVive/apeOgreRenderPlugin.json
Based on the above-mentioned configurations the following two configuration can be modified in the installation folder of VLFT Gamification:
c:\VLFT_Gamification\13\samples\virtualLearningFactory\apeCore.json
c:\VLFT_Gamification\13\samples\virtualLearningFactory\apeOgreRenderPlugin.json

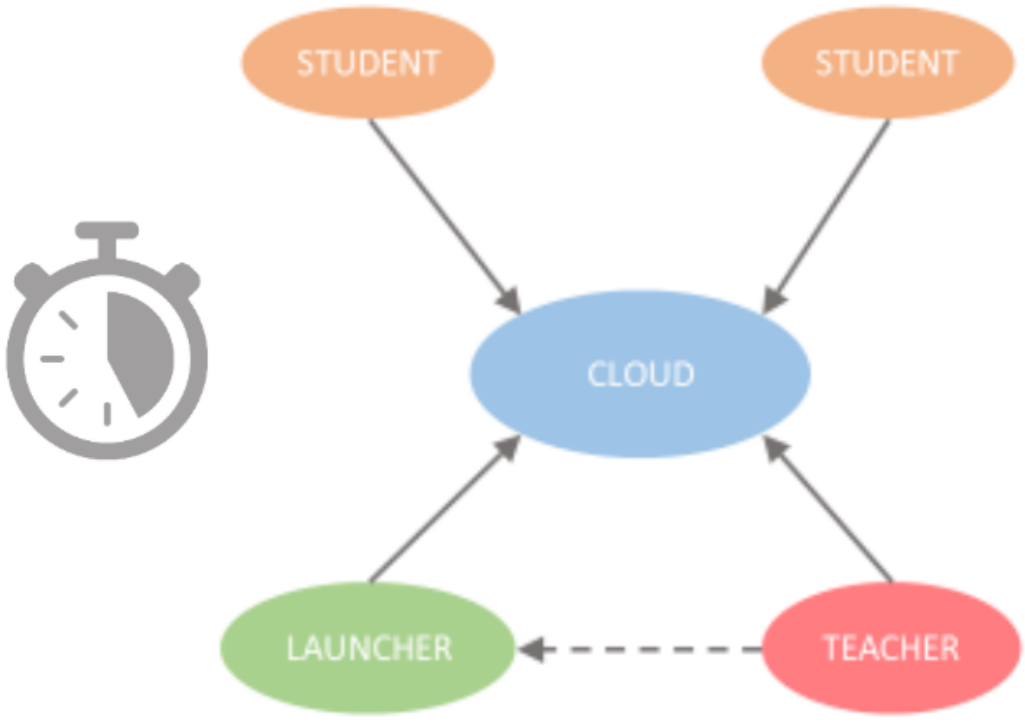
The OpenXR Plugin of ApertusVR can be used for visualizing a real manufacturing system during a VLFT Gamification Session. The following configurations provide sample how to utilize the OpenXR Plugin of ApertusVR during a VLFT Gamification Session.
https://github.com/MTASZTAKI/ApertusVR/blob/0.9.1/samples/hmd/openXR/apeCore.json
https://github.com/MTASZTAKI/ApertusVR/blob/0.9.1/samples/hmd/openXR/apeOgreRenderPlugin.json
Based on the above-mentioned configurations the following two configuration can be modified in the installation folder of VLFT Gamification:

c:\VLFT_Gamification\13\samples\virtualLearningFactory\apeCore.json
c:\VLFT_Gamification\13\samples\virtualLearningFactory\apeOgreRenderPlugin.json

Virtual Learning Factory Toolkit Gamification - Version 2

The aim of VLFT Gamification is exploiting the advances and technologies of modern games to provide students with a realistic representation of a real manufacturing system, making use of VR visualization experiences. Students will learn how to analyse the configuration of a manufacturing systems in terms of: • products manufactured and the associated process;
• machines, associated capability and processing times;
• occurrence of failures and their statistics;
• identify the bottlenecks in the system.

In particular, the students could enter the VR environment of a manufacturing systems where they will find its VR representation, i.e., a production line.

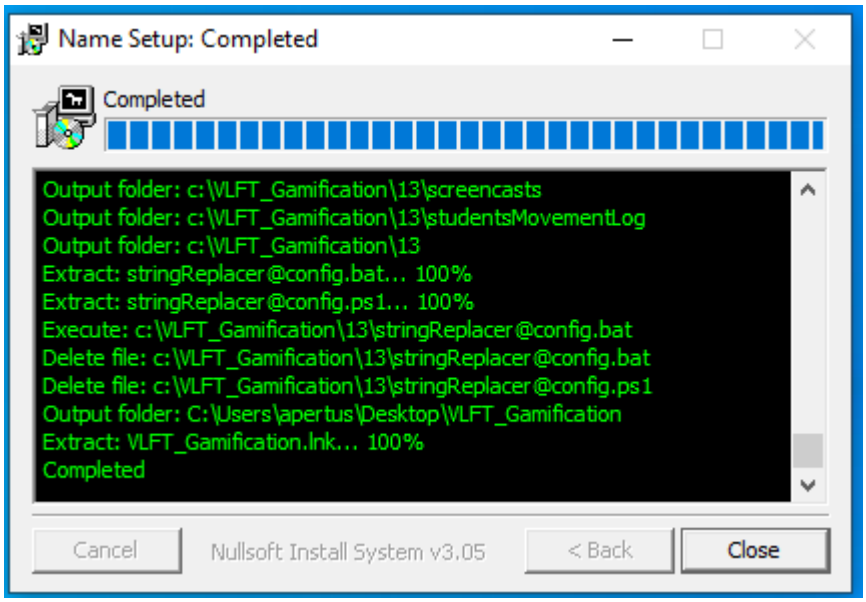


Workflow of a VLFT Gamification Session:
1: The teacher starts a Gamification Session on the cloud server

2: Students can connect to the desired Gamification Session and sharing the same VR visualization of the manufacturing system.
3: Interactions between the participants can be made via the specific Student User Interface and Teacher User Interface

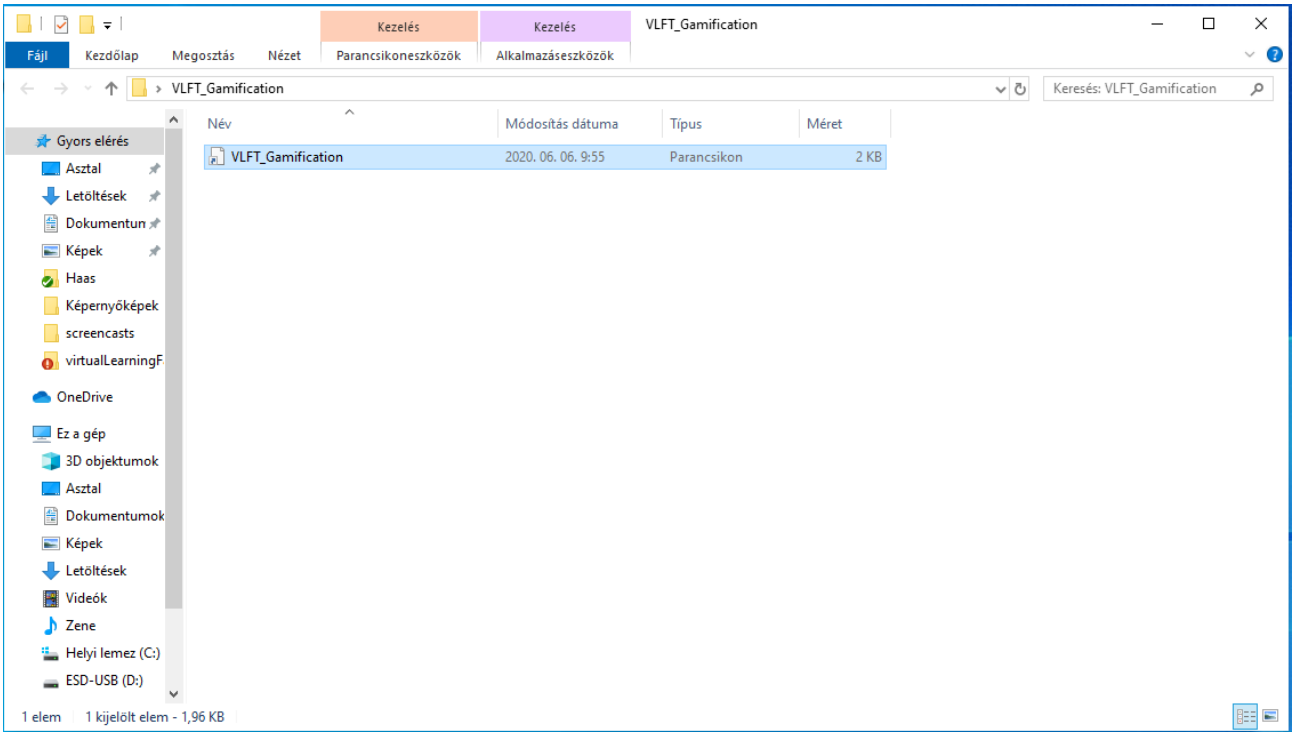
Installation on Windows

The latest version of the VLFT Gamification can be downloaded here



VLFT Gamification Installer

After the installation was completed a folder named VLFT_Gamification can be founded in the desktop.

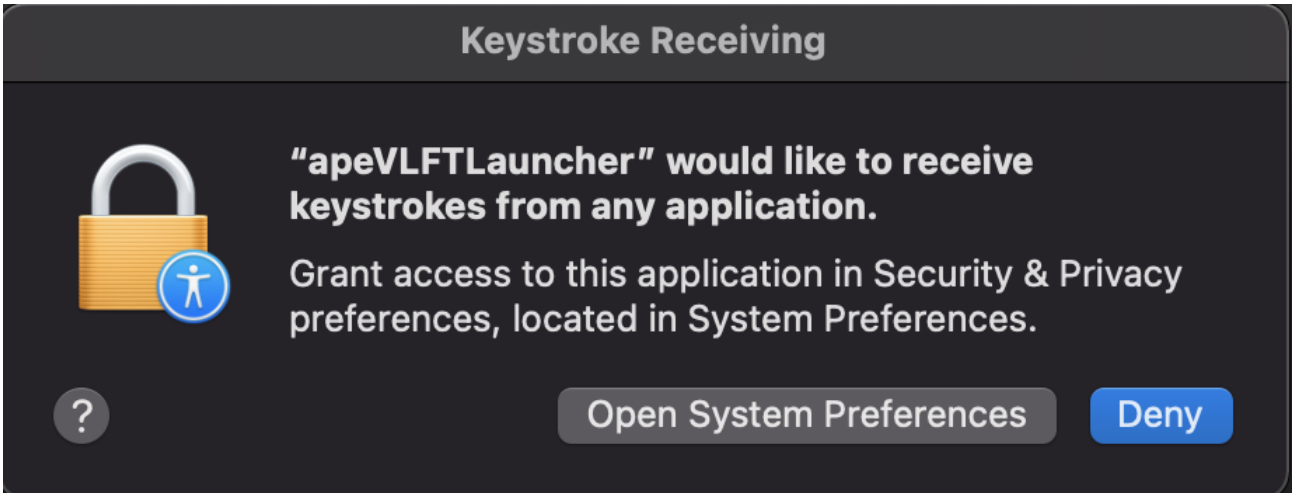


Installation on Apple

The latest version of the VLFT Gamification can be downloaded here

Unzip it and move it into the applications folder, after that right click->open. If a window pops up with a text that you can not open it because the developer cannot be verified, then click close, and right click again->open.

After that it should start and a window should pop up ascing for permission. Click Open System Preferences and with the + icon grant it permission for keystrokes and grant it full disk access.



After this, you should be able to use the application.

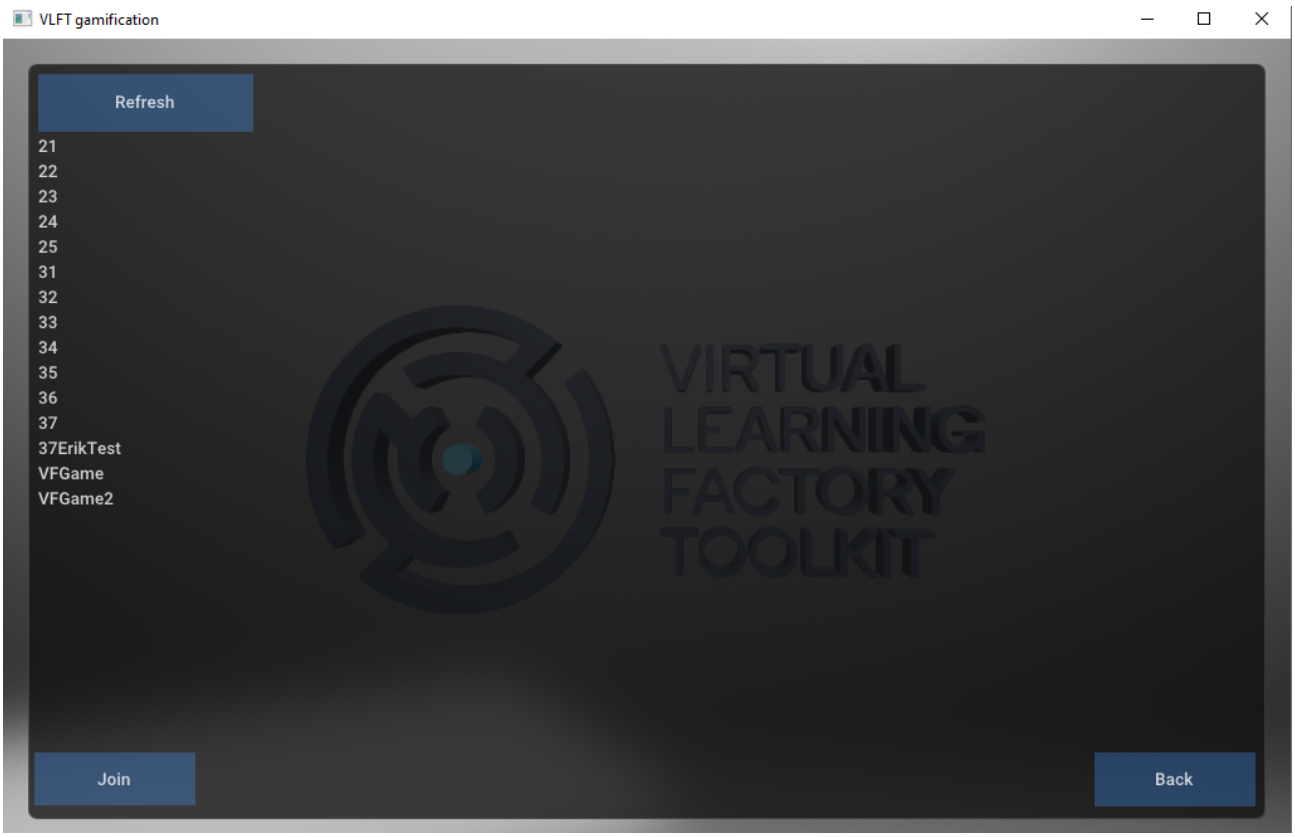
Lobby

After the VLFT Gamification Application started the Lobby GUI will appear.

Two options are available:

1: Single-player mode
2: Multi-player mode
   2.1: Student mode
   2.2: Teacher mode

The list of uploaded rooms will appear in the case of Single-player mode. After the desired room was selected and the "Start" button was clicked then the resources will be updated on-demand and the configuration files of the selected room will be downloaded. The room is only available locally for one user no other participant can be involved.



The list of online rooms will appear in the case of Multi-player mode. After the desired room was selected and the "Join" button was clicked then the resources will be updated on-demand and the user will be involved in a shared room with other participants.

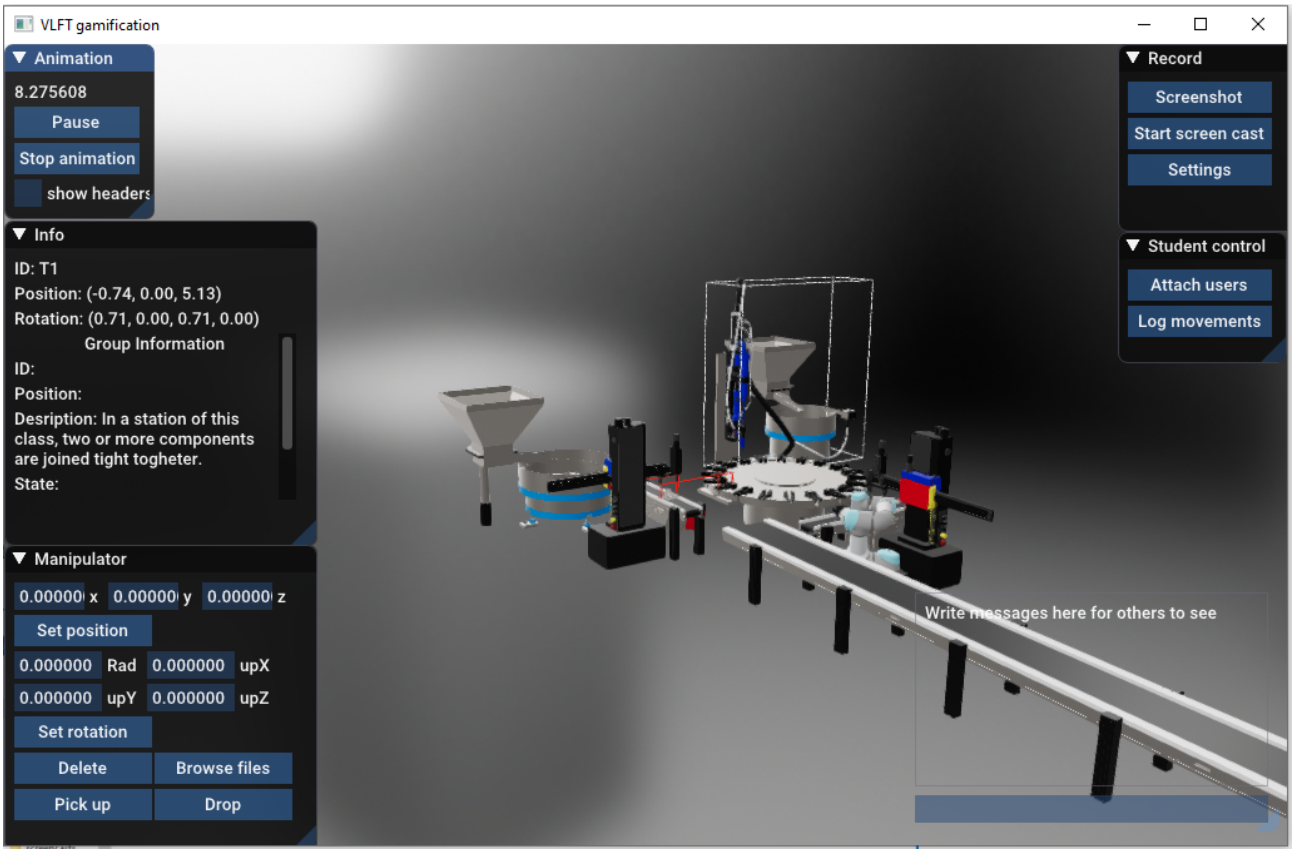
Single Player

The Single Player Interface can be used for testing the resources and the animation on a local machine.

The "Info" contains the properties of the clicked object. This menu element can be toggled to hide/show by the ▽

By the help of the "Manipulator", the selected/clicked object can be moved to the desired place. This menu element can be toggled to hide/show by the ▽

The "Animation" menu allows controlling of the play of the pre-recorded motions of the objects. This menu element can be toggled to hide/show by the ▽

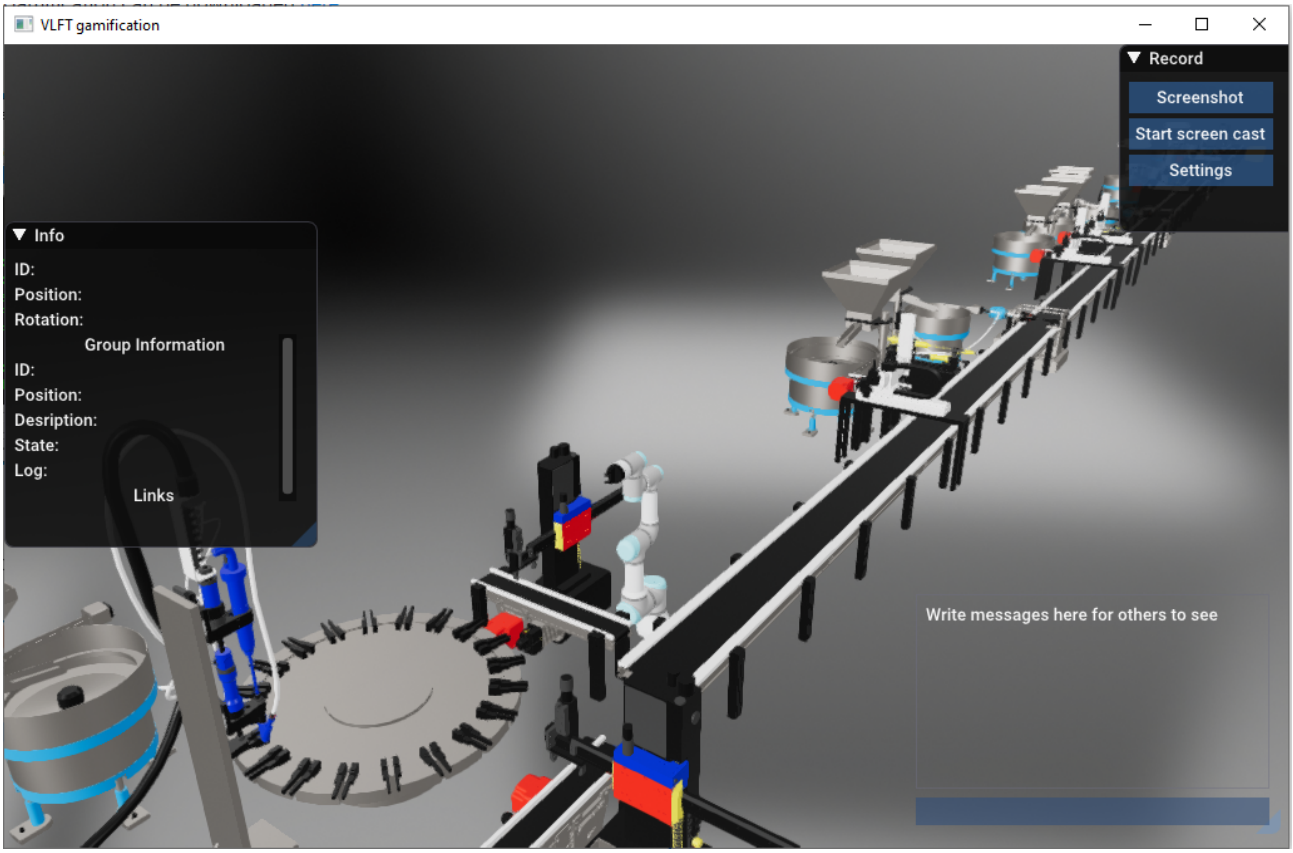


The animation can be controlled by the pause and stop buttons. The current state of the objects can be toggled to hide/show by the "show headers" checkbox.

The "Screenshot" Button and the "Screencast" Button allow the possiblity to make a picture or a video which will be saved into the installation folder.

Multi Player - Student

The Student User Interface is responsible for allowing the desired interactions with the VR space and its elements and also with the other participants in the VLFT Gamification Session.



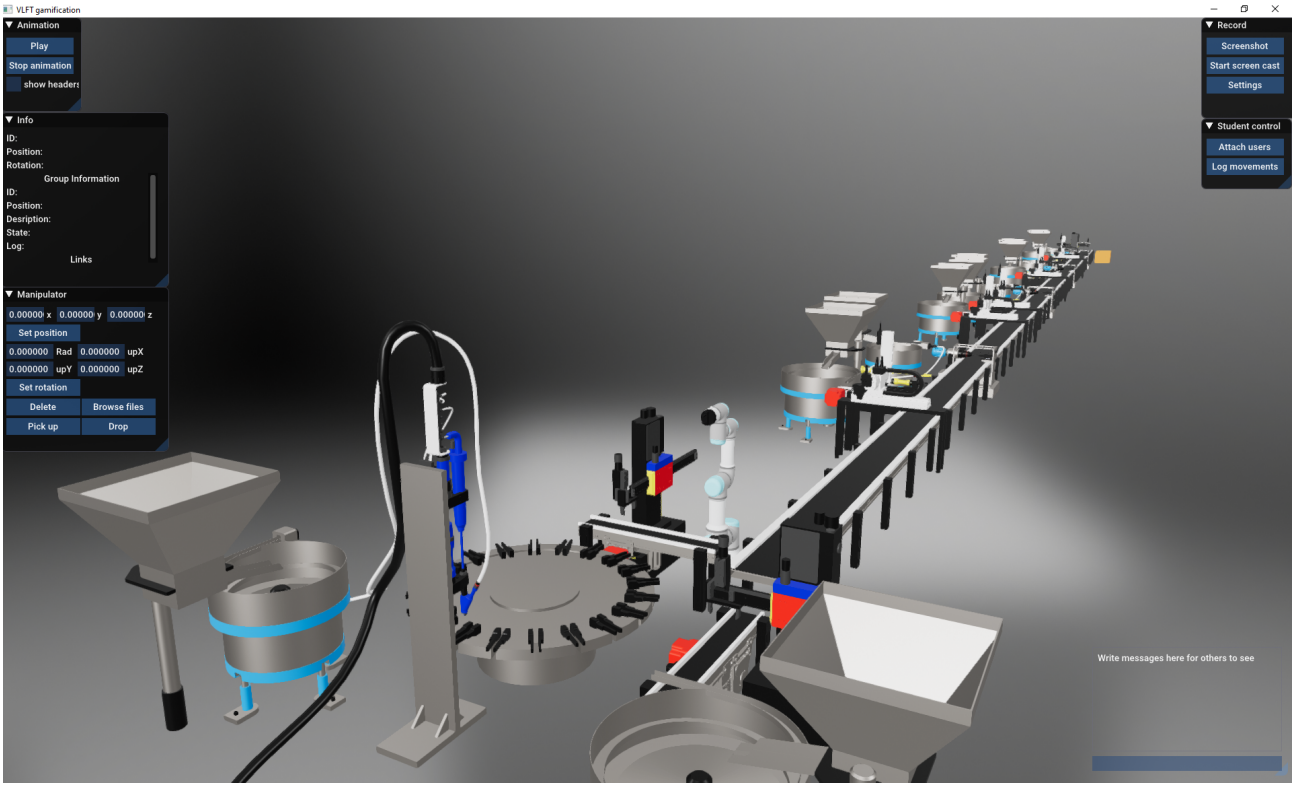
The "Info" contains the properties of the clicked object. This menu element can be toggled to hide/show by the ▽

The "Screenshot" Button and the "Screencast" Button allow the possiblity to make a picture or a video which will be saved into the installation folder.

The Chat Window can be used to send a massage to the other participants in the VLFT Gamification Session.

Multi Player - Teacher

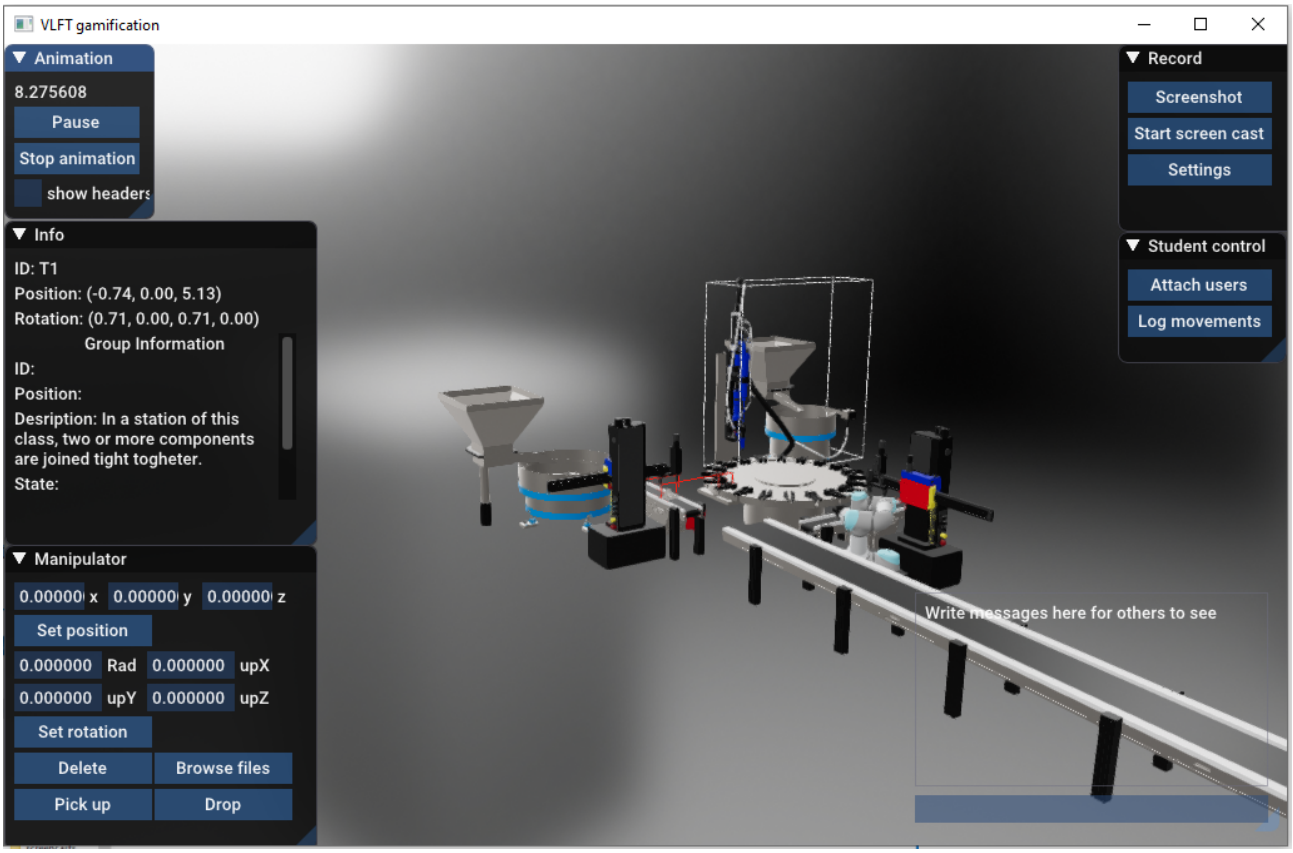
The Teacher User Interface allows the supervisor functionality during a VLFT Gamification Session. Also responsible for allowing the desired interactions with the VR space and its elements. Moreover, the student participants can be guided during the VLFT Gamification Session via the Teacher User Interface.



The "Info" contains the properties of the clicked object. This menu element can be toggled to hide/show by the ▽

By the help of the "Manipulator", the selected/clicked object can be moved to the desired place. This menu element can be toggled to hide/show by the ▽

The "Animation" menu allows controlling of the play of the pre-recorded motions of the objects. This menu element can be toggled to hide/show by the ▽



The animation can be controlled by the pause and stop buttons. The current state of the objects can be toggled to hide/show by the "show headers" checkbox.

The "Screenshot" Button and the "Screencast" Button allow the possiblity to make a picture or a video which will be saved into the installation folder.

The control of the students' can be done by the following buttons:

1: "Attach users" button attaches the students' viewpoint to the teacher's point of view

2: "Log movements" button logs the movement of the student into the installation folder as a log file

3.5 MTM

Introduction

The methods time measurement - universal analyzing system (MTM-UAS) is a methodology to clearly measure the processing times of some determined processes made by the operators, both manually and with the aid of some tools. The MTM methodology has been introduced in the '90s, and the last version is one of the most flexible and fast-to apply techniques, suitable in cases where relatively simple and short actions of the operator are involved. It is based on the classification of the possible operations done by the operator in 7 sections, that are translated in tables with different attributes from each other that returns values of processing TMUs depending on the situation.

MTM-UAS is a widely used method in the industry, especially when speaking about lot production. It is fast to use and quite flexible at the same time but it can have some limitations:

- the focus is on basic operations and not on basic motions like some other versions of the MTM: this lead to a **less accuracy** in the evaluation, an important issue in situations where complex and high-effort actions are required;
- **subactivities are not taken into account**, since the single activity has to be univocally assaigned to a specific section, and the time value is founded basing on the weight of the part handled, the accuracy needed in the movement and its difficulty;
- due to this, it could be necessary to put attenction in the definition of the activities, to avoid the underestimation of the time due to a choice of a very large aggregate activities not providing the due details for the analysis.

How to start

The tool implementing the MTM-UAS methodology consists of three components:

- an **excel input file**, to be filled with the input data according to the specific instructions provided;
- a **python script** processing the inputs and classifying the activities according to the the methodology;
- a **python script** processing input data to estimate the processing times according to the MTM-UAS methodology;

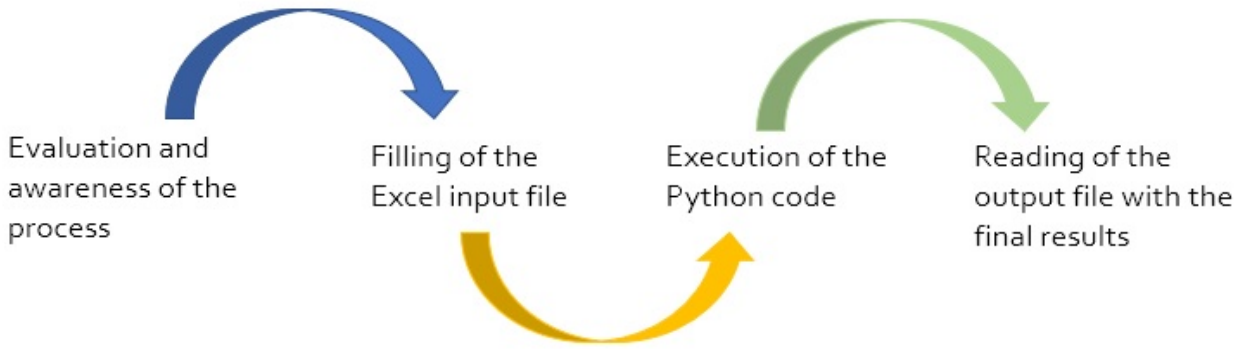


Figure 1: General architecture of the MTM-UAS tool

Formalise the process

In order to prepare the input data, the process to be descrived and assessed, also interviewing the opertator in charge of it. The main information to be determined and defined are:

1. the main basic operations, the main steps of the work at issue;
2. the typology of each operation identified;
3. the information required by each of the operations in order to get the right time value, in terms of accuracy, difficulty and distance covered.

The MTM-UAS methodology considers different classes of activities reported in the table below, together with a description:

Type	Description
Get and Place	grasp and put an object, taking into account its distance and shape
Place	put an object in a more or less precise way
Handle Aid	tools handling using for example pliers or screwdrivers
Operate	the actuation or press of a botton or a lever
Motion Cycles	tasks with movements carried out continuously and repeatedly
Body Motions	the movements are done to move
Visual Inspection	action of the head and eyes to control an element

These clsses of activities must be further detailed according to the 7 tables shown in the figure below.

motion length in cm	≤ 20	> 20 to ≤ 50	> 50 to ≤ 80
distance range	1	2	3

Get and Place			Code	1	2	3
				TMU		
≤1 daN	easy	approx.	AA	20	35	50
		loose	AB	30	45	60
		tight	AC	40	55	70
	difficult	approx.	AD	20	45	60
		loose	AE	30	55	70
		tight	AF	40	65	80
	handful	approx.	AG	40	65	80
> 1 daN to ≤ 8 daN		approx.	AH	25	45	55
		loose	AJ	40	65	75
		tight	AK	50	75	85
> 8 daN to ≤ 22 daN		approx.	AL	80	105	115
		loose	AM	95	120	130
		tight	AN	120	145	160

Place		Code	1	2	3
			TMU		
	approx.	PA	10	20	25
	loose	PB	20	30	35
	tight	PC	30	40	45

Handle Aid	Code	1	2	3
		TMU		
approximate	HA	25	45	65
loose	HB	40	60	75
tight	HC	50	70	85

Operate	Code	1	2	3
simple	BA	10	25	40
compound	BB	30	45	60

Motion Cycles	Code	1	2	3
one motion	ZA	5	15	20
motion sequence	ZB	10	30	40
shift and one motion	ZC	30	45	55
tighten or loosen	ZD	20		

Body Motions	Code	TMU
walk / m	KA	25
bend, stoop, kneel (incl. arise)	KB	60
sit and stand	KC	110

Visual Inspection	VA	15

According to the different classes of activities, further specifications are needed.

With respect to 'Get and place', the first column refers to the WEIGHT, expressed in daN. Then the DIFFICULTY is considered only for the low weight case. For the 'Get and place', 'Place' and 'Handle aid' classes also the ACCURACY dimension must be declared.
or the 'Operate', 'Motion cycles' and 'Body motions' classes the specific TYPOLOGY of activity executed must be specified.
For the classes of activities involving walking, 3 different ranges of distance can be selected.

Grounding on these specifications, a value for the Time Measurement Unit (**1 TMU = 36 milliseconds**) is provided,

After this initial assessment it is possible to move to the preparation of the input data.

Prepare the input data

The descrition of the process to be analysed must be provided through an Excel file whose template can be dowloaded here:

MTMgen.xlsx MTMgen.xlsx - 70KB

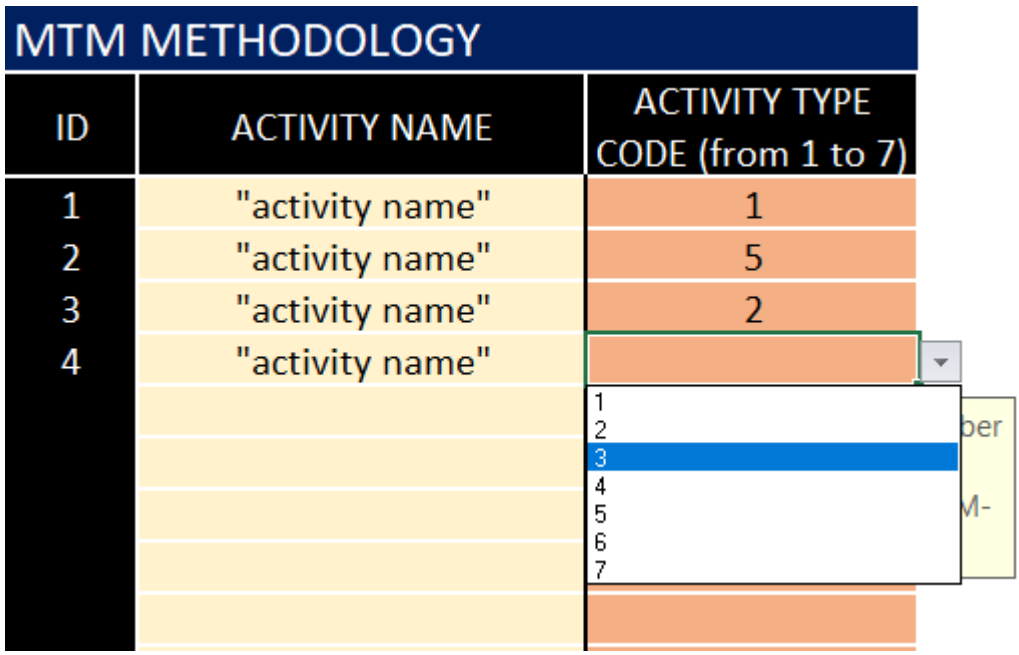
It is advisable to change the name of the file according to the process to be analysed, to have a clear, univocal name without spaces.

The Excel file contains 4 sheets to be filled in:

- **Activities**

This sheet is used to declare the activities to be executed by the human operator (see figure below):

1. the first column requires the **ID** of the activities, which have to be put in the order of execution, starting from 1 until a maximum of 100, that is the maximum number of activities that this toolkit can analyse in a single process;
2. The second column is for **Activity Name**, that requires a complete, well identifying and univocal name of the activity considered (if one name is repeated twice, this means that the activities are exactly the same);



1. The third column is the most important one, since it requires the **Activity Type Code**, that is a number refered to the section/table which the activity has been assigned to: the right code can be identified by means of the Table 2, and it has to be selected by the evaluator by chosing the most suitable option between the ones described in the drop down menu :

Table	Code
'Get and Place'	1
'Place'	2
'Handle Aid'	3
'Operate'	4
'Motion Cycles'	5
'Body Motions'	6
'Visual Inspection'	7

Table 2

- **Paths** In the second sheet the assessor will see two tables in the interface: in the first one, named *Locations* (Figure 5), he has to insert the meaningful locations of the work, that are all the starting and the ending points of all the parts and tools at issue, in addiction to the base working station. Each of them have to be identified by:
- a name in the first column
- an acronym in the second column, composed by one capital letter and one number
- the coordinates in centimeters in the third and the fourth columns, for the X and the Y respectively. These are the coordinates of the centers of gravity previously identified, in the step of the analysis of the situation.

LOCATIONS		COORDINATES	
NAME	ACRONYM	Xcog	Ycog
"location name"	"acronym"	...	...
"location name"	"acronym"	...	...
"location name"	"acronym"	...	...

Figure 5

Once filled in, this table will be a reference for the definition of the second one, named *Paths*, shown in Figure 6.

PATHS	COORDINATES					
ID	START	END	Xs	Ys	Xe	Ye
"path id"	"location name"	"location name"	...	...	...	...
"path id"	"location name"	"location name"	...	...	...	...
"path id"	"location name"	"location name"	...	...	...	...

Figure 6

Here the evaluator needs to compile the columns in the following way:

1. first the **ID** of the path, with the capital letter "P" followed by numbers from 1 to a maximum of 100, that is the maximum number of paths that this code can analyse in a single process;
2. then the second and the third column (**START** and **END**) require the acronyms of the starting and the ending location of the path, respectively;
3. finally in the columns '**Xs**' and '**Ys**' the coordinates of the starting locations have to be inserted from the previous table, while in '**Xe**' and '**Ye**' the ones of the ending locations.
4. **Movements** This sheet is made up by a main table, named '*Movements*' (Figure 7), whose aim is that of showing the whole movement and so the whole distance covered by each activity, expressed by the sum of all the paths done during it.

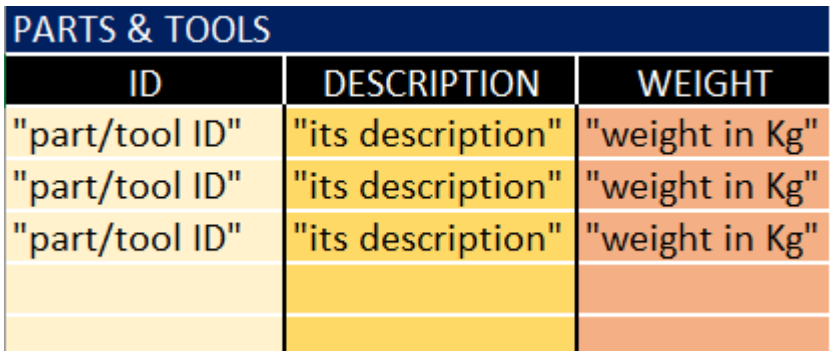
MOVEMENTS		PATHS SEQUENCE									
ID	ACTIVITY NAME	1	2	3	4	5	6	7	8	9	10
1	"activity name"	Px									
2	"activity name"	Px	Py								
3	"activity name"	Pz									
4	"activity name"	Py									
0	0										
0	0										
0	0										

Figure 7

The initial layout seen by the assessor has the first and the second columns, the '**ID**' and the '**Activity Name**' respectively, already filled with the activities inserted in the first sheet 'Activities', thanks to a direct link between the two sheets. So the input required by the successive columns, '**Paths Sequence 1, 2, 3,...**', is the list of paths done in each activity, indicated with the path ID defined in the previous sheet, one for each column until a maximum of 20, that is the maximum number of paths that can be composed to form the movement of a single activity in this code. To ease this operation, in the same sheet on the right is reproduced the dual table about "*Paths*" just seen above, so that the assessor doesn't need to change the sheet many times to complete his evaluation.

- **PartsTools**

The filling of this last sheet, dedicated to the Parts and Tools used during the working process, is quite simple (Figure 8):



PARTS & TOOLS		
ID	DESCRIPTION	WEIGHT
"part/tool ID"	"its description"	"weight in Kg"
"part/tool ID"	"its description"	"weight in Kg"
"part/tool ID"	"its description"	"weight in Kg"

Figure 8

1. The first column '**ID**' requires the ID of the part or tool: for the sake of clarity, the evaluator should first list all the Parts involved in the work, whose ID has to be composed by the letter X followed by a univocally identifying number; then, all the Tools used to accomplish the process have to be listed, with an ID made up with the letter T followed by a univocally identifying number.

The maximum total number of parts and tools supported by this toolkit is 100.

1. This second column is related to a general '**Description**' of the part or tool;
2. The last column, named '**WEIGHT**', requires a number indicating the weight of the part or tool in Kg, previously measured with the aid of the right instruments or by looking at the technical product sheet.

As a general reminder, the assessor needs to literally follow all the indications given above, so that the code and consequently the methodology can work properly giving back the right results. Particularly, a great attention is recommended in being complaint with the names and acronyms choosen for a given element: they have to be univocal for the same elements along all the tables. Before passing to the analysis of the Python code, an example to clarify the approach is showed.

2.1.1 Example (Catenaccio case - Layout 1)

The example provided from the layout 1 of the Catenaccio case analysed in the VLFT project is very important to understand how to practically apply this part of the toolkit, and to clearly get the flow of reasoning required by it. In the Figures 9, 10, 11 and 12 are shown the the just described tables of the Excel input file, properly filled by the assessor with all the data needed; respectively the "Activities", "Paths", "Movements" and "PartsTools" sheets are shown.

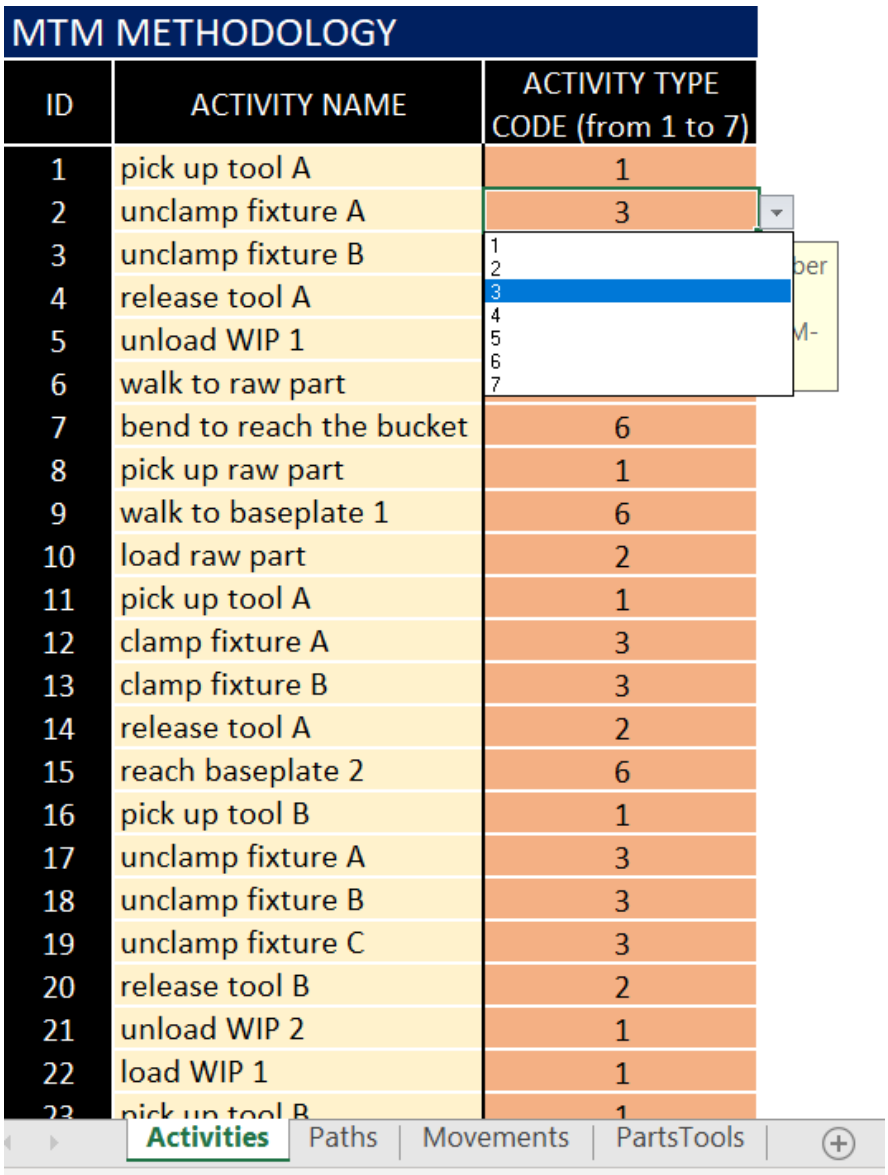


Figure 9 : the Activities sheet.

LOCATIONS		COORDINATES	
NAME	ACRONYM	Xcog	Ycog
Raw	RAW	118,8	103,3
Workpiece 1	WP1	0	-15
Work in progress 1	WIP1	-25	-15
Workpiece 2	WP2	-50	-15
Work in progress 2	WIP2	-75	-15
Workpiece 3	WP3	-100	-15
Finished	F	-268,8	103,3
Tool place 1	T1	0	-50
Tool place 2	T2	-50	-50
Tool place 3	T3	-100	-50
Base 1	B1	0	0
Base 2	B2	-50	0
Base 3	B3	-100	0

PATHS	COORDINATES					
ID	START	END	Xs	Ys	Xe	Ye
P1	B1	T1	0	0	0	-50
P2	B1	WP1	0	0	0	-15
P3	B1	RAW	0	0	118,8	103,3
P4	B1	WIP1	0	0	-25	-15
P5	B1	B2	0	0	-50	0
P6	B2	T2	-50	0	-50	-50
P7	B2	WP2	-50	0	-50	-15
P8	B2	WIP2	-50	0	-75	-15
P9	B2	WIP1	-50	0	-25	-15
P10	B2	B3	-50	0	-100	0
P11	B3	WIP2	-100	0	-75	-15
P12	B3	T3	-100	0	-100	-50
P13	B3	WP3	-100	0	-100	-15
P14	B3	F	-100	0	-268,8	103,3
P15	B3	B1	-100	0	0	0

Activities | **Paths** | Movements | PartsTools

Figure 10 : the Paths sheet.

MOVEMENTS		PATHS SEQUENCE																			
ID	ACTIVITY NAME	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	pick up tool A	P1	P1																		
2	unclamp fixture A																				
3	unclamp fixture B																				
4	release tool A	P1	P1																		
5	unload WIP 1	P4	P4																		
6	walk to raw part	P3																			
7	bend to reach the bucket																				
8	pick up raw part																				
9	walk to baseplate 1	P3																			
10	load raw part	P2	P2																		
11	pick up tool A	P1	P1																		
12	clamp fixture A																				
13	clamp fixture B																				
14	release tool A	P1	P1																		
15	reach baseplate 2	P5	P5																		
16	pick up tool B	P6	P6																		
17	unclamp fixture A																				
18	unclamp fixture B																				
19	unclamp fixture C																				
20	release tool B	P6	P6																		
21	unload WIP 2	P8	P8																		
22	load WIP 1	P9	P9																		
23	pick up tool B	P6	P6																		
24	clamp fixture A																				

Activities | Paths | **Movements** | PartsTools

Figure 11 : the Movements sheet.

PARTS & TOOLS		
ID	DESCRIPTION	WEIGHT
X1	raw part	1,5
X2	wprkpiece 1	1
X3	workpiece 2	0,7
X4	workpiece 3	0,7
T1	tool 1	0,7
T2	tool 2	0,6

Figure 12 : the PartsTools sheet.

Execution and results

The final step of the model developed for the MTM-UAS methodology is the downloading of the Excel file modified by the Python code with the desired outputs: they will be two new different sheets, one called "Acitivity codes", where basically the times for each single activity are reported, and the other one called "Total times", where the total processing time of the working cycle is finally shown. As already mentioned before, also in this case the attached images are refered to the Excel file of the layout 1 of the Catenaccio case, since these are output of the Python code so a general model is not available.

x	ID	ACTIVITY	ACTIVITY CODE	ACTIVITY TIME
0	1	pick up tool A	AA	50
1	5	unload WIP 1	AE	70
2	8	pick up raw part	AH	25
3	11	pick up tool A	AA	50
4	16	pick up tool B	AA	50
5	21	unload WIP 2	AE	70
6	22	load WIP 1	AE	70
7	23	pick up tool B	AA	50
8	29	pick up tool A	AA	50
9	33	unload finished part	AE	55
10	37	load WIP 2	AE	70
11	4	release tool A	PA	25
12	10	load raw part	PB	30
13	14	release tool A	PA	25
14	20	release tool B	PA	25
15	27	release tool B	PA	25
16	32	release tool A	PA	25
17	35	release finished part	PA	10
18	41	release tool A	PA	25
19	2	unclamp fixture A	HB	40
20	3	unclamp fixture B	HB	40
21	12	clamp fixture A	HB	40
22	13	clamp fixture B	HB	40
23	17	unclamp fixture A	HB	40
24	18	unclamp fixture B	HB	40
25	19	unclamp fixture C	HB	40

In the above figure, that is the "Activity codes" sheet, four columns are displayed:

1. in the first one, called "**ID**", the ID of the activity considered is shown: the order is not the one of execution, instead it is considered the typology of activity according to the MTM-UAS tables, from 'Get and place' to 'Visual inspection';

2. the second one is named "**ACTIVITY**" and it is for the name of the activities correspondent to the ID;
3. the third column, called "**ACTIVITY CODE**", shows a code related to the specific section to which each activity has been allocated by the code, so that the assessor is able able to verify both the functioning of the toolkit and make some considerations about his assumptions and data inserted;
4. the last column is named "**ACTIVITY TIME**" and contains the time values in TMUs related to each single activity.

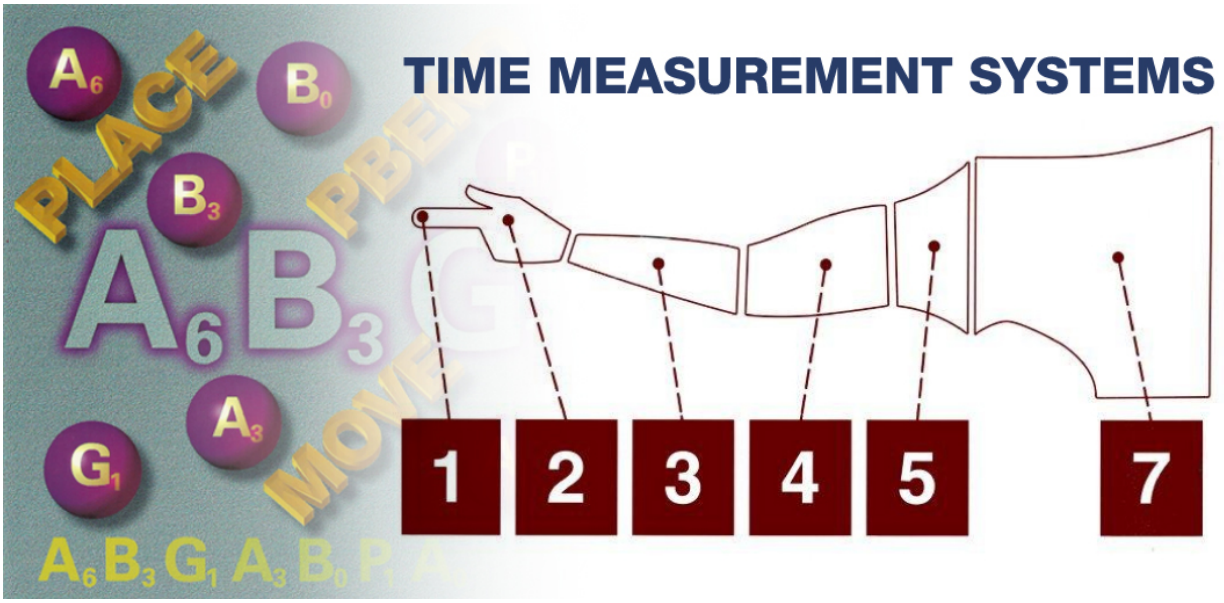
Finally the second output sheet is the one showed in the figures below that basically reproduce the content of one Excel sheet:

x	TOTt Get and place	TOTt Place	TOTt Handle aid	TOTt Operate	TOTt Motion cycles	TOTt Body motions	TOTt Visual inspection
TIME	610	190	560	0	0	300	0

TOTAL TIME TMU	TOTAL TIME SECONDS	TOTAL TIME MINUTES
1660	59,76	0,996

1. In the first seven columns the values of the processing times for each single section are reported in TMUs;
2. in the last three columns is instead possible to see the final total value of the processing time for the working cycle considered, in terms of TMUs, seconds and minutes, that is the final and most important outcome of this methodology.

3.6 MOST

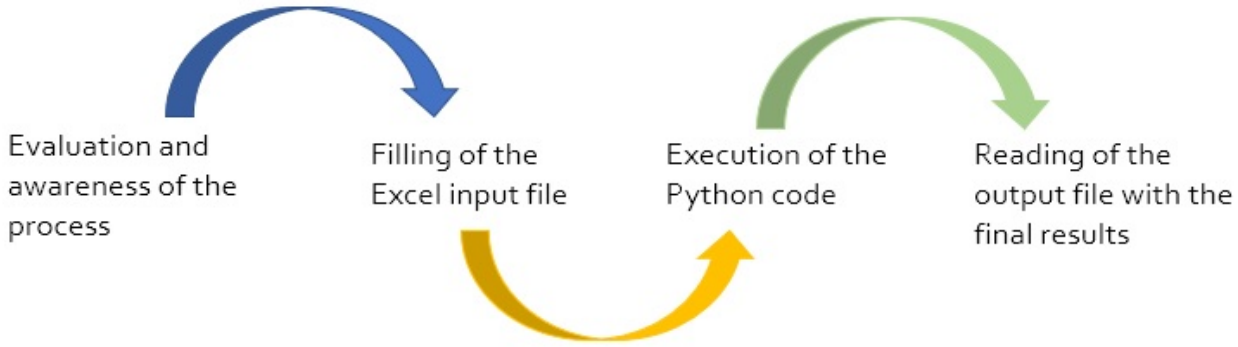


Introduction

The Mynard Operation Sequence Tecnique is a well known methodology in the field of processing times estimation, based on the object to be moved rather than on the movement itself, since each motion and action considered is always directly linked to the object at issue. It is very well structured, since the tables are defined in a univocal way with well defined sections, so that many aspects of the movements can be taken into account. On the other hand it is also very fast in the application (faster than MTM-UAS) and flexible, since it is easy to modify when a change in the working cycle or in an activity occurs. The Basic MOST is based to a division in three main sections, corresponding to three tables, named **General Move** sequence model, for the activities done moving freely in the space, **Tool Use** sequence model, for those actions accomplished with the aid of a tool, and **Controlled Move** sequence model, for the activities done with a contact with other surfaces. Each of them are characterized by a predefined sequence of letters, each of which requires an index to be assaigned by looking at the tables. This brings to the main advantage that this methodology has, compared to the MTM-UAS: here subactivities are taken into account, since for each activity inserted, depending on the section, many letters are available to specify which sub-parts of the macro action contributes to the total time required. The final sum of all the indexes in the tables will give an indication of the

processessing time of the working cycle at issue. Following the main objective of this project, that is to give back a toolkit of digital instruments to be used in order to ease the implementation of some methodologies for the measurement of the work, in this specific package for the Basic MOST the main elements are:

- a **general excel input file**, to be filled in by the operator by following some specific instructions;
- **a python code**, that takes the table in the Excel input file and uses its values to compute the required processing time;
- **one example** related to one of the layouts considered for the manual operations in the Catenaccio case, on which the project is based, so to give the possibility to the user to have a practical interface for a complete understanding of the methodology.



Limitations

As already said, the MOST is a very widely used methodology in the industrial environment, very fast and flexible in the application, but it presents also some limitations:

- despite the presence of many subactivities to specify the details of the lines inserted by the assessor, this can also represent a disadvantage, in fact in this way it is very easy to fall in **redundancy**, by overestimating the time values for example by inserting indexes related to the same situation in more than one cell, also related to different macro-activities. This will be further exaplained in the next chapter;
- the tables, as will be seen later, are not so rigorous in the way of assaigning a value instead of another, in fact the descriptions leave space to a certain **arbitrariness** in the

judgement;

- the choosen version, the Basic MOST, is the most general one and the most widely used since it includes the majority of the industrial situations to deal with; anyway in cases with very low distances and cycle times or on the contrary very high distances with very big loads to handle, this methodology is not so adapted, and the **Mini MOST** and **Maxi MOST** respectively would be a better choice for a good analysis.

How to start

Prepare the assessment

In this first phase the assessor needs to well analyse and understand the process to be assessed, first by his own and then by interviewing the worker, so to have a clear and complete view of the situation. The main things to be determined and defined are:

- the main operations of the work at issue;
- the typology of each operation identified;
- the single steps in which each operation considered is diveded.

The fundamental base in order to accomplish these three points is represented by the following tables, related to three main sections of this methodology:

1. in Figure 2 it is possible to see the '***General move sequence model*** table:
   Figure 2 - Images-MOST/ugim3.PNG
   "General Move deals with the spatial displacement of one or more objects. Under manual control, the object follows an unrestricted path through the air. If the object is in contact with, restricted by or attached to another object during the move, the General Move Sequence Model is not applicable". (Kjell B. Zandin - MOST Work Measurement Systems)
   Here it can be seen that the general sequence of letters describing the considered subactivities is made up by: **A B G A B P A**, where:
   - A = *action distance*: "this parameter is used to analyze all spatial movements or actions of the fingers, hands and/or feet, either loaded or unloaded"
   - B = *body motion*: "this parameter is used to analyze either vertical motions of the body or the actions necessary to overcome an obstruction or impairment to body movement"
   - G = *gain control*: "this parameter is used to analyze all manual motions (mainly finger, hand and foot) employed to obtain complete manual control of an object and release the object after placement"
   - P = *placement*: "this parameter is used to analyze actions at the final stage of an object's displacement to align, orient and or engage the object with another object before control of it is relinquished".

In particular, since one of the objectives of the VLFT project is to create a link between this methodology and the OCRA kpis estimation methodology, and for the sake of clarity of the letter sequence, it has been assaigned a univocal *sub-activity name* to each single letter: **A** —> move empty **B** —> body motion **G** —> pick object **A** —> move object **B** —> body motion **P** —> place object **A** —> move back in position
By looking at the descriptions in the cells of the table it is possible to identify for each letter which 'box' is the more adapted for the case under consideration.

2. in Figure 3 and 4 the '***Tool use sequence model*** tables are shown:
   Figure 3 - Images-MOST/ugim4.PNG
   Figure 4 - Images-MOST/ugim5.PNG
   "The Tool Use Sequence Model is comprised of phases and sub-activities from the General Move Sequence Model, along with specially designed parameters describing the actions performed with hand tools or, in some cases, mental processes required when using the senses as a tool". (Kjell B. Zandin - MOST Work Measurement Systems)
   So in this section the activities done with the aid of a tool are analysed, and the general sequence of letters is the foolowing: **A B G A B P * A B P A**, where:
   - A = *action distance*: "this parameter is used to analyze all spatial movements or actions of the fingers, hands and/or feet, either loaded or unloaded"
   - B = *body motion*: "this parameter is used to analyze either vertical motions of the body or the actions necessary to overcome an obstruction or impairment to body movement"
   - G = *gain control*. "this parameter is used to analyze all manual motions (mainly finger, hand and foot) employed to obtain complete manual control of an object and release the object after placement"
   - P = *placement*: "this parameter is used to analyze actions at the final stage of an object's displacement to align, orient and or engage the object with another object before control of the object is relinquished"
     For what regards the asterisk in the sequence, it has to be replaced with one of the following letters, refered to the specific tool actions applied:
   - F/L = *Fasten or Loosen*: this parameter is used to establish the time for manually or mechanically assembling/disassembling one object to another, using the fingers, hand or a hand tool"
   - C = *Cut*: "this parameter covers the manual actions employed to separate, divide or remove part of an object using a sharp-edged hand tool such as pliers, scissors or a knife"
   - S = *Surface treat*: "this parameter covers the activities aimed at removing unwanted material or particles from, or applying a substance, coating or finish to,

the surface of an object"

- M = *Measure*: "this parameter includes the actions employed in determining a certain physical characteristic of an object by using a standard measuring device"
- R = *Record*: "this parameter covers the manual actions performed with a pencil, pen, marker, chalk or other marking tool for the purpose of recording information"
- T = *Think*: "this parameter refers to the eye actions and mental activity employed to obtain information (read) or to inspect an object, including reaching to touch, when necessary, to feel the object"

  In particular, since one of the objectives of the VLFT project is to create a link between this methodology and the OCRA kpis estimation methodology, and for the sake of clarity of the letter sequence, it has been assaigned a univocal *sub-activity name* to each single letter: **A** —> move empty **B** —> body motion **G** —> pick object **A** —> move object **B** —> body motion **P** —> place object **F/L - C - S - R - M - T** —> tool action **A** —> move object **B** —> body motion **P** —> place object **A** —> move back in position

  By looking at the descriptions in the cells of the table it is possible to identify for each letter which 'box' is the more adapted for the case under consideration. Of course for the letters A, B, G and P, the previous table related to the general move has to be considered for the choice.

3. in Figure 5 is finally showed the '***Controlled move sequence model*** table:
   Figure 5 - Images-MOST/ugim6.PNG
   "Controlled Move describes the manual displacement of an object over a 'controlled' path. That is, movement of the object is restricted in at least one direction by contact with or attachment to another object or the nature of the work demands that the object be deliberately moved along a specific or controlled path". (Kjell B. Zandin - MOST Work Measurement Systems) Also here a general sequence of letters related to different sub-activities is considered: **A B G M X I A**, where:
   - A = *action distance*: "this parameter is used to analyze all spatial movements or actions of the fingers,vhands and/or feet, either loaded or unloaded"
   - B = *body motion*: "this parameter is used to analyze either vertical motions of the body or the actions necessary to overcome an obstruction or impairment to body movement"
   - G = *gain control*. "this parameter is used to analyze all manual motions (mainly finger, hand and foot) employed to obtain complete manual control of an object and release the object after placement"
   - M = *Move controlled*: "this parameter is used to analyze all manually guided movements or actions of an object over a controlled path"

- X = *Process time*: "this parameter is used to account for the time for work controlled by electronic or mechanical devices or machines, not by manual actions"
- I = *Alignment*: "this parameter is used to analyze manual actions following the Move Controlled or at the conclusion of Process Time to achieve the alignment of objects"
  In particular, since one of the objectives of the VLFT project is to create a link between this methodology and the OCRA kpis estimation methodology, and for the sake of clarity of the letter sequence, it has been assaigned a univocal *sub-activity name* to each single letter: **A** –-> move empty **B** –-> body motion **G** –-> pick object **M** –-> move in contact **X** –-> process time **I** –-> align object **A** –-> move back in position
  By looking at the descriptions in the cells of the table it is possible to identify for each letter which 'box' is the more adapted for the case under consideration. Of course for the letters A, B and G, the previous table related to the general move has to be considered for the choice.

To conclude this part, it is important to specify that for a complete understanding of the methodology and so of the way in which the activities have to be considered, it is possible to look at the "MOST Work Measurement Systems" book by Kjell B. Zandin. After this initial assessment it is possible now to go and see how the concrete application of the methodology works.

Formalise the process

Prepare the assessment

In this first phase the assessor needs to well analyse and understand the process to be assessed, first by his own and then by interviewing the worker, so to have a clear and complete view of the situation. The main things to be determined and defined are:

- the main operations of the work at issue;
- the typology of each operation identified;
- the single steps in which each operation considered is diveded.

The fundamental base in order to accomplish these three points is represented by the following tables, related to three main sections of this methodology:

1. in Figure 2 it is possible to see the '***General move sequence model*** table:
   Figure 2 - Images-MOST/ugim3.PNG
   "General Move deals with the spatial displacement of one or more objects. Under manual control, the object follows an unrestricted path through the air. If the object is in contact with, restricted by or attached to another object during the move, the General Move Sequence Model is not applicable". (Kjell B. Zandin - MOST Work Measurement Systems)
   Here it can be seen that the general sequence of letters describing the considered subactivities is made up by: **A B G A B P A**, where:
   - A = *action distance*: "this parameter is used to analyze all spatial movements or actions of the fingers, hands and/or feet, either loaded or unloaded"
   - B = *body motion*: "this parameter is used to analyze either vertical motions of the body or the actions necessary to overcome an obstruction or impairment to body movement"
   - G = *gain control*: "this parameter is used to analyze all manual motions (mainly finger, hand and foot) employed to obtain complete manual control of an object and release the object after placement"
   - P = *placement*: "this parameter is used to analyze actions at the final stage of an object's displacement to align, orient and or engage the object with another object before control of it is relinquished".

In particular, since one of the objectives of the VLFT project is to create a link between this methodology and the OCRA kpis estimation methodology, and for the sake of clarity of the letter sequence, it has been assaigned a univocal *sub-activity name* to each single letter: **A** —> move empty **B** —> body motion **G** —> pick object **A** —> move object **B** —> body motion **P** —> place object **A** —> move back in position
By looking at the descriptions in the cells of the table it is possible to identify for each letter which 'box' is the more adapted for the case under consideration.

2. in Figure 3 and 4 the '***Tool use sequence model*** tables are shown:
   Figure 3 - Images-MOST/ugim4.PNG
   Figure 4 - Images-MOST/ugim5.PNG
   "The Tool Use Sequence Model is comprised of phases and sub-activities from the General Move Sequence Model, along with specially designed parameters describing the actions performed with hand tools or, in some cases, mental processes required when using the senses as a tool". (Kjell B. Zandin - MOST Work Measurement Systems)
   So in this section the activities done with the aid of a tool are analysed, and the general sequence of letters is the foolowing: **A B G A B P * A B P A**, where:
   - A = *action distance*: "this parameter is used to analyze all spatial movements or actions of the fingers, hands and/or feet, either loaded or unloaded"
   - B = *body motion*: "this parameter is used to analyze either vertical motions of the body or the actions necessary to overcome an obstruction or impairment to body movement"
   - G = *gain control*. "this parameter is used to analyze all manual motions (mainly finger, hand and foot) employed to obtain complete manual control of an object and release the object after placement"
   - P = *placement*: "this parameter is used to analyze actions at the final stage of an object's displacement to align, orient and or engage the object with another object before control of the object is relinquished"
     For what regards the asterisk in the sequence, it has to be replaced with one of the following letters, refered to the specific tool actions applied:
   - F/L = *Fasten or Loosen*: this parameter is used to establish the time for manually or mechanically assembling/disassembling one object to another, using the fingers, hand or a hand tool"
   - C = *Cut*: "this parameter covers the manual actions employed to separate, divide or remove part of an object using a sharp-edged hand tool such as pliers, scissors or a knife"
   - S = *Surface treat*: "this parameter covers the activities aimed at removing unwanted material or particles from, or applying a substance, coating or finish to,

the surface of an object"

- M = *Measure*: "this parameter includes the actions employed in determining a certain physical characteristic of an object by using a standard measuring device"
- R = *Record*: "this parameter covers the manual actions performed with a pencil, pen, marker, chalk or other marking tool for the purpose of recording information"
- T = *Think*: "this parameter refers to the eye actions and mental activity employed to obtain information (read) or to inspect an object, including reaching to touch, when necessary, to feel the object"

  In particular, since one of the objectives of the VLFT project is to create a link between this methodology and the OCRA kpis estimation methodology, and for the sake of clarity of the letter sequence, it has been assaigned a univocal *sub-activity name* to each single letter: **A** —> move empty **B** —> body motion **G** —> pick object **A** —> move object **B** —> body motion **P** —> place object **F/L - C - S - R - M - T** —> tool action **A** —> move object **B** —> body motion **P** —> place object **A** —> move back in position

  By looking at the descriptions in the cells of the table it is possible to identify for each letter which 'box' is the more adapted for the case under consideration. Of course for the letters A, B, G and P, the previous table related to the general move has to be considered for the choice.

3. in Figure 5 is finally showed the '***Controlled move sequence model*** table:
   Figure 5 - Images-MOST/ugim6.PNG
   "Controlled Move describes the manual displacement of an object over a 'controlled' path. That is, movement of the object is restricted in at least one direction by contact with or attachment to another object or the nature of the work demands that the object be deliberately moved along a specific or controlled path". (Kjell B. Zandin - MOST Work Measurement Systems) Also here a general sequence of letters related to different sub-activities is considered: **A B G M X I A**, where:
   - A = *action distance*: "this parameter is used to analyze all spatial movements or actions of the fingers,vhands and/or feet, either loaded or unloaded"
   - B = *body motion*: "this parameter is used to analyze either vertical motions of the body or the actions necessary to overcome an obstruction or impairment to body movement"
   - G = *gain control*. "this parameter is used to analyze all manual motions (mainly finger, hand and foot) employed to obtain complete manual control of an object and release the object after placement"
   - M = *Move controlled*: "this parameter is used to analyze all manually guided movements or actions of an object over a controlled path"

- X = *Process time*: "this parameter is used to account for the time for work controlled by electronic or mechanical devices or machines, not by manual actions"
- I = *Alignment*: "this parameter is used to analyze manual actions following the Move Controlled or at the conclusion of Process Time to achieve the alignment of objects"
  In particular, since one of the objectives of the VLFT project is to create a link between this methodology and the OCRA kpis estimation methodology, and for the sake of clarity of the letter sequence, it has been assaigned a univocal *sub-activity name* to each single letter: **A** –-> move empty **B** –-> body motion **G** –-> pick object **M** –-> move in contact **X** –-> process time **I** –-> align object **A** –-> move back in position
  By looking at the descriptions in the cells of the table it is possible to identify for each letter which 'box' is the more adapted for the case under consideration. Of course for the letters A, B and G, the previous table related to the general move has to be considered for the choice.

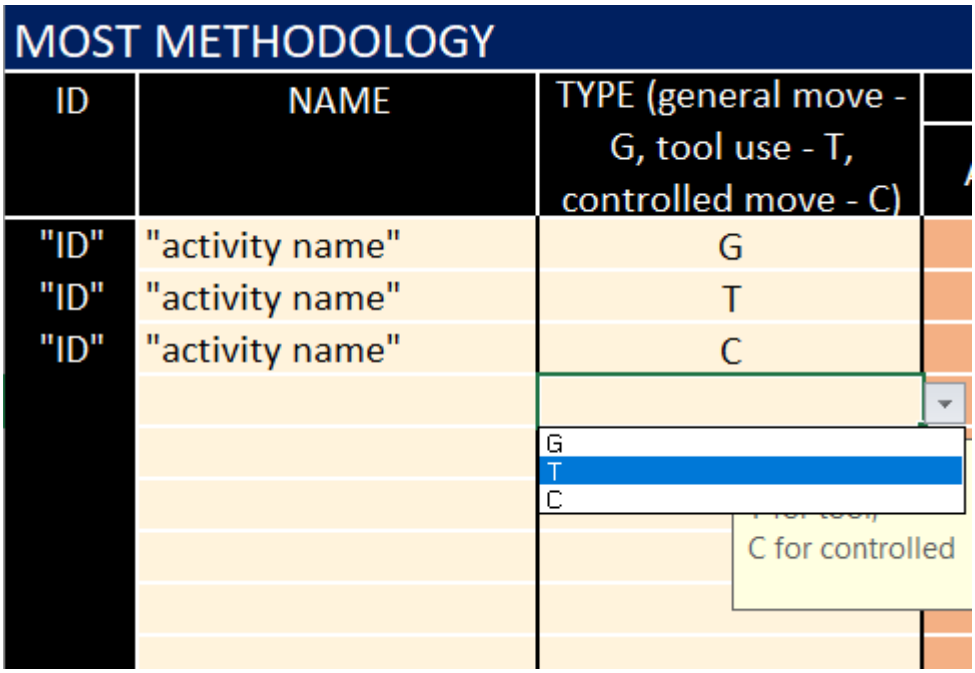
To conclude this part, it is important to specify that for a complete understanding of the methodology and so of the way in which the activities have to be considered, it is possible to look at the "MOST Work Measurement Systems" book by Kjell B. Zandin. After this initial assessment it is possible now to go and see how the concrete application of the methodology works.

Prepare the input data

Before analyzing the methodology in detail, it is important to show the structure of the following explanation: first of all a general approach will be adopted, analysing the general templates of the input files, explaining all the single steps in order to fill in it; secondly a specific example will be proposed to show practically the execution of the described steps and to have a real case of application.

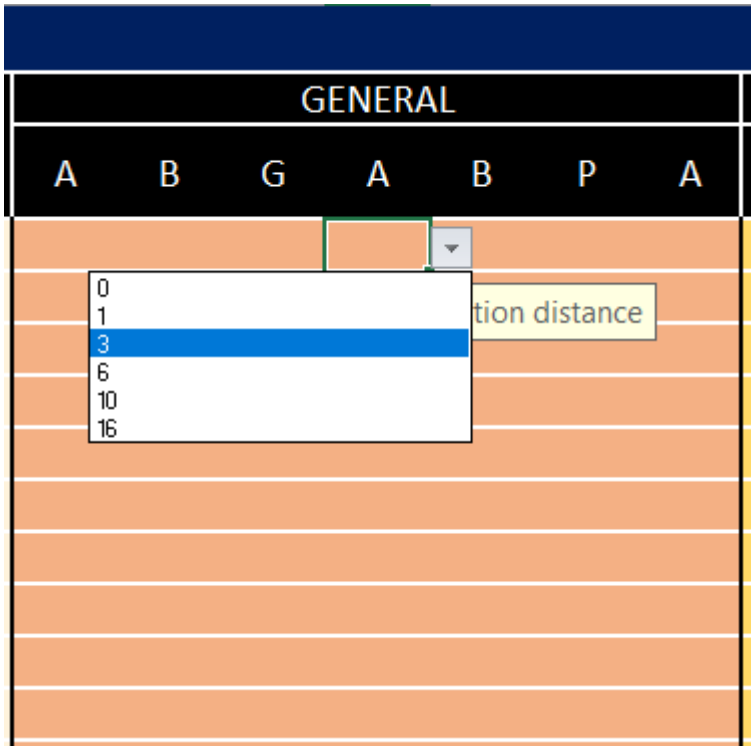
2.1 Excel input file

The input of the procedure will be as already said an Excel file, whose general model can be found by following the path (folders): PROCESSING TIME ESTIMATION - MOST - PYTHON CODE, with the name 'MOSTgen.xlsx'. To start the evaluation of a completely new process, it is necessary to 'copy and paste' this Excel file, that is the model, in the same folder where the python code will be run. It is advisable to change the name of the file with a name related to the process to be analysed, clear, univocal and without spaces. After this, it is possible to go into the details of this file, where there is only one sheet to be filled in: its name is '**MOST**', and it contains many different sections including many different columns to be filled in, that can be analysed by considering 4 different sections. The first one is comprehensive of the first three columns of the table, as shown in Figure 6:



1. the first is named **ID** and it requires the ID of the activity considered, that has to be a univocal number, put in the order of execution, starting from 1 until a maximum of 100, that is the maximum number of activities that this toolkit can analyse in a single process;
2. the second column is for the **NAME**, that requires a complete, well identifying and univocal name of the activity considered (if one name is repeated twice, this means that the activities are exactly the same);
3. the third column, **TYPE**, is very important since it provides the selection of the section to which each single activity is related, like shown in the previous chapter: there are three options, '**G**' for general move, '**T**' for tool use and '**C**' for controlled move; the choice can be done by selecting the letters in the drop down menu.

The next three sections to be explained are of course related to the main divisions of the methodology: so starting from Figure 7 it can be seen the *general move sequence model* table;

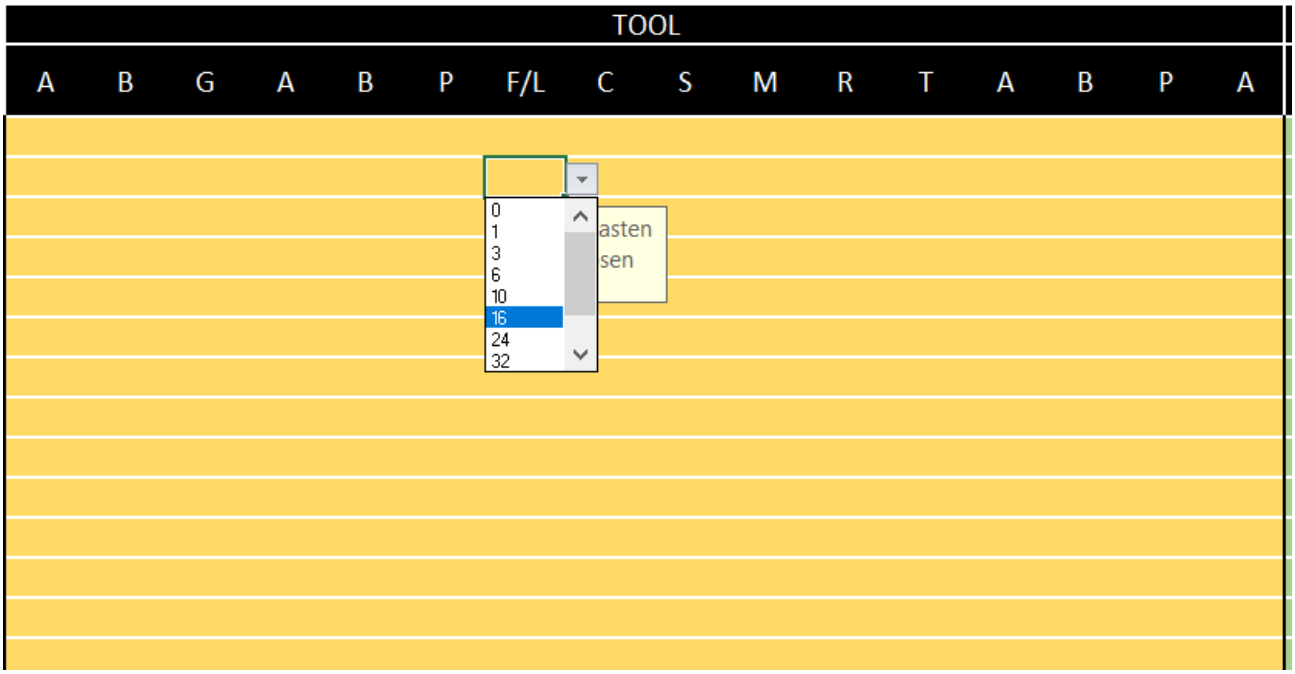


First of all the lines of this section have to be completed only if the Type Code of the considered activity row is 'G': if it is different, the respective line must be left blank, without selecting any code from the drop down menu. If instead it has to be filled in, the assessor needs to insert the right number (1-3-6-10-16) of the list by looking at the MOST table seen before and reported hereafter (Figure 2), one for each column, correspondent to the

different subactivities: if one subactivity is evaluated as not necessary for that specific action, the number 0 has to be selected in the cell.

BasicMOST® System	General Move				A B G A B P A
Index x 10	A Action Distance	B Body Motion	G Gain Control	P Placement	Index x 10
0	≤ 2 in. (5 cm)			Pickup Toss	0
1	Within Reach		GRASP: Light Object Light Objects Simo	PUT: Lay Aside Loose Fit	1
3	1 – 2 Steps	Sit or Stand Bend and Arise 50% occ.	GET: Light Objects Non-Simo Heavy or Bulky Blind or Obstructed Disengage Interlocked Collect	PLACE: Loose Fit Blind or Obstructed Adjustments Light Pressure Double Placement	3
6	3 – 4 Steps	Bend and Arise		POSITION: Care or Precision Heavy Pressure Blind or Obstructed Intermediate Moves	6
10	5 – 7 Steps	Sit or Stand with Adjustments			10
16	8 – 10 Steps	Stand and Bend Bend and Sit Climb On or Off Through Door			16

The next section of the table, as shown in Figure 8, is related to the *tool use sequence model*:



First of all the lines of this section have to be completed only if the Type Code of the considered activity row is 'T': if it is different, the respective line must be left blank, without

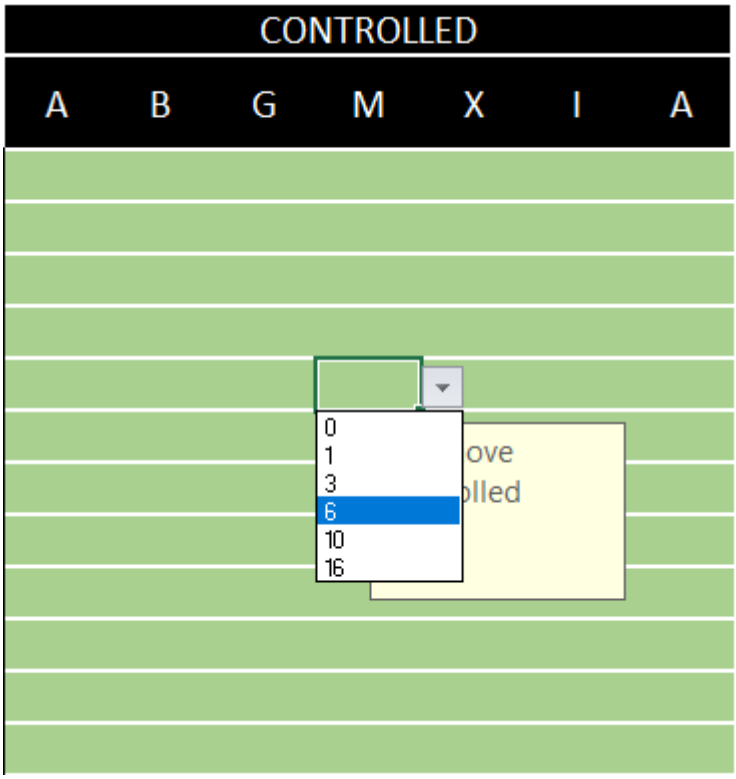
selecting any code from the drop down menu. If instead it has to be filled in, a distinction has to be done between two different group of letters:

1. for what regards the columns A, B, G and P the assessor needs to insert the right number in the list 1-3-6-10-16, by looking at the MOST table about the general move shown above (Figure 2), one for each column, correspondent to the different subactivities: if one subactivity is evaluated as not necessary for that specific action, the number 0 has to be selected in the cell;
2. for what regards instead the columns F/L, C. S, R, M and T the assessor needs to insert the right number from the list 1-3-6-10-16-24-32-54, by looking at the MOST tables seen before about the tool use and reported hereafter (Figure 3-4), one for each column, correspondent to the different subactivities: if one subactivity is evaluated as not necessary for that specific action, the number 0 has to be selected in the cell: it has to be specified that generally only one of this cells will contain a value different than zero, for each activity.

BasicMOST® System	Tool Use							A B G A B P *			A B P A
Index x 10	F Fasten or L Loosen										Index x 10
	Finger Action	Wrist Action				Arm Action					Power Tool	
	Spins	Turns	Strokes	Cranks	Taps	Turns		Strokes	Cranks	Strikes	Screw Diam.	
	Fingers, Screwdriver	Hand, Screwdriver, Ratchet, T-Wrench	Wrench	Wrench, Ratchet	Hand, Hammer	Ratchet	T-Wrench, 2-Hands	Wrench	Wrench, Ratchet	Hammer	Power Wrench	
1	1	-	-	-	1	-	-	-	-	-	-	**1**
3	2	1	1	1	3	1	-	1	-	1	1/4 in. (6 mm)	**3**
6	3	3	2	3	6	2	1	-	1	3	1 in. (25 mm)	**6**
10	8	5	3	5	10	4	-	2	2	5		**10**
16	16	9	5	8	16	6	3	3	3	8		**16**
24	25	13	8	11	23	9	6	4	5	12		**24**
32	35	17	10	15	30	12	8	6	6	16		**32**
42	47	23	13	20	39	15	11	8	8	21		**42**
54	61	29	17	25	50	20	15	10	11	27		**54**

BasicMOST® System			Tool Use						A B G A B P * A B P A						
Index x 10	C Cut				S Surface Treat			M Measure	R Record			T Think			Index x 10
	Cutoff	Secure	Cut	Slice	Air-Clean	Brush-Clean	Wipe	Measure	Write		Mark	Inspect	Read		
	Pliers		Scissors	Knife	Nozzle	Brush	Cloth	Measuring Tool	Pencil/Pen		Marker	Eyes/ Fingers	Eyes		
	Wire		Cuts	Slices	sq. ft. (0.1 m^2)	sq. ft. (0.1 m^2)	sq. ft. (0.1 m^2)		Digits	Words	Digits	Points	Digits, Single Words	Text of Words	
1		Grip	1	-	-	-	-		1	-	Check Mark	1	1	3	1
3	Soft		2	1	-	-	1/2		2	-	1 Scribe Line	3	3 Gauge	8	3
6	Medium	Twist Form Loop	4	-	1 Spot Cavity	1	-		4	1	2	5 Feel for Heat	6 Scale Value Date or Time	15	6
10	Hard		7	3	-	-	1	Profile Gauge	6	-	3	9 Feel for Defect	12 Vernier Scale	24	10
16		Secure Cotter Pin	11	4	3	2	2	Fixed Scale Caliper ≤ 12 in. (30 cm)	9 Signature or Date	2	5	14	Table Value	38	16
24			15	6	4	3	-	Feeler Gauge	13	3	7	19		54	24
32			20	9	7	5	5	Steel Tape ≤ 6 ft. (2 m) Depth Micrometer	18	4	10	26		72	32
42			27	11	10	7	7	OD-Micrometer ≤ 4 in. (10 cm)	23	5	13	34		94	42
54			33					ID-Micrometer ≤ 4 in. (10 cm)	29	7	16	42		119	54

The next and last section of the table, shown in Figure 9, is related to the *controlled move sequence model*:



First of all the lines of this section have to be completed only if the Type Code of the considered activity row is 'C': if it is different, the respective line must be left blank, without selecting any code from the drop down menu. If instead it has to be filled in, a distinction has to be done between two different group of letters:

1. for what regards the columns A, B, and G the assessor needs to insert the right number in the list 1-3-6-10-16, by looking at the MOST table about the general move shown above (Figure 2), one for each column, correspondent to the different subactivities: if one subactivity is evaluated as not necessary for that specific action, the number 0 has to be selected in the cell;
2. for what regards instead the columns M, X and I the assessor needs to insert the right number from the list 1-3-6-10-16, by looking at the MOST table seen before about the controlled move and reported hereafter (Figure 5), one for each column, correspondent to the different subactivities: if one subactivity is evaluated as not necessary for that specific action, the number 0 has to be selected in the cell.

BasicMOST® System	Controlled Move					A B G M X I A
Index x 10	M Move Controlled		X Process Time			I Alignment	Index x 10
	Push/Pull/Turn	Crank	Seconds	Minutes	Hours		
1	≤ 12 in. (30 cm) Button Switch Knob		.5 Sec.	.01 Min.	.0001 Hr.	1 Point	1
3	> 12 in. (30 cm) Resistance Seat or Unseat High Control 2 Stages ≤ 24 in. (60 cm) Total	1 Rev.	1.5 Sec.	.02 Min.	.0004 Hr.	2 Points ≤ 4 in. (10 cm)	3
6	2 Stages > 24 in. (60 cm) Total 1 – 2 Steps	2 – 3 Rev.	2.5 Sec.	.04 Min.	.0007 Hr.	2 Points > 4 in. (10 cm)	6
10	3 – 4 Stages 3 – 5 Steps	4 – 6 Rev.	4.5 Sec.	.07 Min.	.0012 Hr.		10
16	6 – 9 Steps	7 – 11 Rev.	7.0 Sec.	.11 Min.	.0019 Hr.	Precision	16

As a general statement, as already said in the limitations of this methodology, it is important not to have redundancy in the selection of the numbers, putting attention in not to replicate some sub-activities in different macro-activities, and so avoiding an overestimation of the processing times. To better explain this, in the Figure 10 is shown a tyical error in the assaignment, that if done along the whole cycle it can lead to a considerable rise of the total time:

		G, tool use - T, controlled move - C)	A	B	G	A	B	P	A
1	pick up tool A	G	1	0	1	1	0	3	0
2	unclamp fixture A	T							
3	unclamp fixture B	T							
4	release tool A	G	0	0	0	1	0	1	1
5	unload WIP 1	G	1	0	3	1	0	1	0
6	load raw part	G	3	3	3	3	0	3	0
7	pick up tool A	G	1	0	1	1	0	3	0

The two cells highlighted in yellow are actually related to the same movement: in fact, the 1 put in the last column of the 'release tool A' activity would mean that the operator moves back in the base position after the movement, while the 1 put in the first column of the 'unload WIP 1' activity would mean that the operator moves from the base position to the position of the WIP 1: this is an unuseful repetition, since the operator concretely doesn't need to move back in position after having realesed the tool, for then moving again from the base position to the next position to be reached; the real movement is that of moving directly after having realesed the tool to the position of the WIP1, selecting in this way only the 1 of the second activity and deleting that of the previous. Before passing to the analysis of the Python code, an example to clarify the approach is showed.

2.1.1 Example (Catenaccio case - Layout 2)

The example provided from the layout 2 of the Catenaccio case analysed in the VLFT project is very important to understand how to practically apply this part of the toolkit, and to clearly get the flow of reasoning required by it. In the Figures 11, 12, 13 and 14 the just described sections of the Excel input file table are shown, properly filled by the assessor with all the data needed.

MOST METHODOLOGY

ID	NAME	TYPE (general move - G, tool use - T, controlled move - C)	A
1	pick up tool A	G	1
2	hold part		
3	unclamp fixture A		
4	unclamp fixture B		
5	release tool A	G	0
6	unload WIP 1	G	1
7	load raw part	G	3
8	pick up tool A	G	1
9	hold part	C	
10	clamp fixture A	T	
11	clamp fixture B	T	
12	release tool A	G	0
13	activate pallet rotation	C	
14	pick up tool B	G	1
15	hold part	C	
16	unclamp fixture A	T	
17	unclamp fixture B	T	
18	unclamp fixture C	T	
19	release tool B	G	0
20	unload WIP 2	G	1
21	load WIP 1	G	1
22	pick up tool B	G	1

G
T
C

C for controlled

MOST

GENERAL						
A	B	G	A	B	P	A
1	0	1	1	0	3	0
0	0	0	1	0	1	0
1	0	1	1			
3	3	1	6		3	0
1	0	1	1		3	0
0	0	0	1	0	1	0
1	0	1	1	0	3	0
0	0	0	1	0	1	0
1	0	1	1	0	1	0
1	0	1	1	0	3	0
1	0	1	1	0	3	0

0
1
3
6
10
16

dy motion

TOOL															
A	B	G	A	B	P	F/L	C	S	M	R	T	A	B	P	A
0	0	0	0	0	0	3	0	0	0	0	0	0	0	3	0
0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	3	0	0	0	0	0	0	0	3	0
0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	3	0	0	0	0	0	0	0	3	0
0	0	0	0	0	0	3	0	0	0	0	0	0	0	3	0
0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0

3
6
10
16
24
32
42
54

asten
sen

CONTROLLED						
A	B	G	M	X	I	A
1	0	0	1	0	0	0
1	0	0	1	0	0	0
1	0	0	1	1	0	0
1	0	0	1	0	0	0

0
1
3
6
10
16

ove
olled

Execution and results

Python code

A technical remark needs to be done before describing the procedure: the Excel sheets taken by the Python as an input are different than the use-interface sheets, the ones filled in by the assessor: in fact, they are directly linked to dual hidden sheets in the same file, structured in a way that is more easilly readable by the code. The Python code has two different versions, the first one named '**MOST_OCRA input loading.ipynb**', an extended version divided in many cells, so to have the possibility to see all the intermediate results of the code, and the second one named '**MOST_OCRA input loading compressed.ipynb**', where the entire code is put in one cell, so to have the possibility to quickly run it without taking care about the details. In both the cases the objective is to take the input activities in the sheet "*MOST*" of the Excel file just described, starting from the 'Type code' column, where the code understand which of the three sections it has to consider for that specific activity (G-T-C); basing on this, the next step is that of reading the correspondent numerical values inserted in the cells by the assessor and if the number is different than zero, a sub-activity is formed: for the specific scope of this link-code, a set of sub-activity names has been created, as already seen in the chapter 1 of this guide, and reported hereafter in the Table 1:

General move		Tool use		Controlled move	
A	move empty	A	move empty	A	move empty
B	body motioin	B	body motion	B	body motion
G	pick object	G	pick object	G	pick object
A	move object	A	move object	M	move in contact
B	body motion	B	body motion	X	process time
P	place object	P	place object	I	align object

A	move back in position	F/L-C-S-M-R-T	tool action	A	move back in position
		A	move object		
		B	body motion		
		P	place object		
		A	move back in position		

So basing on this information and considering the cell at issue, the right name is found and stored to be given as an output. In the same time, the code also analyse the second input sheet, "*Objects_weight*", to find and link to the sub-activity just created the respective weight range of the tool or part used, if there is one. The last thing done by the code is to organize all the information gathered in order to give them as an Excel output easilly readable and compliant with the OCRA methodology: in fact the output file now is not again the input one, but it is a new file that is the Excel input of the OCRA methodology. Anyway the specific structure will be shown in the next chapter. For all the details regarding the python code it is possible to look directly at it, and specially at the comments done as an explanation of all the single steps.

In order to run the model, it was used ***Microsoft Azure Notebooks*** program with Python3.6 version.

Please, follow the next **mandatory** steps:

- First of all it is necessary to open the code (the extention is .ipynb), and insert the right Excel file name in the first and the last cell of it, as shown in Figures 21 and 22:

```
In [26]: # Introduction of the main libraries (numpy and pandas) and loading of the 'MOSTpy' sheet from the excel file
import numpy as np
import pandas as pd
file = 'MOST_OCRAcatenaccio_s4_layout1.xlsx'
cycle = pd.read_excel(file,
sheet_name='MOSTpy',
header=0,
index_col=False,
keep_default_na=True)
cycle
```

```
In [27]: # Introduction of the main libraries (numpy and pandas) and loading of the 'Objects_weight' sheet from the exce
cycles = pd.read_excel(file,
sheet_name='Objects_weightpy',
header=0,
index_col=False
```

```
In [30]: # Final export of the dataframe in the excel file MOSTcatenaccio as an output of this code
from openpyxl import load_workbook
from openpyxl.styles.borders import Border, Side
from openpyxl.styles import Font

book = load_workbook('OCRAinput.xlsx')
writer = pd.ExcelWriter('OCRAinput.xlsx', engine = 'openpyxl')
writer.book = book
output.to_excel(writer, sheet_name='FPRpy')
# The following rows are written for defining the dimensions of the cells to be printed as an out
writer.sheets['FPRpyprova3'].column_dimensions['A'].width = 15
writer.sheets['FPRpyprova3'].column_dimensions['B'].width = 30
writer.sheets['FPRpyprova3'].column_dimensions['C'].width = 30
writer.sheets['FPRpyprova3'].column_dimensions['D'].width = 30
writer.sheets['FPRpyprova3'].column_dimensions['E'].width = 30
writer.sheets['FPRpyprova3'].column_dimensions['F'].width = 30
writer.sheets['FPRpyprova3'].column_dimensions['G'].width = 30
writer.save()
```

- The code must be run step by step in the same order in which it was developed if the extended version is choosen, while for the compressed version only one run command is needed;
- Warning: the code is tailor made for this methodology, so be careful in modifying it, since all the modifications needs to be extended to the whole code in order to make it funciton;
- Anyway this is an open-source toolkit, so any improvement to the code is very appreciated, following the objective of the VLFT project.

Observations:

- The order in which the Input File was fulfilled, does not affect the result of the code.
- If after running the model, this generates a message of error on Python, please go back to the Input File sheet and check that the data inserted are totally compliant with the guidelines expressed in this guide.

Link OCRA-MOST Expected Outcomes

The final step of the model developed for the MOST-OCRA methodology is the downloading of the output Excel file modified by the Python code with the desired output: as already mentioned it will be different than the input one, in fact it will be the OCRA Excel input file, already structured with many sheets needed for that specific methodology. What this code does, is to create a new sheet in the file, called '**FPRpy**', where the subactivities and their related information will be displayed. To have a clearer explanation of this, it is possible to look at the Figure 23:

	TASK CODE	TASK NAME	TECHNICAL ACTION	MOVE TYPE	ACTION DURATION
0	1	pick up tool A	Move empty	1	0.36
1	1	pick up tool A	Pick Light Object	3	0.36
2	1	pick up tool A	Move Light Object	6	0.36
3	1	pick up tool A	Place Light Object	9	1.08
4	2	unclamp fixture A	Light Tool Action	13	1.08
5	2	unclamp fixture A	Place Light Object	9	1.08
6	3	unclamp fixture B	Light Tool Action	13	1.08
7	4	release tool A	Move Light Object	6	0.36
8	4	release tool A	Place Light Object	9	0.36
9	5	unload WIP 1	Move empty	1	0.36
10	5	unload WIP 1	Pick Medium Object	4	1.08
11	5	unload WIP 1	Move Medium Object	7	0.36
12	5	unload WIP 1	Place Medium Object	10	0.36
13	6	load raw part	Move empty	1	1.08
14	6	load raw part	Body motion	2	1.08
15	6	load raw part	Pick Medium Object	4	1.08
16	6	load raw part	Move Medium Object	7	1.08
17	6	load raw part	Place Medium Object	10	1.08
18	7	pick up tool A	Move empty	1	0.36
19	7	pick up tool A	Pick Light Object	3	0.36
20	7	pick up tool A	Move Light Object	6	0.36
21	7	pick up tool A	Place Light Object	9	1.08
22	8	clamp fixture A	Light Tool Action	13	1.08
23	8	clamp fixture A	Place Light Object	9	1.08
24	9	clamp fixture B	Light Tool Action	13	1.08

Shift data | Shiftpy | FPRdata | PSTpy | AdMdata | FPRpy

1. In the first of the five columns, named '**TASK CODE**', the code of the macro-activity considered is inserted, to give to the operator the possibility to have an idea of which point the process is while considering the different subactivities: this is very important since new columns have to be filled in aside these, and the assessor need to now which macroactivities are going to be considered;

2. the second column is called '**TASK NAME**' and it is refered to the name of the macro-activity considered. It is directly related to the previous column, so it also has the same objective;
3. in the third column, '**TECHNICAL ACTION**', the sub-activities names are displayed, divided per macro-activity, in line with the requirements of the OCRA methodology;
4. in the column '**MOVE TYPE**' is put a code univocally indicating the sub-activity considered: two sub-activities called with the same name are considered as identical and so their Move Type will be the same;
5. the last column. '**ACTION DURATION**', actually provides the time duration of the specific sub-activity in seconds, basing on the MOST methodology just applied, which responds in a good way to the OCRA needs.

The final action required to the operator in order to set in the wright way these data is that of 'copying and paste' **as values** the five columns of the *FPRpy* sheet just analysed, in the first five columns of the sheet *FPRdata*, that are left black in the original format, as can be seen in Figure 24:

Task_Code	Task_Name	Technical_Action	Move_Type	Action_Duration	Force_Load	Shc
					0,5	
					1	
					2	
					0,5	
					5	
					1	
					0,5	
					5	
					2	
					1	
					0,5	
					0,5	
					2	
					3	
					2	
					2	
					3	
					1	
					0,5	
					0,5	
					1	
					2	
					0,5	
					5	
					1	

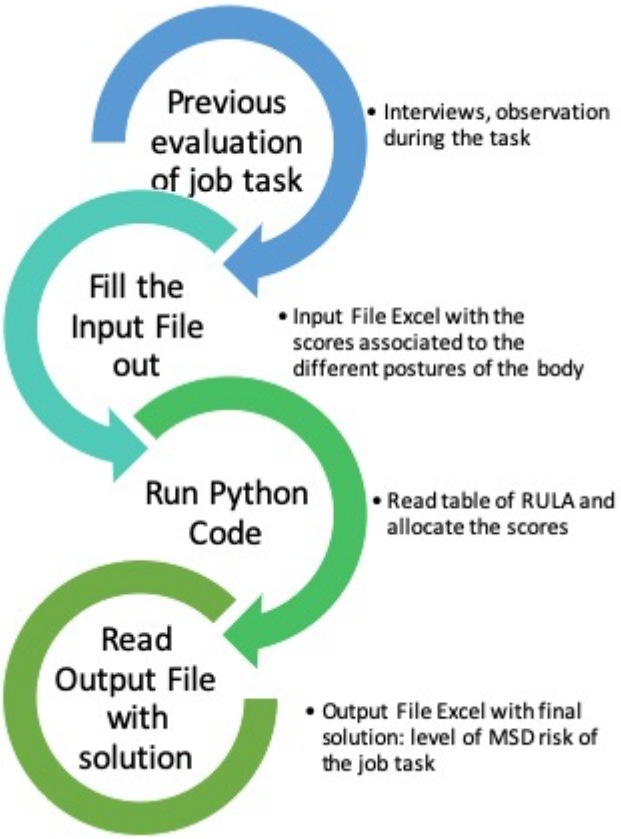
Shift data | Shiftpy | **FPRdata** | PSTpy | AdMdata | FPRpy

This is the final step for what regard the link between MOST and OCRA methodologies; of course now the implementation of the OCRA is required, and for this it is possible to look at the **OCRA User Guide** in this toolkit.

3.7 RULA

Introduction

The Rapid Upper Limb Assessment (RULA) is a tool that allow us to evaluate the level of ergonomic risk associated with the upper extremity postures incurring by a worker who performs repetitive job tasks. This tool was develop using Microsoft Excel and Python3 in order to avoid manual procedures to allocate certain scores on the different tables of the RULA methodology. This methodology assigns a range of scores for the different postures in upper extremities, neck, trunk and legs taking into account the force and repetition of the activity, based on the observation of the job task. Being an easy and quick method of ergonomic analysis, this tool does not require expensive equipment to complete the information needed to perform the assessment.



Limitations

The RULA is a practical method in terms of saving time and resources; however the precision of this tool could be affected by lack of information such as duration of the task,

available recovery time, or hand-arm vibration. Moreover, the tool may require several RULA assessments for one task. After the interview and observation of the worker, it is possible to determine the criticality of the job task: if only one arm should be evaluated or if the assessment is needed for both sides, that's why usually it is recommended choose the most extreme posture to evaluate.

Applications

RULA method is usually applied on sedentary tasks such as computer tasks, manufacturing or retail tasks where the worker is seated or standing without moving about.

How to start

Prepare the assessment

The evaluator should do a previous interview to the worker who is going to be evaluated in order to understand the job tasks and observe several times the movements and postures of the worker during the work cycles of the activity.

Related to the job task that should be evaluated, it is important to choose it with the following criteria:

- The most difficult postures and work tasks (given by the previous analysis)
- The posture performed for the longest period of time
- The posture where the highest force loads occurs.

After this previous analysis, it is possible to start the assessment of the level of MSD risk in a certain job task in the tool.

Formalise the process

Prepare the input data

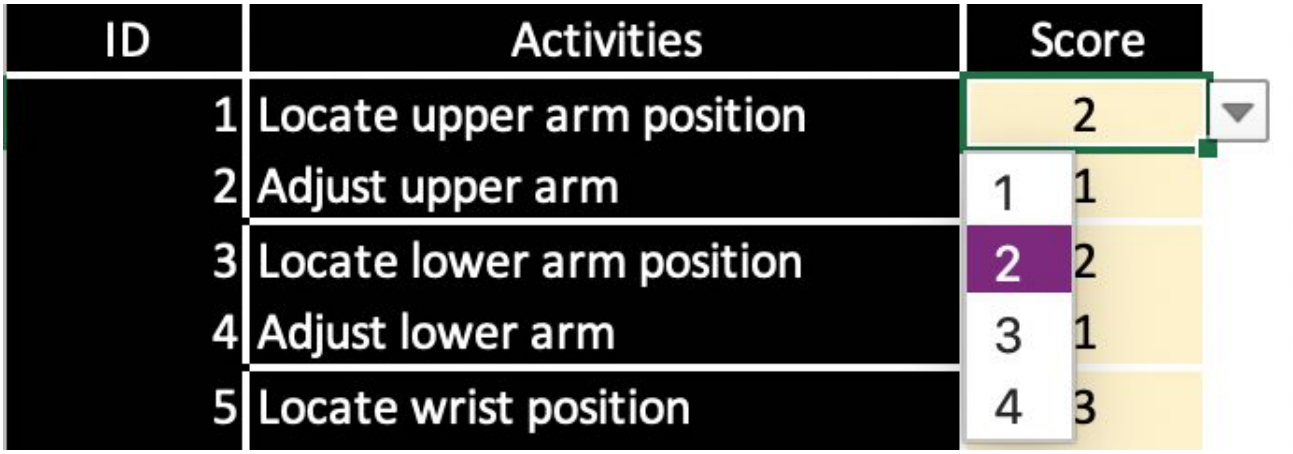
Input File

In the Excel Input File called "RULAInputFile.xlsx" (sheet "InputsRULA"), there is a list of postures that should be evaluated with a given posture scoring scale (an integer value). This list has an implicit segmentation in two parts: The first part is related to arm and wrist and the second one is related to neck, trunk and legs. These two parts have additional adjustments that should be part of the evaluation.

- **Locate upper arm position**

> For upper arm position, there is a range of scores between +1 and +4. These scores are related to the different positions of the upper arm performed in the job task chosen on the previous step, and this scoring scale could be seen in the Figure 2:

> After to choose the most suitable score for this posture, it should be introduce it on the "Locate upper arm position" - "Score" (Figure 3):



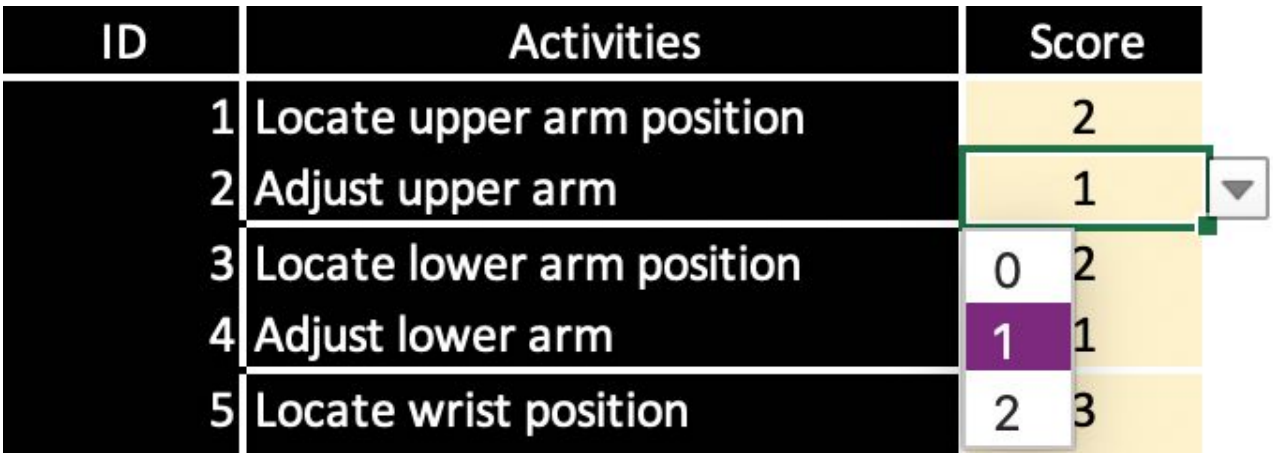
- **Adjust upper arm**

> In order to evaluate other aspects of the posture, an adjustment of the previous score should be made with the criteria given in Table 1:

Table 1:

Observation	Score
If shoulder is raised	+1
If upper arm is abducted	+1
If arm is supported or person is leaning	+0

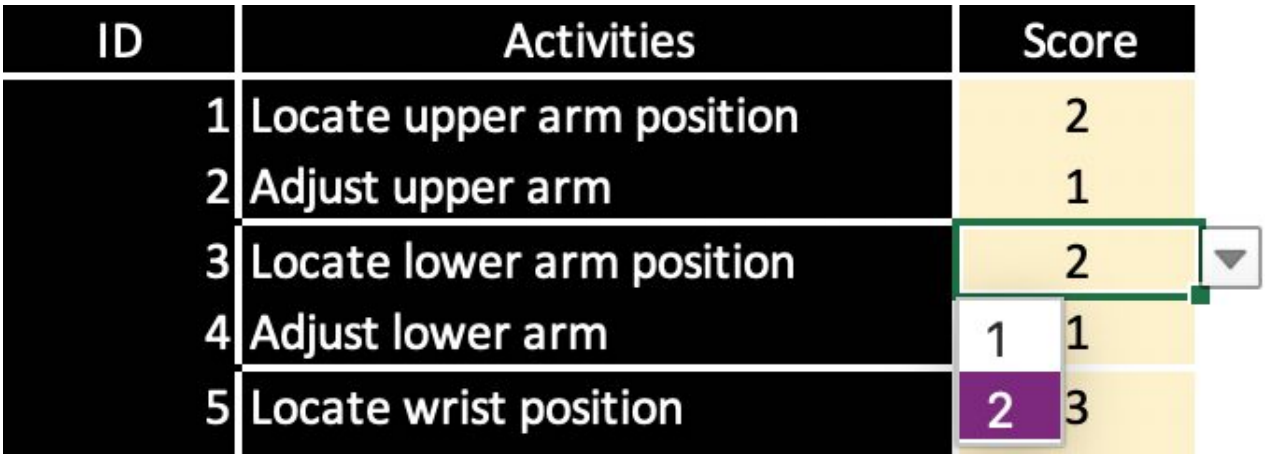
> After to choose the most suitable adjustment, it should be introduce it on the "Adjust upper arm" - "Score" (Figure 4). In case of no necessary adjustment, the score should be considered zero "0".



- **Locate lower arm position**

> For lower arm position, there is a range of scores between +1 and +2. These scores are related to the different positions of the lower arm performed in the job task chosen, and this scoring scale could be seen in the Figure 5:

> After to chose the most suitable score for this posture, it should be introduce it on the "Locate lower arm position" - "Score" (Figure 6):



- **Adjust lower arm**

> In order to evaluate other aspects of the posture, an adjustment of the previous score should be made with the criteria given in Table 2:

Table 2:

Observation	Score
If either arm is working across midline or out to side of body: Add	+1

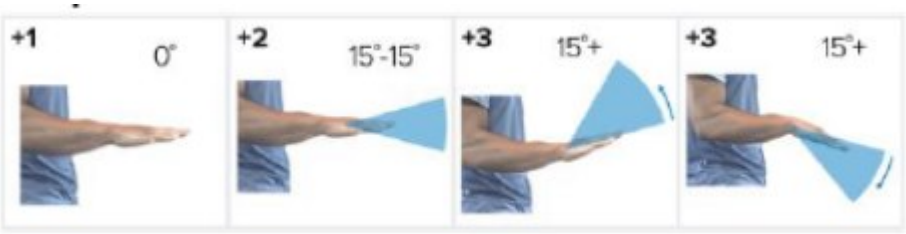
> After to choose the most suitable adjustment, it should be introduce it on the "Adjust lower arm" - "Score" (Figure 7). In case of no necessary adjustment, the score should be considered zero "0".

ID	Activities	Score
1	Locate upper arm position	2
2	Adjust upper arm	1
3	Locate lower arm position	2
4	Adjust lower arm	1
5	Locate wrist position	3
6	Adjust wrist	1

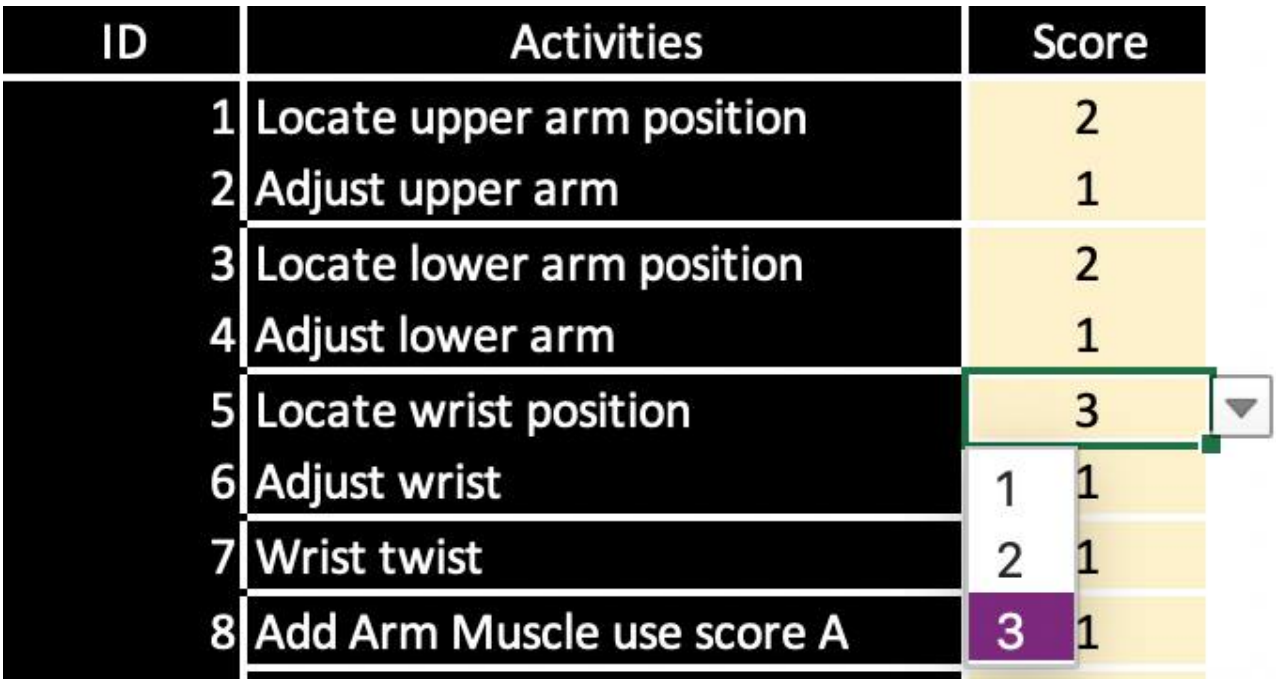
0
1

- **Locate wrist position**

> For locate wrist position, there is a range of scores between +1 and +3. These scores are related to the different positions of the wrist performed in the job task chosen and this scoring scale could be seen in the figure 8:



> After to chose the most suitable score for this posture, it should be introduce it on the "Locate wrist position" - "Score" (Figure 9):



- **Adjust wrist**

> In order to evaluate other aspects of the posture, an adjustment of the previous score should be made with the criteria given in Table 3:

Table 3:

Observation	Score
If wrist is bent from midline: Add	+1

> After to choose the most suitable adjustment, it should be introduce it on the "Adjust wrist" - "Score" (Figure 10). In case of no necessary adjustment, the score should be considered zero "0".

ID	Activities	Score
1	Locate upper arm position	2
2	Adjust upper arm	1
3	Locate lower arm position	2
4	Adjust lower arm	1
5	Locate wrist position	3
6	Adjust wrist	1
7	Wrist twist	1
8	Add Arm Muscle use score A	1

0
1

- **Wrist twist**

> For wrist twist position, there are two scores: +1 and +2. These scores are related to the different twist movements of the wrist, performed in the job task chosen. The motivations of this scoring scale could be seen in the Table 4:

Table 4:

Observation	Score
If wrist is twisted in mid-range	+1
If wrist is at or near end of range	+2

> After to chose the most suitable score for this posture, it should be introduce it on the "Wrist twist" - "Score" (Figure 11):

ID	Activities	Score
1	Locate upper arm position	2
2	Adjust upper arm	1
3	Locate lower arm position	2
4	Adjust lower arm	1
5	Locate wrist position	3
6	Adjust wrist	1
7	Wrist twist	1
8	Add Arm Muscle use score A	1 1
9	Add Force/Load score A	2 2

- **Add arm muscle use on arms and wrist**

> For the usage of the arm muscle, there is a score of +1. This score is related to the different efforts of the arm and wrist muscle performed in the job task chosen and this scoring scale could be seen in the Table 5:

Table 5:

Observation	Score
If posture mainly static (i.e. held>1 minute)	+1
If action repeated occurs 4X per minute	+1

> After to choose the most suitable score for this adjustment, it should be introduce it on the "Add Arm Muscle use score A" - "Score" (Figure 11). In case of no necessary adjustment, the score should be considered zero "0".

ID	Activities	Score
1	Locate upper arm position	2
2	Adjust upper arm	1
3	Locate lower arm position	2
4	Adjust lower arm	1
5	Locate wrist position	3
6	Adjust wrist	1
7	Wrist twist	1
8	Add Arm Muscle use score A	1
9	Add Force/Load score A	2
10	Locate neck position	2

0
1

- **Add force/load on arms and wrist**

> For the force/load on arms and wrist, there is a range of scores between +0 and +3. These scores are related to the different ranges of weight of the force/load performed by arms and wrist in the job task chosen and this scoring scale could be seen in the Table 6:

Table 6:

Observation	Score
If load < 4.4 lbs. (intermittent)	+0
If load 4.4 to 22 lbs. (intermittent)	+1
If load 4.4 to 22 lbs. (static or repeated)	+2
If more than 22 lbs. or repeated or shocks	+3

> After to choose the most suitable score for this adjustment, it should be introduce it on the "Add Force/Load score A" - "Score" (Figure 13).

ID	Activities	Score
1	Locate upper arm position	2
2	Adjust upper arm	1
3	Locate lower arm position	2
4	Adjust lower arm	1
5	Locate wrist position	3
6	Adjust wrist	1
7	Wrist twist	1
8	Add Arm Muscle use score A	1
9	Add Force/Load score A	2
10	Locate neck position	2
11	Adjust neck	1
12	Locate trunk position	2
13	Adjust trunk	2

0
1
2
3

- **Locate neck position**

> For locate neck position, there is a range of scores between +1 and +4. These scores are related to the different positions of the neck performed in the job task chosen and this scoring scale could be seen in the figure 14:

> After to chose the most suitable score for this posture, it should be introduce it on the "Locate neck position" - "Score" (Figure 15):

ID	Activities	Score
1	Locate upper arm position	2
2	Adjust upper arm	1
3	Locate lower arm position	2
4	Adjust lower arm	1
5	Locate wrist position	3
6	Adjust wrist	1
7	Wrist twist	1
8	Add Arm Muscle use score A	1
9	Add Force/Load score A	2
10	Locate neck position	2
11	Adjust neck	1
12	Locate trunk position	2
13	Adjust trunk	2
14	Legs	1

1
2
3
4

- **Adjust neck**

> In order to evaluate other aspects of the posture, an adjustment of the previous score should be made with the criteria given in Table 7:

Table 7:

Observation	Score
If neck is twisted	+1
If neck is side bending	+1

> After to choose the most suitable adjustment, it should be introduce it on the "Adjust neck" - "Score" (Figure 16). In case of no necessary adjustment, the score should be

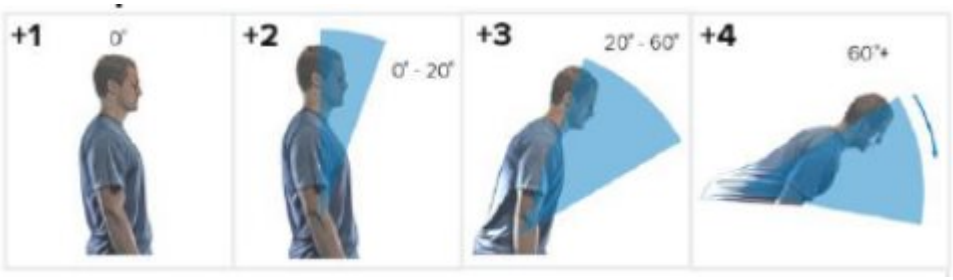
> considered zero "0".

ID	Activities	Score
1	Locate upper arm position	2
2	Adjust upper arm	1
3	Locate lower arm position	2
4	Adjust lower arm	1
5	Locate wrist position	3
6	Adjust wrist	1
7	Wrist twist	1
8	Add Arm Muscle use score A	1
9	Add Force/Load score A	2
10	Locate neck position	2
11	Adjust neck	1
12	Locate trunk position	2
13	Adjust trunk	2
14	Legs	1

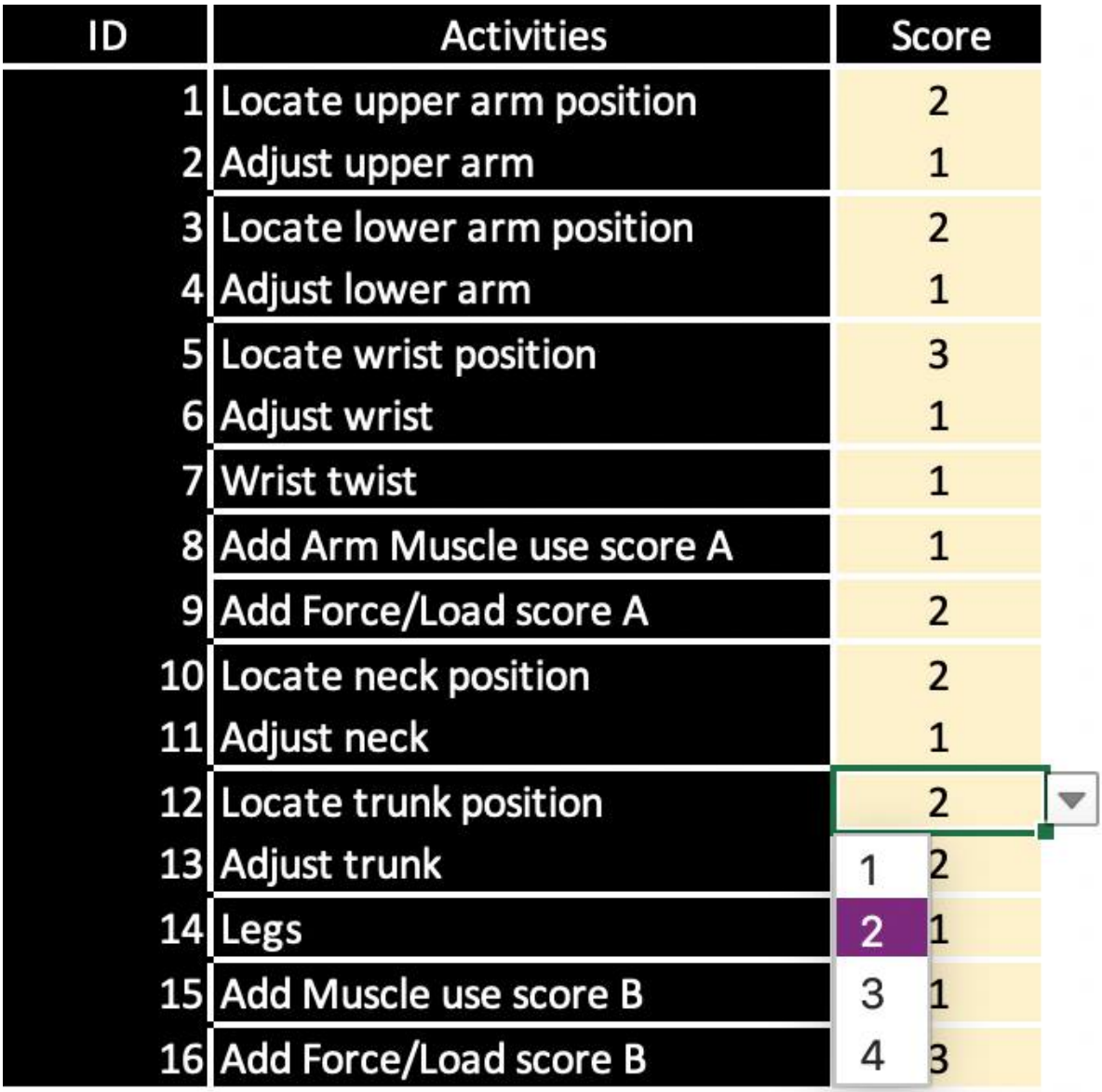
0
1
2

- **Locate trunk position**

> For locate trunk position, there is a range of scores between +1 and +4. These scores are related to the different positions of the trunk performed in the job task chosen and this scoring scale could be seen in the figure 17:



> After to chose the most suitable score for this posture, it should be introduce it on the "Locate trunk position" - "Score" (Figure 18):



- **Adjust trunk**

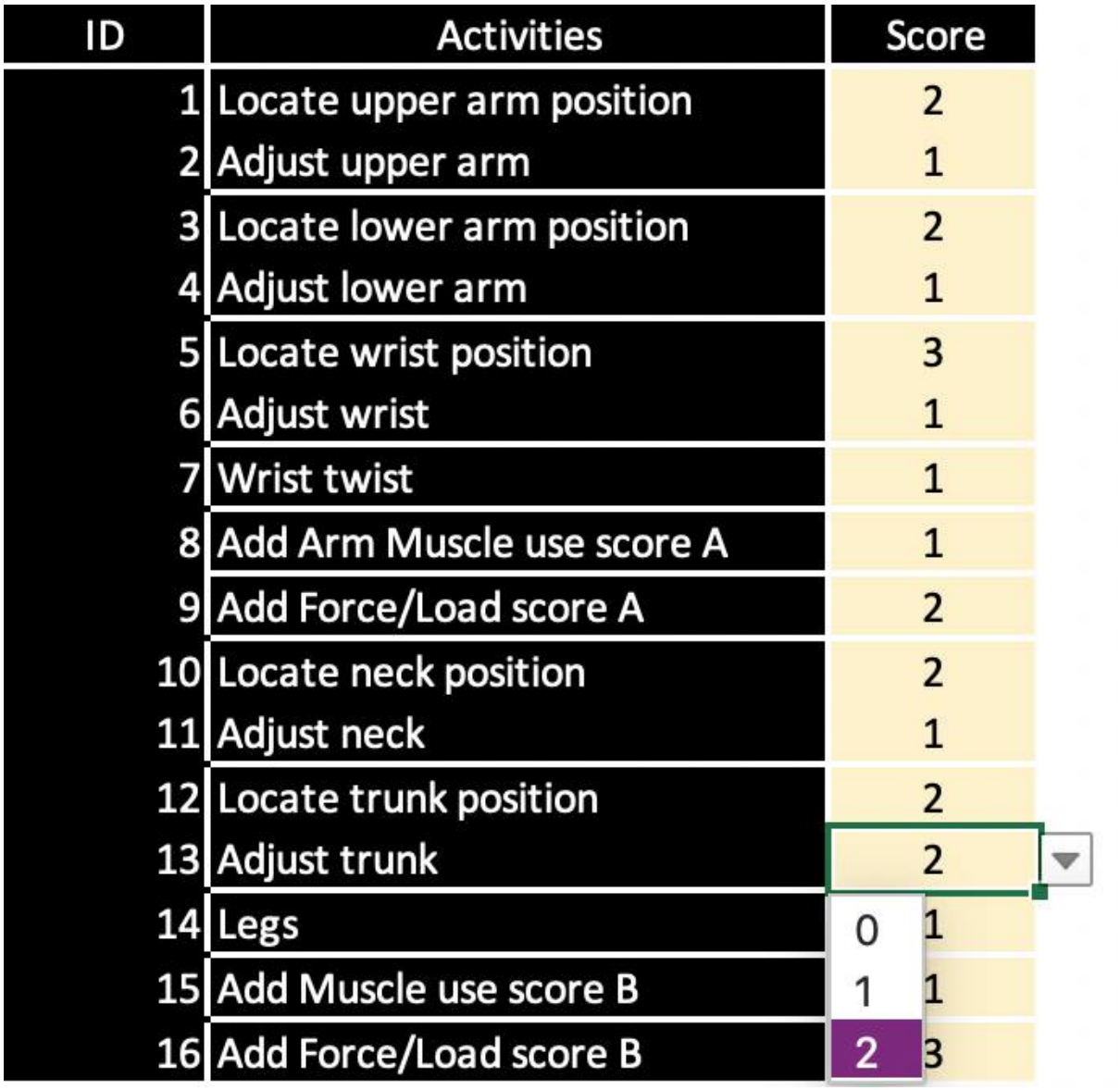
> In order to evaluate other aspects of the posture, an adjustment of the previous score should be made with the criteria given in Table 8:

Table 8:

Observation	Score
If trunk is twisted	+1

If trunk is side bending	+1

> After to choose the most suitable adjustment, it should be introduce it on the "Adjust trunk" - "Score" (Figure 19). In case of no necessary adjustment, the score should be considered zero "0".



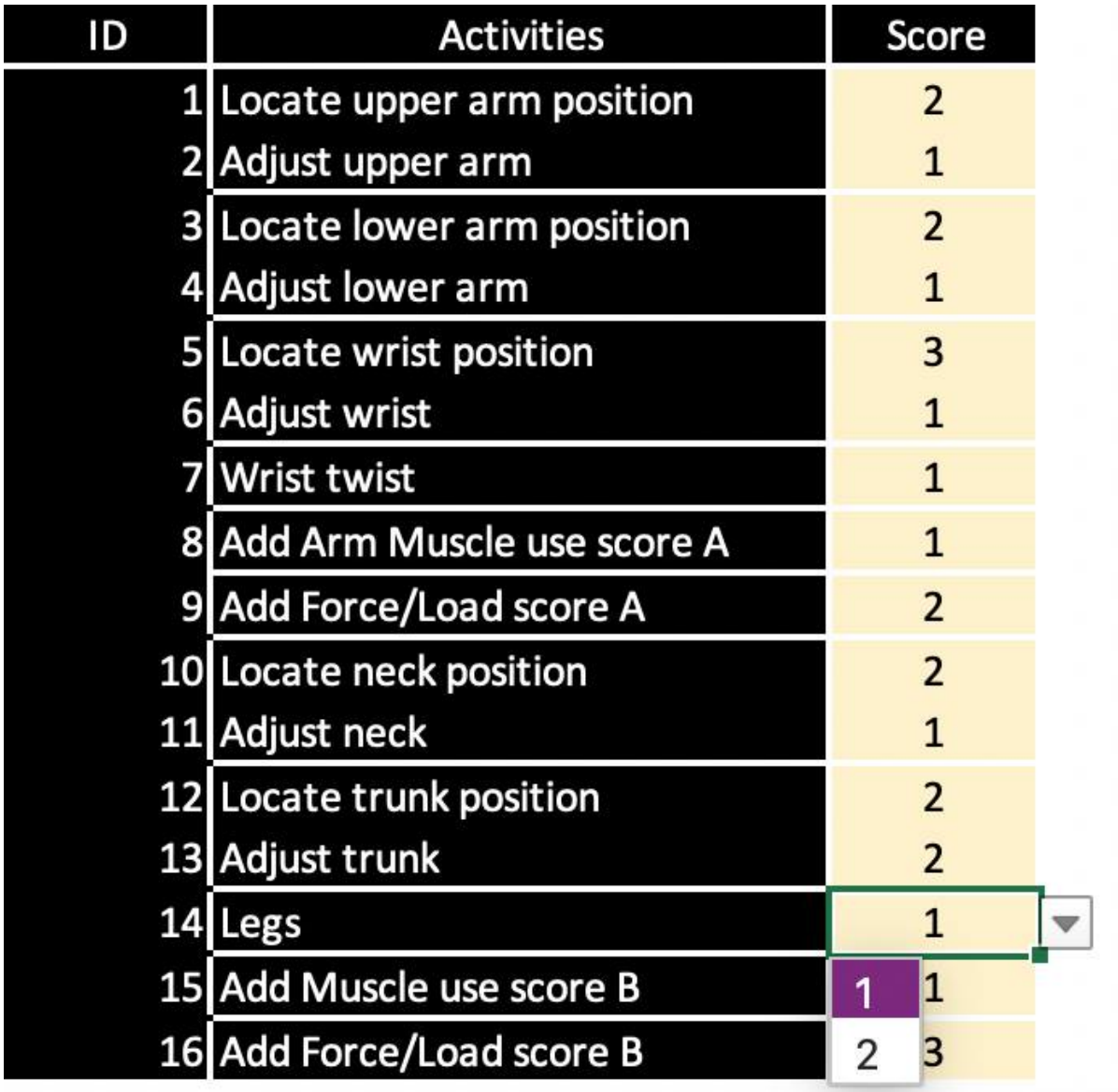
- **Legs**

> For locate legs position, there are two scores: +1 and +2. These scores are related to the different positions of the legs performed in the job task chosen and this scoring scale could be seen in the Table 9:

Table 9:

Observation	Score
If legs and feet are supported	+1
If legs and feet are not supported	+2

> After to chose the most suitable score for this posture, it should be introduce it on the "Legs" - "Score" (Figure 20):



ID	Activities	Score
1	Locate upper arm position	2
2	Adjust upper arm	1
3	Locate lower arm position	2
4	Adjust lower arm	1
5	Locate wrist position	3
6	Adjust wrist	1
7	Wrist twist	1
8	Add Arm Muscle use score A	1
9	Add Force/Load score A	2
10	Locate neck position	2
11	Adjust neck	1
12	Locate trunk position	2
13	Adjust trunk	2
14	Legs	1
15	Add Muscle use score B	1
16	Add Force/Load score B	3

- **Add muscle use on neck, trunk and legs**

> For the usage of the muscle on neck, trunk and legs, there is a score of +1. This score is related to the different efforts of the neck, trunk and legs muscle performed in the job task chosen and this scoring scale could be seen in the Table 10:

Table 10:

Observation	Score
If posture mainly static (i.e. held>1 minute)	+1
If action repeated occurs 4X per minute	+1

> After to choose the most suitable score for this adjustment, it should be introduce it on the "Add Muscle use score B" - "Score" (Figure 21). In case of no necessary adjustment, the score should be considered zero "0".

ID	Activities	Score
1	Locate upper arm position	2
2	Adjust upper arm	1
3	Locate lower arm position	2
4	Adjust lower arm	1
5	Locate wrist position	3
6	Adjust wrist	1
7	Wrist twist	1
8	Add Arm Muscle use score A	1
9	Add Force/Load score A	2
10	Locate neck position	2
11	Adjust neck	1
12	Locate trunk position	2
13	Adjust trunk	2
14	Legs	1
15	Add Muscle use score B	1
16	Add Force/Load score B	3

0
1

- **Add force/load on neck, trunk and legs**

> For the force/load on neck, trunk and legs, there is a range of scores between +0 and +3. These scores are related to the different ranges of weight of the force/load performed by neck, trunk and legs in the job task chosen and this scoring scale could be seen in the Table 11:

Table 11:

Observation	Score
If load < 4.4 lbs. (intermittent)	+0
If load 4.4 to 22 lbs. (intermittent)	+1

If load 4.4 to 22 lbs. (static or repeated)	+2
If more than 22 lbs. or repeated or shocks	+3

> After to choose the most suitable score for this adjustment, it should be introduce it on the "Add Force/Load score B" - "Score" (Figure 22). In case of no necessary adjustment, the score should be considered zero "0".

ID	Activities	Score
1	Locate upper arm position	2
2	Adjust upper arm	1
3	Locate lower arm position	2
4	Adjust lower arm	1
5	Locate wrist position	3
6	Adjust wrist	1
7	Wrist twist	1
8	Add Arm Muscle use score A	1
9	Add Force/Load score A	2
10	Locate neck position	2
11	Adjust neck	1
12	Locate trunk position	2
13	Adjust trunk	2
14	Legs	1
15	Add Muscle use score B	1
16	Add Force/Load score B	3

0
1
2
3

For a matter of information, it will be presented the three following sheets in the RULA Input File: "TableA" (Figure 23), "TableB" (Figure 24) and "TableC" (Figure 25). They are part of the theoretical basis of the RULA methodology and represent the different combinations of the scores chosen in the previous sheet "Input File" classified by:

- TableA or Section A: Arms (Upper and Lower), Wrist and Wrist Twist
- TableB or Section B: Neck, Trunk and Legs
- TableC or Section C: Final result from combination of both, final score TableA and final score TableB.

Table A		Wrist Score							
		1		2		3		4	
Upper Arm	Lower Arm	Wrist Twist		Wrist Twist		Wrist Twist		Wrist Twist	
		1	2	1	2	1	2	1	2
1	1	1	2	2	2	2	3	3	3
	2	2	2	2	2	3	3	3	3
	3	2	3	3	3	3	3	4	4
2	1	2	3	3	3	3	4	4	4
	2	3	3	3	3	3	4	4	4
	3	3	4	4	4	4	4	5	5
3	1	3	3	4	4	4	4	5	5
	2	3	4	4	4	4	4	5	5
	3	4	4	4	4	4	5	5	5
4	1	4	4	4	4	4	5	5	5
	2	4	4	4	4	4	5	5	5
	3	4	4	4	5	5	5	6	6
5	1	5	5	5	5	5	6	6	7
	2	5	6	6	6	6	7	7	7
	3	6	6	6	7	7	7	7	8
6	1	7	7	7	7	7	8	8	9
	2	8	8	8	8	8	9	9	9
	3	9	9	9	9	9	9	9	9

Neck Posture Score	Table B: Trunk Posture Score											
	1		2		3		4		5		6	
	Legs		Legs		Legs		Legs		Legs		Legs	
	1	2	1	2	1	2	1	2	1	2	1	2
1	1	3	2	3	3	4	5	5	6	6	7	7
2	2	3	2	3	4	5	5	5	6	7	7	7
3	3	3	3	4	4	5	5	6	6	7	7	7
4	5	5	5	6	6	7	7	7	7	7	8	8
5	7	7	7	7	7	8	8	8	8	8	8	8
6	8	8	8	8	8	8	8	9	9	9	9	9

Table C		Neck, Trunk, Leg Score						
		1	2	3	4	5	6	7+
Wrist / Arm Score	1	1	2	3	3	4	5	5
	2	2	2	3	4	4	5	5
	3	3	3	3	4	4	5	6
	4	3	3	3	4	5	6	6
	5	4	4	4	5	6	7	7
	6	4	4	5	6	6	7	7
	7	5	5	6	6	7	7	7
	8+	5	5	6	7	7	7	7

Remember the user must not complete any information in those tables, that's why after to finish the process of choose the different scores, the RULA Input File must be save it. Finally, this document will be imported into Python in order to continue with the programming part in the next steps.

Execution and results

Python Code

In order to run the model, it was used Anaconda "Jupyter Notebook" program with Python3 version.

Please, follow the next mandatory steps:

- Before to run the model, it is necessary to verify if the Input File document is saved in the same folder where the Python code is stored.
- The model must be run step by step in the same order in which the model was developed, installing first the available libraries on Python (NumPy, Pandas).
- The next steps in the Python Code import the Tables (A, B and C) from the RULA Input File, in order to use this information to find the final score per each section: Section A = Arms and Wrist, Section B = Neck, Trunk and Legs and the final solution: Section C = Total Assessment
- In the last part of the code, the final result per each section will be export to the RULA Input File.

Observations:

- The order in which the Input File was fulfilled, does not affect the result of the code.
- If after running the model, this generates a message of error on Python, please go back to the Input File sheet and check that the scores are within the scoring scale range corresponding to each position/adjustment.

Expected Outcomes

After exporting the final result of the Python Code into the RULA Input File, this will be stored in the last sheet of the document with the name "OutputsRULA".

The table with the results will be organised in the following way (Figure 26):

RESULT A	Level_of_risk	Final_Message
Arms and Wrist	4	Low risk, change may be needed
Neck, Trunk and Legs	3	Low risk, change may be needed
Total Assessment	5	Medium risk, further investigation, change soon

The range of Level of MSD Risk from the RULA methodology, is given by the following table (Figure 27):

Score	Level of MSD Risk
1-2	negligible risk, no action required
3-4	low risk, change may be needed
5-6	medium risk, further investigation, change soon
6+	very high risk, implement change now

3.8 OCRA

Introduction: the OCRA method

The OCRA method is a widely adopted approach for analysing workers' exposure to tasks featuring various upper limb (UL) risk factors like *repetitiveness, force, awkward postures and movements, lack of recovery periods, and others* . This methodology has been approved as best practise by the International Ergonomics Association (IEA) to monitor and predict the risk of upper limbs WMSDs, the most frequently reported cause of injury among european workers (source) . Nowadays, OCRA is adopted by over 30'000 technical specialists in Europe , like OSH operators, ergonomists and production engineers, guiding them to re-design organisational and physical workspace improving both productivity and operators' health in the long run.

Application

Despite its sophistication, this methodology is very time-consuming demanding several days to train operators in recognise and collect data regarding different MSDs risk dimensions: for this reason this procedure can be adopted only by experts who want to properly re-design the operators tasks. Furthermore, as the user will see in the following chapters, this methodology gives high relevance to the relationship between the duration of a task and the risk associated to it, deeply penalising manual processes in which the Cycle Time is equally distributed among tasks i.e. mounting/dismounting operations involving a limited set of screwdrivers.

For this reason, the OCRA method has been applied in a wide portfolio of industrial cases in the manufacturing and service sector, where jobs involving repetitive movements and/or efforts of the upper limbs were heterogeneous and complex. Main examples: Manufacture of mechanical components, electrical appliances, automobiles but also cloth textile and food processing.

This guide will support future users to understand the theoretical concepts concerning the OCRA methodology and will clarify the main steps for their adoption in the digital toolkit.

How to start

Main concepts and definitions

- *Job or Organised Work*: it represents the set of tasks (even repeated) made by an operator during his/her shift (even considering multiple cycles of execution).
- *Task*: A specific work activity aimed at obtaining a specific result (e.g., clamping/unclamping a piece, loading/unloading a pallet etc.) that can be repeated after each cycle.
- *Cycle*: A sequence of tasks performed by the upper limbs, which is repeated several times in the same way; in a practical contest, a cycle is considered as the set of tasks that an operator must accomplish to close a working loop, after which he/she will start again.
- *Cycle Time*: The total time assigned to carry out the sequence of tasks that characterises the cycle
- *Technical Action*: An elementary movement involving the upper limbs that represents the simplest operation to be evaluated by the OCRA method; a set of technical actions make a task.
- *Frequency*: Number of technical actions performed per unit of time [min]
- *Force*: Physical effort required by the worker to perform a technical action.
- *Awkward Postures and Movements*: Non-neutral postures and movements of the main joints of the upper limbs adopted to complete a sequence of technical actions characterising a cycle, whom impact in terms of MSDs is functional to the time exposure.
- *Stereotypy or Repetitiveness* :The repetition of the same gesture or series of gestures for most of the work period or shift.
- *Recovery Period*: The time interval within a shift during which the upper limbs are substantially inactive (i.e., the limbs are not performing any technical actions).
- *Additional Factor of Risk*: factor that takes into account the presence of additional risk of both physio-mechanical and organisational nature at the task level.

OCRA Index main formulas

OCRA Index	Risk Level	Corrective Actions
O.I. ≤ 1,5	Optimal	Not Required
1,6 < O.I. ≤ 2,2	Acceptable	Not Required
2,3 < O.I. ≤ 3,5	Very Low	Anamnestic health check
3,6 < O.I. ≤ 4,5	Low	Correct tasks organization
4,6 < O.I. ≤ 9	Medium	Correct tasks & workplace
O.I. > 9	Intense	Re-design tasks & workplace

The OCRA Index is obtained by the ratio between the number of actual technical actions (ATA) currently performed in a work shift and the corresponding number of recommended technical actions (RTA):

$OCRA_Index = ATA/RTA$

The number of actual technical actions performed in a shift: ATA**

The definition of the ATA for a single task is a simple procedure divided into two steps:

1. The computation of the Net Duration of Repetitive tasks (D) in a shift [min/shift]
2. The computation of the Average Frequency of Action per minute (F) for the task [act./min]
    - Total Net Duration of Repetitive tasks: D-

The first step in the OCRA methodology always looks at the organisational background of the company: which kind of shifts are distributed over the week? How many shifts per day are there? Which is the break schedule? How significant is the amount of time spent in non-conventional activities like cleaning, maintenance or financial inspections? To derive D, the following formula is applied:

D = Shift Duration - Total break time - Total non repetitive work time - Work time valued as recovery

Where all figures are derived by either organisational structure of the company's shifts or by direct observation of workers' behaviour. Since the literature suggests that the D parameter

should be computed for each task considered in the cycle, repeating the same procedure anytime we want to add another job to the set of operations performed in a shift, the amount of required data regarding the scheduling efficiency of a company becomes very high and subjected to uncertainty. (e.g. break times are not always respected precisely, non-repetitive activities can impact on the available time of different sets of tasks, anytime they happen).

For this reason, we assumed for our toolkit a simplification that, despite changing the computation approach for the D parameter of the methodology, doesn't affect significantly the outcome of the analysis, while it reduces significantly the time spent in the data collection phase:

Assumption: The Net duration of a single repetitive task is given by the product between D and the %duration of a task in a working cycle, relative to the Cycle Time:

Dtask = D · taskdur/CT

- Average Frequency of Action per minute/shift for a task: F-

It requires as input the set of tasks performed by the operator within a working cycle, the decomposition of each task in a set of elementary movements (least aggregated data analysed in the methodology), the shift length in minutes, the #pieces manufactured in a shift and the Cycle Time in [sec]. Then, **F** is computed as:

FF = (#actions\task · #cycles\shift) · 60/CT

Assumption: the average number of pieces produced in a shift corresponds to the number of cycles in which all the tasks are performed within a shift. To derive this value, if not already available, is sufficient to estimate the total time (human and machine) required to manufactured a finished product and divide the shift duration for that value:

#pieces manufactured = Shift Duration / Total time to manufactured a finished product

Finally, the **ATA** is obtained through the following formula, for each task in the cycle:

$ATAtask = F · Dtask$

The ATA of the entire cycle is given from the sum of all the ATA_tasks in one cycle.

1.2 The number of recommended technical actions performed in a shift: RTA

The computation of the Recommended Number of Technical actions is a more complex procedure involving the computation of the following parameters: DuM, RcM, CF, FoM, PoM, ReM and AdM.

```
-Duration Multiplier: DuM-
```

Once D has been computed for the entire cycle of tasks, the table below is addressed to directly computing the value of the Duration Multiplier:

Elements for Determining the Overall Duration Multiplier (in Minutes) for Repetitive Tasks per Shift (DuM)

Minutes Performing Repetitive Tasks/Shift	≤120	121–180	181–240	241–300	301–360	361–420	421–479	480–540	541–600	601–660	661–720	>720
Duration multiplier for total net duration of repetitive tasks/shift (DU_M)	2	1.7	1.5	1.3	1.2	1.1	1	0.83	0.66	0.5	0.35	0.25

```
-Recovery Multiplier: RcM-
```

The Recovery Multiplier is empirically estimated once observing the operator's working behaviour and deriving the average maximum time without recovery within a shift; a reference table can then associate this indicator to the RcM:

Elements for Determining Recovery Period Multipliers (Rc_M)

No. of Hours without Adequate Recovery	< 0.5 h	1 h	1.5 h	2 h	2.5 h	3 h	3.5 h	4 h	4.5 h	5 h	5.5 h	6 h	6.5 h	7 or More Hours
Corresponding Recovery Multiplier (RCM)	1	0.90	0.85	0.80	0.75	0.70	0.65	0.60	0.52	0.45	0.3	0.25	0.17	0.10

```
-CF parameter-
```

The action frequency constant is generally set equal to 30 actions/min as a reference value, but to guarantee a good estimated of the RTA, it should be defined roughly considering the ratio between the number of technical action in one cycle and the cycle time.

```
-Force Multiplier : FoM-
```

The computation of the force multiplier requires a considerable degree of accuracy in the data collection phase, since values for the duration, the force load and the body area (right, left or total) involved in the technical action must be properly extrapolated either through operator movements' simulation, general assumptions or direct observation (latter case may not be possible due to privacy issues).

The FoM computation is divided into three steps:

1. *Task division into technical actions (T.A.):* to ensure the highest degree of accuracy possible, a process supervisor (e.g. an ergonomist, a qualified operator) must decompose all the tasks of a cycle into elementary movements for whom posture and force factors can be estimated without bias.

2. *Force Score (F.S.) computation*: The force score is obtained by computing the *force load*, expressed through Borg CR-10 Scale (table below), that accounts for the weight of the object carried out in a task. If multiple T.A. are included in a single task, then the *Weighted Force Score* is computed by weighting the force load of each T.A. for the *%time spent* in the force exertion relative to the Cycle Time:

Borg CR-10 Scale			
0.5	Extremely light		
1	Very light	6	
2	Light	7	Very hard
3	Moderate	8	
4		9	
5	Hard	10	Extremely hard (almost maximum)

3. *FoM computation*: once the Weighted F.S. has been defined for a task, a multiplier table can be directly consulted for the extrapolation of the FoM:

Force level as % MVC/F_L	**5%**	**10%**	**15%**	**20%**	**25%**	**30%**	**35%**	**40%**	**45%**	**≥50%**
Borg CR-10 scale scores	0.5	1	1.5	2	2.5	3	3.5	4	4.5	≥5
Force multiplier (FoM)	1	0.85	0.75	0.65	0.55	0.45	0.35	0.2	0.1	0.01

```
-Posture Multiplier: PoM-
```

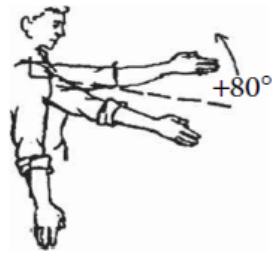
The estimation of the risk associated to operators'exposure to awkward postures is the most time-consuming step of the OCRA methodology, due to the high accuracy required and the multiple body areas considered in the analysis. Like the FoM, also the *posture multiplier* computation is task-specific and considers a weighted value if multiple T.A. are investigated for a single task.

Assumption: For the sake of consistency, in our model we rely on the same tasks' duration considered in the previous step, assuming that there is no distinction between the T.A. execution time and the time spent in an awkward posture (regardless the body area involved) for that T.A.;

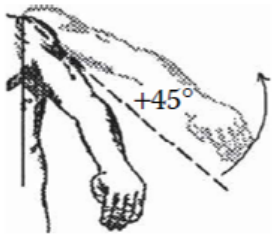
According to the method, 4 body areas for the Upper Limbs are assessed in terms of MSD risk: Shoulders, Elbows, Wrists and Hands. As shown in the tables below, for each area of the operator's upper limbs, a set of reference harmful movements and postures is defined; a posture score is then set by matching the type of harmful movement/posture and the %time spent in that posture relative to CT:

	5%	1%	15%	20%	25%	30%	35%	40%	45%	51%	55%	60%	65%	70%	75%	81%	>84%
ABD 45°	0.25	0.5	1.3	2.4	4	4	4.4	5.2	6.5	8	8	8.3	8.9	9.7	10.7	12	12
EXT°	0.25	0.5	1.3	2.4	4	4	4.4	5.2	6.5	8	8	8.3	8.9	9.7	10.7	12	12
FL-AB 80°	2	4	6	8	10	12	14	16	19	24	24	24.6	25.5	26.8	28	28	28

Arm flexion



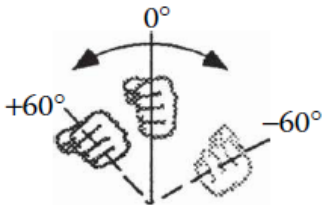
Arm abduction



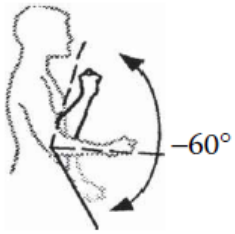
Arm extension

0	5%	1%	15%	20%	25%	30%	35%	40%	45%	51%	55%	60%	65%	70%	75%	81%	>84%
FLX/EXT	0	0.3	0.7	1.3	2	2	2.2	2.5	3.1	4	4	4.1	4.3	4.7	5.3	6	6
PRON	0	0.3	0.7	1.3	2	2	2.2	2.5	3.1	4	4	4.1	4.3	4.7	5.3	6	6
SUPIN	0.25	0.5	1.3	2.4	4	4	4.4	5.2	6.5	8	8	8.3	8.9	9.7	10.7	12	12

Elbow pronation-supination

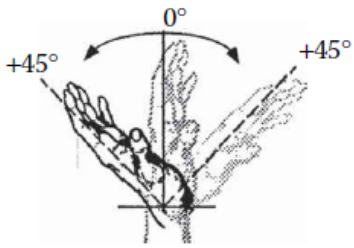


Elbow flexion-extension

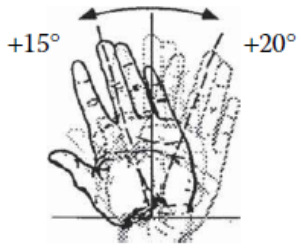


0	5%	1%	15%	20%	25%	30%	35%	40%	45%	51%	55%	60%	65%	70%	75%	81%	>84%
Radioulnar	0	0.3	0.7	1.3	2	2	2.2	2.5	3.1	4	4	4.1	4.3	4.7	5.3	6	6
FLEX	0.2	0.4	1	1.8	3	3	3.3	3.8	4.7	6	6	6.2	6.6	7.2	8	9	9
EXT	0.25	0.5	1.3	2.4	4	4	4.4	5.2	6.5	8	8	8.3	8.9	9.7	10.7	12	12

extension-flexion



radioulnar deviation



0	5%	1%	15%	20%	25%	30%	35%	40%	45%	51%	55%	60%	65%	70%	75%	81%	>84%
Wide grip	0	0.15	0.35	0.6	1	1	1.05	1.2	1.5	2	2	2.05	2.2	2.4	2.7	3	3
Narrow grip	0	0.3	0.7	1.3	2	2	2.2	2.5	3.1	4	4	4.1	4.3	4.7	5.3	6	6
Pinch	0.2	0.4	1	1.8	3	3	3.3	3.8	4.7	6	6	6.2	6.6	7.2	8	9	9
Palmar/hook	0.25	0.5	1.3	2.4	4	4	4.4	5.2	6.5	8	8	8.3	8.9	9.7	10.7	12	12

Wide grip (3–4 cm) Narrow grip Pinch Palmar and hook grip

Assumption: considering the same body area, at most one dangerous movement can be identified in a T.A. performed by the operator; however, a T.A. can be harmful for multiple body areas. To sum up, the number of awkward postures associated to a T.A. can go from 0 to 4 at most.

The computation of the PoM follows two steps:

1. *Posture Scores (P.S.) computation*: respecting the task sub-division defined in the previous analysis , each elementary movement is assessed by either simulation or direct observation of the operator's behaviour to identify the presence of possible awkward postures. If multiple T.A. belonging to the same task have the same awkward posture, the cumulative %time respect to the CT, spent in an awkward posture must be considered in the score table. In this way, posture scores are properly computed and grouped at the task level.

2. *Total posture score (Total P.S.) computation*: Once computing the P.S. for each awkward posture made in a task, the Total P.S. is obtained by summing together the partial scores belonging to the same Posture Area (shoulder, elbow, wrist or hand).

3. *PoM computation*: for each task, the highest among the Total P.S. is picked considering the four body areas; then, this value is used in the posture risk table to derive the correspondent Posture Multiplier:

Elements for Determining Basic and Intermediate Postural Effort Multipliers (Po_M)

Score (select the highest from among the shoulder, elbow, wrist, hand)	0–3	4–7	8–11	12–15	16–19	20–23	24–27	≥28
Awkward posture and movement multiplier (Po_M)	1	0.70	0.60	0.50	0.33	0.1	0.07	0.03

0	4	5.4	6.7	8	9.4	10.7	12	13.4	14.7	16	17.4	18.7	20	21.4	22.7	24	25.4	26.7	28
1	0.7	0.6667	0.6333	0.6	0.5667	0.5333	0.5	0.4433	0.3867	0.33	0.2533	0.1767	0.1	0.09	0.08	0.07	0.0567	0.0433	0.03

```
-Repetitiveness as a risk factor: ReM-
```

The literature review suggests that repetitiveness of movements or postural efforts can become a relevant driver of MSD when at least one out of three of these conditions is satisfied:

a. Identical technical actions or groups of identical technical actions are repeated for almost the entire cycle time (0-50% absent risk; 50%-80% moderate risk; >80% high risk)

b. Static postures are sustained along a great portion of the cycle time (0-50%: absent risk; 50%-80%: moderate risk; >80%: high risk)

c. Operators execute multiple times extremely short cycles of activities featuring actions that involve the upper limbs. (CT>15 sec: low/absent risk; 8<CT<15 sec: medium risk; CT<8 sec: high risk)

Assumption: Being an indicator that is representative for the entire shift evaluation, a single value of the ReM is computed for the entire set of tasks referring to the same data collected about the awkward movements in the PoM computation phase.

Finally, the worst case among the three conditions is taken as a reference to get the correspondent ReM value from the table provided below:

Elements for Determining the Stereotypy Multiplier (Re_M)

	Stereotypy Multiplier (RE_M): High Stereotypy
0.7	Identical technical actions or groups of identical technical actions repeated for almost the entire cycle (more than 80%). Static postures sustained for over 80% of the cycle time (e.g., pro-longed gripping of a knife or screwdriver). Extremely short cycles lasting less than 8 s, featuring actions that involve the upper limbs.
	Stereotypy Multiplier (RE_M): Moderate Stereotypy
0.85	Identical technical actions or groups of identical technical actions repeated for over 50% of the cycle time. Static postures sustained for over 50% of the cycle time (e.g., prolonged gripping of a knife or screwdriver). Extremely short cycles lasting less than 15 s, featuring actions that involve the upper limbs.

```
-Additional Risks Assessement: AdM-
```

The Additional Risk Multiplier embodies the incidence on the OCRA Index represented by two components:

- *Presence of physio-mechanical dangerous conditions (PMC) in the working environment*: like the presence of vibrating tools or the operator's exposure to harmful temperatures while performing a task. 8 factors are considered in the methodology and the presence of each of them, with the relative %time exposure to this risk, is assessed by observing the operator task's execution.
- *Organisational risk conditions (ORC) linked to operators working rhythm*: it's not strange to believe that the operator's pace may be affected by the work pace of the machine in some processes. An ORC score of 0 is set if the operator is working independently from the machine; a score of 8 if the operator works together with the machine but flexibility in delays is allowed by buffers; a score of 12 is given if the operator works synchronously with the machine. It's obvious to notice that ORC score is a unique value referred to the entire shift.

The AdM is computed in two steps:

1. *PMC Score computation for each task*: relying on the table shown below, the PMC score is computed as a function of %duration of a task relative to CT and of the type of PMC:

Code	Physio-mechanical factors
0	No physio-mechanical risk
1	Operating vibrating tool
2	High precision Work
3	Localized compression of hand/forearm by tool use
4	Exposure to very low ambient temperature
5	Inadequate Gloves
6	Slippery part surfaces
7	Execution of jerking/forceful movements
8	Execution of actions with contact stress

1. *AdM computation*: the AdM is obtained for each task by firstly adding the ORC score to the PMC score of each task and then associating the Total Risk Score to the correspondent AdM by referring to the additional risk table, shown below;

Elements for Determining the Additional Risk Factor Multiplier (Ad_M)

Scores for additional risk factors	0–3	4–7	8–11	12–15	≥16
Additional multiplier (AD_M)	1	0.95	0.90	0.85	0.80

Finally, the **RTA** is obtained by the following formula, for each task:

$RTAtask = CF · D · DoM · RcM · FoM · PoM · AdM · ReM$

The *overall value of the RTA* for the entire cycle is derived by summing all the RTAtask in the cycle.

It's crucial to remember that, since both the RTA and ATA computation is performed at the task level, both indicators are influenced by the "Body Area" that the operator uses to accomplish each task: hence a value of the OCRAright and OCRAleft index must be computed in a precise and coherent way. At this purpose we have formalised the following assumptions:

Assunption 1: An operator can potentially perform each technical action (T.A.) either with the *Left or Right* part of his/her upper limbs. However, if the T.A. involves the lifting or handling of a heavy object (Hypothesis: *weight of the object over 5kg*), then he/she must use *both upper limbs*.

Assumtpion 2: Once an operator has decided to make a T.A. with either the Right/Left/Both areas of his/her upper limbs, he/she must perform all the remaining T.A. needed to make the correspondent task with the same part of his/her body. E.g.: exchanging of objects from Right to Left hand within a single task are avoided because can be time-consuming.

Prepare the input data

The Excel Spreadsheet Structure

In this section we will discuss the format of the *OCRAinput* excel spreadsheet, highlighting the main characteristics of each table, the constraints on the data that the operator should load and how to load the outputs by running the python code.

Organisational data: the Shiftdata Sheet

Inside the *Shiftdata* sheet, data involving the organisational parameters are loaded by the operator who generally has good knowledge of his/her work schedule and his/her daily working behaviour in terms of unofficial and scheduled breaks. All data regarding time measures are expressed in minutes and the excel file will automatically compute in the same sheet:

- *The Total net repetitive work time in the shift (D)*: derived by subtracting to the shift duration the values of the three rows below it (look at the section on *D* of the chapter *1.1:"The number of actual technical actions performed in a shift: ATA"* for the formula explanation).
- *The Duration Multiplier (DuM)*: computed through the following *if condition* implemented on excel: *$SE(B10<=120;2;SE(B10<=180;1,7;SE(B10<=240;1,5;SE(B10<=300;1,3;SE(B10<=360;B10;SE(B10<=420;1,1;SE(B10<=480;1;SE(B10<=540;0,83;SE(B10<=600;0,66;SE(B10<=660;0,5;SE(B10<=720;0,35;0,25))))))))))$*.

This function replicates the DuM table provided in the section on "DuM" of chapter 1.1. **B10** is the cell reference for the figure "D". (Language in the excel solver: ITA)

- *Recovery Multiplier (RcM)*: computed through an *if condition* on excel that reflects the RcM table provided in the section "RcM" of chapter 1.1: *$SE(B6<=0,5;1;SE(B6<=1;0,9;SE(B6<=1,5;0,85;SE(B6<=2;0,8;SE(B6<=2,5;0,75;SE(B6<=3;0,7;SE(B6<=3,5;0,65;SE(B6<=4;0,6;SE(B6<=4,5;0,52;SE(B6<=5;0,45;SE(B6<=5,5;0,3;SE(B6<=6;0,25;SE(B6<=6,5;0,17;0,1)))))))))))))$* where **B6** is the cell reference for the figure "Number of hours without a recovery period" provided as input by the operator.
- *Organizational Risk Score (ORS)*: computed through an *if condition* that reflects the working rythm of the operator relative to the one of the machine (look at "ORC" section

in the chapter 1.2) : *$SE(B9=0;1;SE(B9=1;8;SE(B9=2;12;"Errore")))$*

where **B9** represents the ORC code considered in the toolkit and inserted by the operator:

- 0: Free work pace
- 1: Work pace set by machine but with buffers
- 2: Work pace set entirely by the machine

Force and Postural data: the FPRdata Sheet

The *FPRdata* sheet represents the core of the ergonomic risk assessment of each operator's task; as shown in the table above, the sheet gathers data regarding:

- The *Technical Actions* performed by the operator inside one cycle, with the correspondent code that identifies them in the process in order to avoid misunderstandings when a same T.A. is executed more times in different tasks;
- The correspondent *Task Code* that allows the python code to group the T.A. in one task during the ergonomic risk assessment;
- The *Duration of each Technical Action* involved in the cycle which is used to derive the *Cycle Time*, required for both the RTA and the ATA computation;
- The *Force Load* measured in each T.A. execution, expressed in values of the Borg CR-10 scale. Bear in mind that, due to the presence of ad hoc constraint on the data insertion, the values of the force load can be either null or the ones of the Borg scale.
- The *Posture Codes* for the 4 Upper Limb areas have been defined in this way:

Code	Shoulder Posture	Elbow Posture	Wrist Posture
0	No wrong shoulder posture	No wrong elbow posture	No wrong wrist posture
1	Arms in abdution >45°	Flexion-Extension >60°	Radiulnar deviation > 20°
2	Arms in extension >20°	Pronation> 60°	Flexion >45°
3	Arms in flexion-abduction >80°	Supination> 60°	Estension > 45°

Code	Hand Posture
0	No wrong hand posture
1	Wide Grip
2	Narrow Grip
3	Pinch
4	Palmar/Hook Grip

Constraints on the data insertion for all these codes are already built in excel in order to respect data consistency and to support rapid data processing in python.

Additional risk factors: the AdMdata sheet

The *AdMdata* sheet collects, at task level, information regarding the possible physio-mechanical (P.M.) risk conditions of the operator. Due to the simplicity of the industrial case on which this toolkit has been validated, (a set of mounting/dismounting operations not requiring the usage on any peculiar tool a part from screwdrivers) we have formulated the following weak assumption, that might be dropped in future improvements of the toolkit to consider more general industrial cases:

Assumption: Considering a single task, at most one among the PM code can be assigned by the operator i.e. at most one Physio-mechanical risk condition can happen. Following the logic of building a consistent machine-readable input file for our toolkit, we have considered 8 PM codes reflecting the possible P.M. risk conditions shown in the literatur and in the "AdM" chapter 2.3.

The hidden sheets: Shiftpy and PSTpy

Two additional sheets are worth of mention in the Excel input file for the OCRA methodology:

- *Shiftpy* sheet: Not seen by the user interface, this hidden sheet as the role of supporting the python toolkit to read easily the column names in the correspondent *Shiftdata* sheet, ensuring a one-to-one correspondence of values and avoiding errors in the reading phase.
- *PSTpy* sheet: This specific sheet has no meaning for the user but it's crucial to speed up the phase of Posture scores computation for all the T.A. in the cycle. It represents in a single table the set of scores for all the posture areas of the body, with an increasing value that is functional to the %time spend in the posture:

Range	SP1	SP2	SP3	EP1	EP2	EP3	WP1	WP2	WP3	HP1	HP2	HP3	HP4
5%	0.25	0.25	2	0	0	0.25	0	0.2	0.25	0	0	0.2	0.25
10%	0.5	0.5	4	0.3	0.3	0.5	0.3	0.4	0.5	0.15	0.3	0.4	0.5
15%	1.3	1.3	6	0.7	0.7	1.3	0.7	1	1.3	0.35	0.7	1	1.3
20%	2.4	2.4	8	1.3	1.3	2.4	1.3	1.8	2.4	0.6	1.3	1.8	2.4
25%	4	4	10	2	2	4	2	3	4	1	2	3	4
30%	4	4	12	2	2	4	2	3	4	1	2	3	4
35%	4.4	4.4	14	2.2	2.2	4.4	2.2	3.3	4.4	1.05	2.2	3.3	4.4
40%	5.2	5.2	16	2.5	2.5	5.2	2.5	3.8	5.2	1.2	2.5	3.8	5.2
45%	6.5	6.5	19	3.1	3.1	6.5	3.1	4.7	6.5	1.5	3.1	4.7	6.5
51%	8	8	24	4	4	8	4	6	8	2	4	6	8
55%	8	8	24	4	4	8	4	6	8	2	4	6	8
60%	8.3	8.3	24.6	4.1	4.1	8.3	4.1	6.2	8.3	2.05	4.1	6.2	8.3
65%	8.9	8.9	25.5	4.3	4.3	8.9	4.3	6.6	8.9	2.2	4.3	6.6	8.9
70%	9.7	9.7	26.8	4.7	4.7	9.7	4.7	7.2	9.7	2.4	4.7	7.2	9.7
75%	10.7	10.7	28	5.3	5.3	10.7	5.3	8	10.7	2.7	5.3	8	10.7
81%	12	12	28	6	6	12	6	9	12	3	6	9	12
84%	12	12	28	6	6	12	6	9	12	3	6	9	12

Execution and results

Output 1: OCRA partial scores

The first output provided by the OCRA digital tool is an overview of the partial scores for each task, used to highlight the incidence of each working task on the overall value of the OCRA index. As shown in the sample table below, in the *Output 1* sheet, which is created after one code run, the absolute and % values of the ATA and RTA of each task are computed; in addition, a partial risk feedback at the task level is provided:

	TC	ATA	%ATA	RTA	%RTA	Ocra_Index	Risk	Body_Area
0	1	912.23	5%	542.41	4%	1.68	Acceptable	R
1	2	114.03	1%	241.07	2%	0.47	Optimal	R
2	3	114.03	1%	241.07	2%	0.47	Optimal	R
3	4	152.04	1%	180.80	1%	0.84	Optimal	R
4	5	912.23	5%	542.41	4%	1.68	Acceptable	B
5	6	2850.71	16%	1356.01	10%	2.10	Acceptable	B
6	7	912.23	5%	542.41	4%	1.68	Acceptable	L
7	8	114.03	1%	241.07	2%	0.47	Optimal	R
8	9	114.03	1%	241.07	2%	0.47	Optimal	R
9	10	342.09	2%	271.20	2%	1.26	Optimal	R
10	11	114.03	1%	271.20	2%	0.42	Optimal	R
11	12	912.23	5%	542.41	4%	1.68	Acceptable	R
12	13	114.03	1%	241.07	2%	0.47	Optimal	R
13	14	114.03	1%	241.07	2%	0.47	Optimal	R
14	15	114.03	1%	241.07	2%	0.47	Optimal	R
15	16	152.04	1%	180.80	1%	0.84	Optimal	R

Output 2: Overall OCRA Index for the right and left body area

Lastly, the desired outcome of the OCRA methodology is provided in the *Output 2* sheet: the overall value for the light and left part of the upper limbs of the OCRA Index and the correspondent Risk Range:

	Ocra Index	Value	Risk
0	Ocra Left	1.7819	Acceptable
1	Ocra Right	1.2312	Optimal

Toolkit application modes

The OCRA toolkit has been tested in two implementation modes:

• Independent OCRA execution with all data loaded by the operator

• Semi-Automated OCRA execution with task processing time estimation through MOST suite integration on python.

The second mode allows an operator to derive the OCRA index in two steps:

1. Loading task data on the MOST input file transferring the Processing times and Technical actions list on the OCRA excel file
2. Loading shift & postural data, by copying and pasting values from the FPRpy file created ad hoc by the OCRA-MOST link python code, and run the OCRA toolkit to get the OCRA index and the Cycle Total Risk

In this way the data collection time is significantly reduced, providing a faster toolkit implementation given that the user has already been trained for handling the MOST toolkit too. It's crucial to specify that the MOSt-OCRA link is represented by the *FPRpy* sheet that is created ad hoc for this implementation mode once the OCRA-MOST data transfering code has been run. In particular, the columns of task code, task name, move type and task duration have all their values set as links to the *FPRpy* sheet.

Final Remarks

This guide provides key pillars for understanding the main theoretical concepts behind the OCRA methodology; furthermore it supports the user in the correct interpretation of the different OCRA sheets in excel, highlighting the main constraints and providing examples of the outputs of the toolkit.

The properly take advantage of this methodology the user should bear in mind that:

- This toolkit is based on assumptions that, despite being reasonable, don't perfectly reflect the theoretical concepts of the OCRA method
- The OCRA method works efficiently with tasks whom processing time are not equally distributed along a cycle

- To avoid uncontrollable errors in the python code execution, the user shouldn't change the names in any column of each excel sheet;
- To avoid wrong reading of the data in python, the user should check that all the columns that have not been filled with data, have been left with empty values (not null values). 200 rows have been set for the FPRdata sheet and the AdMdata sheet.