



ENERGY-AWARE FACTORY ANALYTICS FOR PROCESS INDUSTRIES

Deliverable D4.3

Systemic Cognitive Models Prototypes

Version
1.0

Lead Partner
TUC

Date
25/07/2022

Project Name
FACTLOG – Energy-aware Factory Analytics for Process Industries

Call Identifier H2020-NMBP-SPIRE-2019	Topic DT-SPIRE-06-2019 - Digital technologies for improved performance in cognitive production plants
Project Reference 869951	Start date November 1 st , 2019
Type of Action IA – Innovation Action	Duration 42 Months

Dissemination Level

X	PU	Public
	CO	Confidential, restricted under conditions set out in the Grant Agreement.
	CI	Classified information as referred in the Commission Decision 2001/844/EC

Disclaimer

This document reflects the opinion of the authors only.

While the information contained herein is believed to be accurate, neither the FACTLOG consortium as a whole, nor any of its members, their officers, employees or agents make no warranty that this material is capable of use, or that use of the information is free from risk and accept no liability for loss or damage suffered by any person in respect of any inaccuracy or omission.

This document contains information, which is the copyright of FACTLOG consortium, and may not be copied, reproduced, stored in a retrieval system or transmitted, in any form or by any means, in whole or in part, without written permission. The commercial use of any information contained in this document may require a license from the proprietor of that information. The document must be referenced if used in a publication.

Executive Summary

In the context of FACTLOG, a Process Modelling and Simulation methodology is developed. The implemented models are parametric since the data used for these models are of three main categories. Static data that refer to the structural characteristics of the industrial systems of the pilot, dynamic data from the pilots that refer mainly to products being processed in their systems and in the appearance of events that change system states (such as machine breakdowns) and dynamic data received from the associate services. In particular operation of associate services (mainly optimisation, analytics and knowledge graphs) provides useful data that are used to define the scenarios under study (in the case of schedules) or make the simulated scenarios much more realistic (as in the case of analytics that can detect patterns regarding the appearance of certain events in the system on the simulated time horizon). The described data from the associate services justify the cognitive characteristics of the models developed.

After the definition and description of the proposed methodology, the model prototypes of certain realistic, concerning types and quantity of data, process models of the pilot systems are developed. The implemented models are used to simulate alternative scenarios, KPIs are calculated, and the results obtained are cross-validated with the results received from the optimisation service. From the values of the calculated KPIs, useful conclusions regarding the behaviour and the efficiency of the modelled industrial system are extracted. These KPIs are of general use, but also pilot-specific KPIs can be calculated with respect to the individual needs and available data.

The system cognitive process models developed in the context of FACTLOG have the following attributes:

- They are fully parametric. This makes them easy to update, expandable and useful for studying a wide variety of scenarios regarding the complexity and the behaviour of the system under completely different situations.
- They are dynamic in the sense that key flows, as well as key process control settings, are continuously updated in near real-time, in connection to a real-time monitoring system maintaining a digital Shadow of the physical System.
- They are interconnected through APIs with the associate services. This makes them much more realistic as the data used can describe realistic behaviours of increased complexity and from many different points of view.

The Process Simulation and Modelling Tool is developed in the context of T4.1 and T4.3, to address the process modelling and simulation requirements of the FACTLOG project. PSM Tool allows the user not only to create a process industry model but also to simulate the operation of such an industrial system. An Application Programming Interface has also been created to assist the integration of the Process Simulation Modelling module into the Digital Twins Platform of FACTLOG.

Revision History

Revision	Date	Description	Organisation
0.1	09/05/2022	Document Creation, Contents and Sections	TUC
0.2	29/06/2022	Sections changed, information added	TUC
0.3	01/07/2022	New Pilot information integration	TUC
0.4	04/07/2022	Chapter 2,3,4 Initial version	TUC
0.5	05/07/2022	Internal discussion regarding D4.3	TUC
0.6	06/07/2022	Corrections and Adjustments	TUC
0.7	07/07/2022	Chapter 2,3,4 Final version	TUC
0.8	08/07/2022	Addition of Appendices, List of Tables, List of Figures	TUC
0.9	12/07/2022	Peer review	QLECTOR, HANSE
1.0	25/07/2022	Final version after reviewing from QLECTOR and HANSE	TUC

Contributors

Organisation	Author	E-Mail
TUC	George Arampatzis	garampatzis@tuc.gr
TUC	George Tsinarakis	gtsinarakis@tuc.gr
TUC	Nikos Sarantinoudis	nsarantinoudis@tuc.gr

Table of Contents

Executive Summary3

Revision History4

1 Introduction.....9

 1.1 Purpose and Scope9

 1.2 Relation with other Deliverables9

 1.3 Structure of the Document.....10

2 Methodological Framework11

3 Dynamic Modelling and Simulation15

 3.1 Discrete Process Simulation and Modelling Implementation15

 3.1.1 Petri Net Models.....15

 3.1.2 Model Implementation in Python16

 3.2 Continuous Process Simulation and Modelling Implementation18

 3.2.1 PSM Tool.....18

 3.2.2 PSM Standalone Tool20

 3.3 PSM Online Service (HTTP API)37

4 Implementation per Pilot Case42

 4.1 BRC.....42

 4.1.1 Case Description42

 4.1.2 Process Modelling and Simulation43

 4.1.3 Results and Outcomes45

 4.2 CONTINENTAL47

 4.2.1 Case Description47

 4.2.2 Process Modelling and Simulation49

 4.2.3 Results50

 4.3 TUPRAS.....52

 4.3.1 Case Description52

4.3.2	Process Modelling and Simulation	53
4.3.3	Results	58
References	61
Appendix I – Continuous Process Simulation and Modelling API	63
Appendix II – BRC Simulation Request	68
Appendix III – CONTINENTAL Simulation Request.....	71
Appendix IV – BRC Simulation Result	83
Appendix V – CONTINENTAL Simulation Result	87

List of Figures

Figure 1: Process modelling and simulation methodology main stages.....	11
Figure 2: Modelling and Simulation methodology and associated services interactions	14
Figure 3: Discrete Simulation result .json structure.....	18
Figure 4: Continuous pilot case structural components	20
Figure 5: PSM Tool Interface and Splash Screen.....	20
Figure 6: PSM main user interface areas.....	21
Figure 7: Component Toolbar in PSM Tool interface.....	22
Figure 8: Pointer	22
Figure 9: Process.....	22
Figure 10: Input.....	22
Figure 11: Output.....	23
Figure 12: Junction	23
Figure 13: Process.....	23
Figure 14: Stage	23
Figure 15: Model Editor / Diagram Area in PSM Tool interface	24
Figure 16: Model Explorer / Resources Manager in PSM Tool interface	25
Figure 17: Network Tab	26
Figure 18: Stages Tab	27
Figure 19: Resources Tab	27
Figure 20: Overview Tab.....	28
Figure 21: Properties Editor in PSM Tool interface.....	28
Figure 22: Properties Editor	29
Figure 23: Specification Editor in PSM Tool interface.....	31
Figure 24: Specification Tab for Process	32
Figure 25: Specification Tab for Link.....	33
Figure 26: Specification Tab for Junction, Input / Output Node.....	34
Figure 27: Process Parameters Tab	34
Figure 28: The Node Input-Output Tab.....	35
Figure 29: The Stage Input-Output Tab	36
Figure 30: The System Input-Output Tab.....	36
Figure 31: Discrete systems API Lifecycle.....	38
Figure 32: Discrete PSM API.....	38
Figure 33: Discrete API input .json format	39
Figure 34: Continuous systems API Lifecycle.....	40
Figure 35: PSM API Calls	41
Figure 36: PSM API calls regarding Model	41
Figure 37: BRC Bay 3 Shopfloor structure.....	42
Figure 38: Petri net model of the upper layer of BRC Shopfloor.....	43
Figure 39: Petri net model of a specific scenario in BRC Pilot	44
Figure 40: Structure of the Process Modelling and Simulation output .json file	47
Figure 41: CONTINENTAL Timisoara plant shop floor structure	48
Figure 42: Petri net model of the upper layer of CONTINENTAL Pilot Shopfloor.	49
Figure 43: Structure of the Process modelling and Simulation output .json file	51
Figure 44: TUPRAS Model Levels of Detail.....	53
Figure 45: TUPRAS Level 2 model.....	55
Figure 46: PLT-63 MQD Debutanizer process parameters.....	56

Figure 47: PLT-63 LPG DEA Process Parameters56
Figure 48: F6 Input Flow Composition57
Figure 49: Tank Process Parameters58
Figure 50: Comparison of Simulation and Optimisation Results58
Figure 51: Results grouped per Resources59
Figure 52: Total Energy required59
Figure 53: Results in Tabular Format60

List of Tables

Table 1: Model Properties30
Table 2: BRC pilot input description.....46
Table 3: Percentage of machine usage52
Table 4: PSM API functionalities.....67

1 Introduction

1.1 Purpose and Scope

The objective of Task 4.3, “Systemic Cognitive Models Prototypes”, is the development of a methodology for dynamic modelling and simulation of alternative scenarios regarding the operation of parametric process models of the pilots. Outputs from the associate services of FACTLOG (optimisation, analytics, knowledge-graphs) are exploited to make the implemented models more realistic, study complicated system and components behaviours, and evaluate alternative system operation strategies during the process of decision making.

This document corresponds to the Deliverable 4.3 “Systemic Cognitive Models Prototypes” and summarises the work undertaken for:

- Formalization and finalisation of the Input and Output Datasets required for industrial process modelling and simulation.
- Capabilities extension and adaptation of Process Simulation Modelling tool, initially developed in T4.1 to serve the specific characteristics and the individual needs of FACTLOG Pilot cases.
- Deployment and final testing of the Application Programming Interface, used for integrating Process Simulation Modelling module into the Digital Twins Platform of FACTLOG.
- Development of high-level programming language refined parametric Process Models for FACTLOG pilot cases based on pilot final input and requirements and associate services data.
- Interconnection and interaction with associate services (Optimisation, Analytics and Knowledge Graphs) in the context of the FACTLOG Digital Twin platform.

1.2 Relation with other Deliverables

This deliverable extracts information from the descriptions of the pilot use-cases from D1.1: Reference Scenarios, KPIs and Datasets. It also has a direct connection with all WP4 Deliverables, but especially with D4.1: Process Modelling Methodology which serves as the basis for the work done under T4.3 and presented here. In particular, D4.1 introduces the theory and the process modelling and simulation mechanisms and tools, while in the current deliverable the analytical, parametric process modelling and simulation methodology is developed, applied to the pilot industrial systems, tested and cross-validated to provide realistic results.

Information regarding the integrated services, whose outputs are necessary for implementing the methodology introduced in the current deliverable, are extracted from D5.2: Robust and energy-aware planning and scheduling, D4.2: Knowledge Graph Modelling and D4.4: KG-based analytics for process optimisation. Finally, D4.3 provides information for system integration deliverables (D6.3 – D6.5), system installation and testing deliverables (D7.1, D7.2), cognition-related deliverables (D3.3, D3.4, D7.4) and exploitation, training, educational material, business plans and dissemination activities (D8.6, D8.8, D8.9, D8.10, D8.12, D8.13).

1.3 Structure of the Document

The overall structure of the document follows the typical format of FACTLOG Deliverables. This specific document is structured as follows:

- **Chapter 2** provides an overview of the methodological framework used to perform process modelling and simulation. It explains the main steps of the methodology in detail, referring to constraints and required data and tries to present the interactions with other associated FACTLOG services.
- **Chapter 3** explains the Dynamic Process Modelling approach. The first part of this chapter refers to the two layers of the discrete cases models that are implemented using Petri nets (and variations) and discrete event model simulation with Python, while the second part explains the modelling of continuous cases implemented using PSM Tool which has been implemented from TUC to support FACTLOG project needs. Finally, the deployed APIs for both discrete and continuous cases are explained.
- **Chapter 4** presents the Process Modelling and Simulation per pilot case (two discrete and one continuous), the required static and dynamic input data, the calculated outcomes, their physical meaning and potential extensions.
- **Appendix I – V** contains material regarding an analytical PSM Tool application example, API functionality, and detailed examples of input data and simulation results for BRC and CONTINENTAL pilot cases.

2 Methodological Framework

Figure 1 presents in a glance the general process modelling and simulation methodology that is followed in the current project. Minor differentiations or adaptations of this procedure are possible due to the type of the system under study (Discrete or continuous) or pilot-specific characteristics. Three distinct sets of data (Static data, Dynamic Data and Associate Service Outputs) are used in different stages of the Process Modelling and Simulation method as inputs, and specific outputs (in the form of suitable KPIs) are produced.

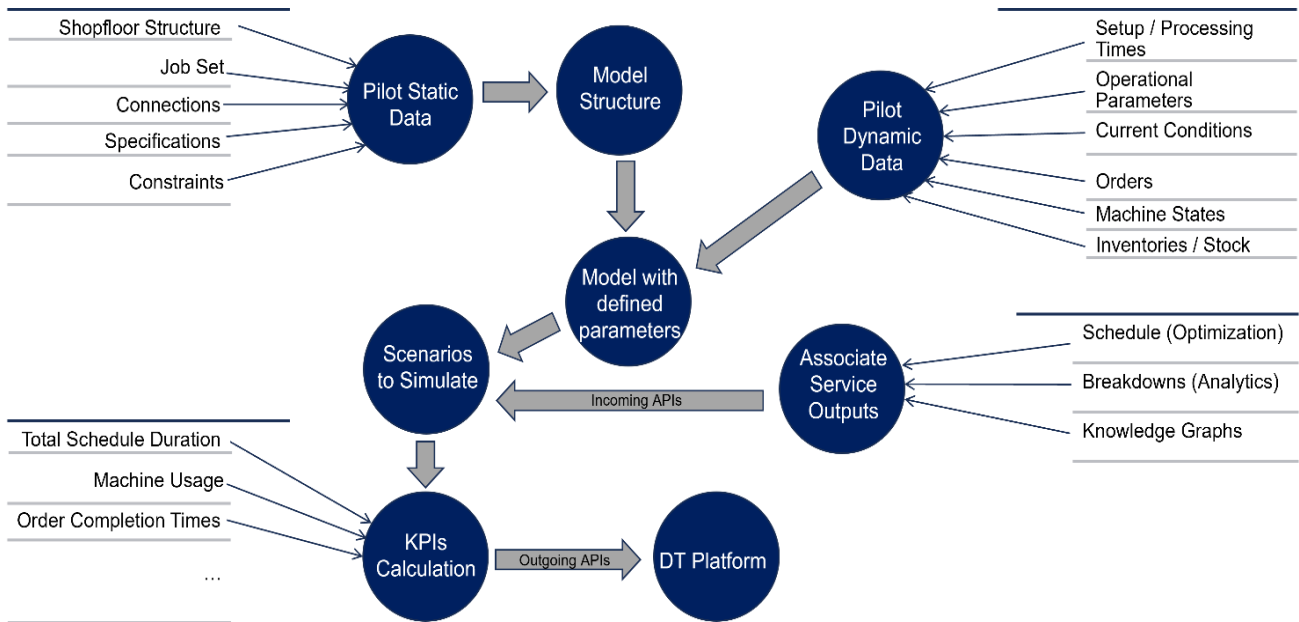


Figure 1: Process modelling and simulation methodology main stages

Initially, the structure of the model has to be defined with respect to certain types of static data provided by the respective pilot. At this stage, the main target is to represent the entities of which the physical system is composed as well as their physical connections and interactions and to define the channels for the exchange of resources and information between them. Data used in this stage are static and not updated frequently (for example when machines are added or removed from the system). Some of the most common types of such static data, with respect to their role, include (but are not limited to) the following:

- Industrial system structure data. These include the available resources of the physical system, possible categories (families) and their capacities. Physical connections (permanent or product specific) between resources are also necessary in order to construct the structure and to define the type of the system under study (e.g., production line, job shop, continuous flow process production system, and continuous flow assembly production system) as the simulation algorithm takes into consideration this characteristic. To obtain these types of information, flow charts and non-structured data can be used as well as knowledge graphs that provide a collection of interlinked descriptions of the entities of the industrial system. Knowledge graphs provide domain knowledge representation to be reused efficiently and prevent waste of time and money [1].
- Product-related static data. Products belong to families with similar or partially common characteristics (setup durations, tools and resources used, specifications

and physical or chemical characteristics, etc.). Also, setup times and types (e.g. sequence dependent, family dependent etc.) and ideal production times of all possible products in each machine must be defined. Finally, if ancillary equipment is considered, data concerning its use should be provided (for example, speeds and transfer capacities of cranes for moving parts between resources and buffers).

- Constraints regarding the use of specific subsets of resources for the performance of a respective subset of jobs. In this category also, priorities between products must be recorded and constraints that have to do with possible flexibility between production stages, maximum batch sizes, machine operational limits, resource sharing, resource management and control policies followed, etc.

In the second stage, quantitative parameters are added to the model. This is necessary to represent a specific dynamic situation in the system and define the initial state of the scenario(s) that will be evaluated through simulation. Such dynamic data types comprise the following:

- Industrial system dynamic data. Such data have to do with the current states of the resources (if machines are operational or under maintenance), current values of system parameters (temperatures, pressures, etc.), the current types of processes performed in each machine (as in many cases, setup times are sequence-dependent) and machine availability because of scheduled maintenance activities in the time horizon in which the system is studied. In addition, if certain resources operate with efficiencies lower than the ideal ones (for safety or energy consumption reasons, for example), this has to be taken into account at this stage in order to define the realistic values of working speeds and process durations.
- Set of orders under schedule. This is the most important type of dynamic data as this would be used as input from the optimisation service to define the schedule that will be then simulated (in fact, this is the definition of the specific problem under study every time as the first stage is static and is not repeated generally). Production order set represents the customers' requirements (external or internal according to market needs forecasting) and refers to the types and quantities of products that have to be produced as well as to their due dates. In many cases production orders are composed of different jobs that need to be processed in different machines and produce products with (partially) different characteristics. At this stage, setup delays and production durations for all the possible resource product combinations are also defined with respect to the ideal ones provided in the previous stage and the resource operation efficiencies.
- Initial raw materials, in-process products, products, tools and other material inventories. This refers to the initial levels of internal buffers of the system as well as in process products in the machines and is used to define some additional operational constraints that have to be taken into account to improve the system's efficiency, as machine idleness may increase in other cases. In addition, raw materials quality is taken into account as specific parameters of the industrial system's operation have to be specified according to this (for example, in a chemical plant the quality of the raw materials affects operational parameters such as temperatures and pressures).

The third stage refers to adding extra information and defining the values of additional parameters to the model whose initial state has been defined through incoming APIs. Such

input data mainly comprise output results from the operation of the associated services in the FACTLOG system. They are necessary to describe the scenario(s) under study and make them even more realistic. The main types of such data are:

- Optimisation output. In the Discrete cases, the optimisation service provides the number and sequence of different jobs performed in each machine (schedule). This can be different for every machine or common for every production line with respect to the specific structural characteristics of the case considered. The proposed schedule might be updated if the objective function changes according to users' needs. In the case of Continuous process industries based on historical data from units, models are created, and their behaviour is described. Process modelling and Simulation service produces a set of alternative operational scenarios with a given step for each operating condition. These scenarios are then transferred to the Optimisation module to be consumed. The optimisation module utilises them to solve the corresponding on-specs recovery problem [2]. Then the alternative scenarios are evaluated, and the dominant one is selected for simulation in order to define the production strategy.
- Non-scheduled maintenance activities (machine breakdowns) from analytics. Analytics can provide data concerning the appearance of non-scheduled machine breakdowns as well as the duration of their repair in the time horizon of the scenario under study. Analytics and machine learning models typically use historical data to detect patterns and predict future outcomes when they receive a company's data.
- Data received from knowledge graphs. According to [3] building a knowledge graph can be advantageous for visualisation, insight derivation, as well as for detecting anomalies and for computational efficiency since graph algorithms can be orders of magnitude faster compared to conventional database algorithms in many applications. Knowledge graphs help Information Technology (IT) to operate with interoperability and standardisation. In the case of FACTLOG Knowledge graphs extract knowledge from the following sources to formalize the pilot cases: 1) scenario description; 2) model formalism; 3) optimisation formalism; 4) anomaly detection.

When all the above types of data are available, the simulation process of alternative scenarios can begin. Alternative scenarios can be defined from a specific user or from a service (for example Optimisation or Analytics) in order to evaluate alternative strategies or check possible root causes of certain behaviours with respect to specific metrics. There are some common KPIs that are calculated (total duration of the simulated scenario, percentage of machine usage for every machine of the system, order completion time, impurities concentration, system throughput, energy consumption) but it is possible to extend this KPI set with respect to the specific needs of a pilot or associate service. These KPIs are important also for the Pilots as through them they can monitor machine availability and idleness, detect bottlenecks, detect quality variations, see when each order is completed and can be transferred to the customers and much more interesting information.

Visualisation of the proposed process modelling and simulation methodology in the context of a DT is shown in Figure 2. In this the interaction and exchange of information between the physical and the digital subsystems are depicted. In addition, actions can be enabled from the digital to the physical subsystem through the use of appropriate actuators.

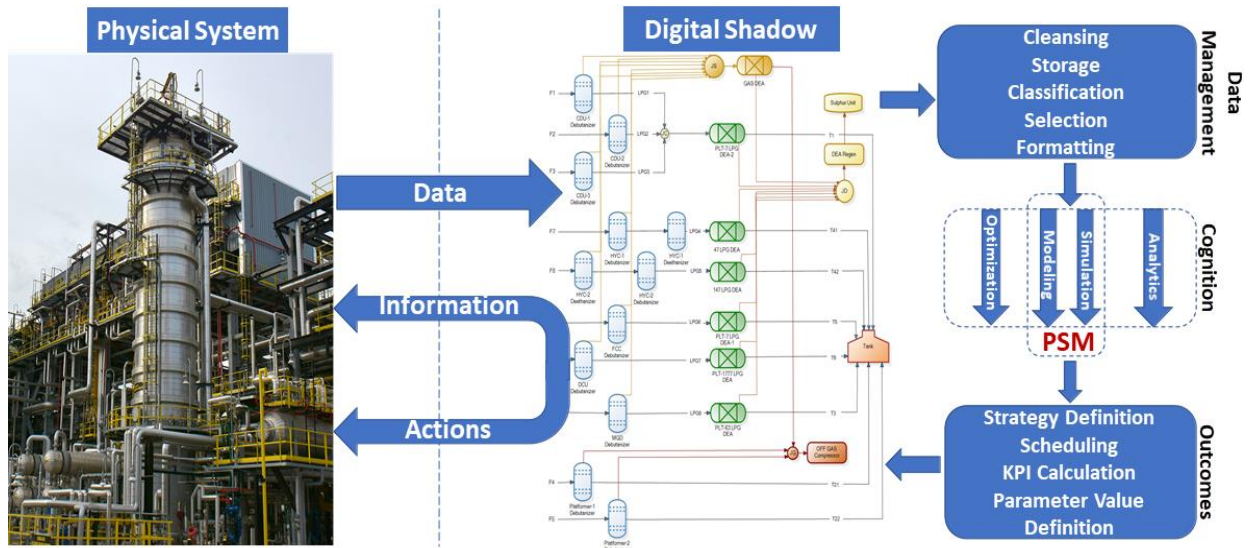


Figure 2: Modelling and Simulation methodology and associated services interactions

3 Dynamic Modelling and Simulation

3.1 Discrete Process Simulation and Modelling Implementation

The methodology followed for modelling of discrete industrial systems is a modular one as this serves hierarchical modelling. The need for hierarchical modelling has to do with the increased complexity that typically industrial systems have. The main idea behind hierarchical modelling is that extended and complicated models can be divided into more compact and easily analysed interacting sub-models. In this way, problems are solved in an easier way and also the produced models are highly reusable, can be updated more efficiently and the information can be presented in different levels. In the literature there are applications such as [4] where even six different layers of a Petri net model are implemented in order to describe the structure and the activities of a multi-agent robotic system.

In the currently followed approach, the upper level of the model is implemented with Petri nets and extensions while for the lower level of the industrial system model Python programming language is used. The upper level of the model is used in order to represent the static structure and the interactions between the system entities while the lower one is used to perform the scenarios simulations and calculations of KPIs. The main reason for this is that the exact entity models are dynamic, as their complexity is defined with respect to dynamic parameters such as the number of jobs performed in each entity as well as the number of machine breakdowns and maintenance activities. As in a typical industrial application this number can vary from hundreds to thousands even for short time periods, it is obvious that the use of a classical graphical modelling and simulation method like Petri nets would not be efficient (the number of states and nodes of such a model would be immense). Python has the necessary flexibility that makes possible to efficiently manage such concepts following a number of well-defined rules and constraints according to the cases under study. These rules and constraints in fact also can be represented by Petri net structures but this level of detail in graphical representation is not necessary. In addition, the existence of already implemented Python libraries increases significantly the flexibility and the speed of such a tool compared to a typical graphical Petri net simulator.

3.1.1 Petri Net Models

Petri nets are a popular mathematical and graphical tool widely used for modelling, analysis, synthesis, performance evaluation, simulation and control of processes and systems typically considered as discrete event. They allow the representation and study of the structure as well as of the dynamic behaviour of systems and processes and have been proven to be a powerful tool for studying system concurrency, sequential, parallel, asynchronous, distributed deterministic or stochastic behaviour, resource allocation, mutual exclusion and conflicts [5], [6], [7]. Representation of the dynamic state of the modelled system enables the simulation of certain scenarios as defined from the definition of the initial state of the system and the state change rules (discrete event simulation).

Ordinary Petri nets (OPNs) are bipartite directed graphs formally defined as the following five-tuple: $PN = \{P, T, I, O, m_0\}$. The respective sets for the two types of nodes are $P = \{p_1, p_2, \dots, p_{np}\}$ which is a finite set of places and $T = \{t_1, t_2, \dots, t_{nt}\}$ which is a finite set of transitions. $(P \cup T) = V$, where V is the set of vertices and $(P \cap T) = \emptyset$. In Petri nets, places describe conditions (e.g., for control purposes) or resource availability. Transitions represent events or actions and arcs (that may have weight equal or greater than one) direct connection,

access rights or logical connection between places and transitions. Places are the passive element of the PN while transitions the active ones. The element I of PN represents an input function, while O denotes an output function. Finally, m_0 is PN's initial token distribution referred to literature as marking [8]. Places and transitions are connected interchangeably while T, define node partitions of the network (i.e. nodes of the same type are not linked). The most important PN properties (reachability, safeness, k-boundedness, conflicts, liveness, reversibility, persistency, deadlock-freeness, P- and T-invariants) capture precedence relations and structural interactions between system components [9], [10].

Inclusion of time delays (constant, following some distribution, or random according to the actions) in the transitions of the initial formalism implements T-timed PNs (TPNs). TPNs are defined as $\{P, T, I, O, m_0, D\}$ with the first five responding exactly to the same features as in the case of OPNs and D representing time delay that is a function from the set of non-negative real numbers [6], [7].

The main structural elements (building blocks) of a Petri net are the following:

- Discrete Places (O) describe system states (represent conditions e.g. for control purposes) or resource availability.
- Discrete Transitions (■) represent events or actions that change system states.
- Tokens (•) describe the dynamic condition of the system. Tokens reside in places, move through transitions and define the marking of the net.
- Connection Arcs define relations between system components as well as local states and events. Three arc types exist:
 - Standard arcs, drawn as usual (\rightarrow), representing connections between entities, define flows of materials, sequence of events etc.
 - Inhibitor arcs are represented by arcs whose end is marked with a small circle ($\text{---}\bigcirc$). Inhibitor arcs disable transitions when input places connected to them contain number of tokens equal to their weight.
 - Test arcs are represented as arcs with dashed line ($\text{---}\text{---}\rightarrow$). Test arcs enable certain transitions when input places connected to them contain number of tokens equal or greater to their weight and allow firing of such transitions without consumption of the tokens residing to these places.

Transitions become enabled when all their input places contain a number of tokens at least equal to the weight of the arc connecting a place to a transition and fire by removing tokens equal to these weights from all the input places and adding tokens to all the output places according to the respective arc weights. Transition remains enabled for a number of time units equal to the delay of the transition before it fires. When a discrete transition fires, it “consumes” from each input place a number of tokens equal to the weight of the respective arc connecting the input place with the transition, and “produces” to each one of the outputs places a number of tokens equal to the respective arc connecting transition to these places.

3.1.2 Model Implementation in Python

In order to implement the models of the Discrete cases, in contrast with the Material Flow Networks followed in the Continuous cases described below, a more generic approach has been selected, utilizing high-level programming languages, existing tools and libraries and re-use of software developed during the project. Among the various high-level programming languages, the one that was the most appropriate as well as the one being used by the

optimisation partners, who provided us with pre-processing scripts was Python programming language. Python serves the intended purpose adequate enough, being easy to use, portable and very easy to write but most importantly debug and perform changes in existing software as the project evolves and new data come from the pilot cases. The abundance of the supporting libraries means that it is very easy to perform any required task with Python quickly and efficiently, letting you focus on the tailor-made software that the project requires to be developed.

Specifically, in the implementation of the models we have used libraries such as pandas [11]; which is one the most well-known data manipulation and analysis library, numpy [12]; which is a library used to work with matrices, linear algebra and Fourier transforms, json; a built-in package to work with *.json* files and csv; a package to work with *.csv* files. All of the aforementioned libraries and packages as well as Python itself are free to use and open source not only for educational but also for commercial use.

The Python models developed for the discrete cases are based on two main pillars; the incoming data from the pilot plant; and the associate services output data. All of the above-mentioned data have been analytically described in Chapter 2. Methodological Framework This makes it apparent that the modelling and simulation service on the discrete cases are linked to the incoming data and the optimisation service; and potentially any hinder to obtain any of the two would not allow to produce a modelling and simulation outcome.

On a more technical level, both pilot input data and optimisation results are in a structured *.json* format; predetermined among the partners. These *.json* inputs are consumed from the Python software developed by TUC, appropriate calculations are performed (according to the pilot structure; different for BRC and different for CONTINENTAL) and the simulation result is produced. During the calculations all the necessary assumptions discussed and agreed among partners have been carefully considered. After the simulation is completed, calculation of important KPIs takes place, such as total time of usage per machine, percentage of machine usage in comparison to the total completion times and completion time per order. These are some representative and common metrics that can be extracted from the simulation; however, it is possible to have tailor-made metrics and KPIs per pilot and according to the request of the end user. The result in question is compiled into a structured file in *.json* format that consist of three vectors; *simulationTime*, *machinesUsed* and *completionTimePerOrder*. The structure can be seen in Figure 3 . More information on the simulation results and the output *.json* file can be found in 4.1.3 for BRC case and 4.2.3 for CONTINENTAL case.

```

{
  "simulationTime":
  [
  ],
  "machinesUsed":
  [
  ],
  "completionTimePerOrder":
  [
  ]
}

```

Figure 3: Discrete Simulation result .json structure

3.2 Continuous Process Simulation and Modelling Implementation

3.2.1 PSM Tool

3.2.1.1 Introduction

The Process Simulation and Modelling suite (PSM) has been developed to address the process modelling and simulation requirements of continuous process industries. It is based on the principles of Material Flow Networks (MFN), which model material and energy flows in production chains. This analysis of MFNs can provide information on the resources used and the corresponding emissions, which can lead to environmental and economic value estimations. All model entities are organised into a Hierarchical Inheritance Registry that provides prototype reconfigurable building blocks for building any industrial system model. In each system described by a PSM model, generic materials (raw materials, energy, products, etc.) are processed and transformed into other materials. The state of the system is characterised by the flows of these generic materials. The main elements of a PSM model are two different types of vertices called processes and places, which are connected with links and can be grouped into stages. A description regarding each of these components follows:

- Processes are the equivalent of Petri net (PN) transitions, representing an activity or task, which takes in all the required materials (input) and, as a result, generates new or modified materials (output). This is how processes link material consumption to production. Process specifications can summarise underlying activities in terms of simple relations between input and output flows (ratios), algorithms that perform the required calculations or even machine learning model of the process.
- Places can be interpreted as storage for resources within the network. They could be *Input Nodes* representing the initial sources of resources flowing towards processes, *Output Nodes* representing the final resources flowing from processes and *Junctions*

connecting processes, acting as an output node for one process and input node for the following process.

- Links, which are the equivalent of PN arcs, represent a mean by which material can flow from a place (*input node* or *junction*) to a process or from a process to a place (*output node* or *junction*).
- Stages serve as containers for grouping multiple model elements. They are usually needed in cases where some parts of a model can be aggregated as an individual unit.

The PSM suite includes both a desktop/standalone (PSM Tool) and a web application (PSM APIs). The desktop application has been developed under the .NET framework using C#. Its core functionalities are (a) the design of the overall system model, following a graphical approach, by drawing the elements as well as their interconnections on a canvas; (b) the specification of material and energy flows to and from processes as well as interrelations between input and output; (c) the evaluation of alternative scenarios by defining model parameter values under given conditions; (d) the calculation of the flows or parameters system-wide as well as between process units through simulation; (e) result presentation and reporting not only in tabular format but also in other common formats for further processing.

The desktop application communicates and interacts with the web application back-end through an Application Programming Interface (API) for uploading and process industry models and scenarios with alternative parameters. The web application exposes functionality through the API that allows users and/or systems to simulate, monitor in real-time, and modify the operation of a process industry model on demand. More importantly, it transforms the process industry model into an active component that can bi-directionally interact with the physical systems. It achieves that by exposing functionality, allowing various parameters to be configurable on the process level based on the initial modelling.

3.2.1.2 Structural Components

As mentioned before, the PSM Tool has been adapted from an existing tool to serve the intended purposes of FACTLOG project regarding modelling and simulation. The use of MFN makes it ideal for the modelling and simulation of TUPRAS case. Due to that, the basic entities used in the tool have been adapted to resemble well established chemical engineering symbols in order to facilitate the easier use of the tool from the pilot plant engineers. Figure 4 summarises these basic entities such as tray columns, fluid contacting vessels, tanks as well as the common input, output and junction nodes.

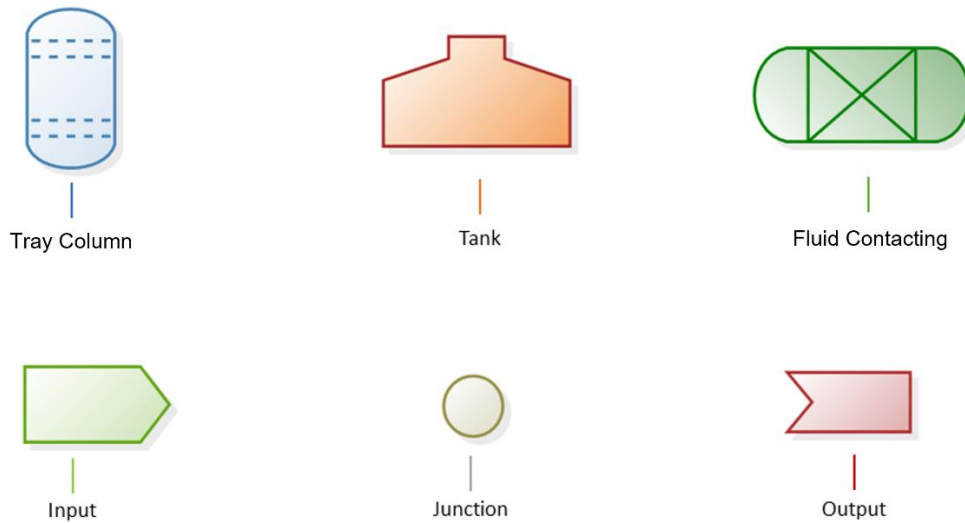


Figure 4: Continuous pilot case structural components

3.2.2 PSM Standalone Tool

The Process Simulation and Modelling standalone Tool (PSM Tool) is the direct extension of a standalone tool initially introduced in [13] – [14]. It has been developed in the .NET Framework, utilizing Visual Basic and C# and it is currently distributed as a freeware; i.e. the tool is free to use for non-commercial purposes (research, academic, etc.), however its source code is not made publicly available and any modification, redistribution by third parties or reverse engineering is prohibited. It should be mentioned here that PSM Tool uses third party libraries for the creation of the graphical user interface and controls, specifically Diagramming for WinForms from MindFusion and WinForms Component from DevExpress.

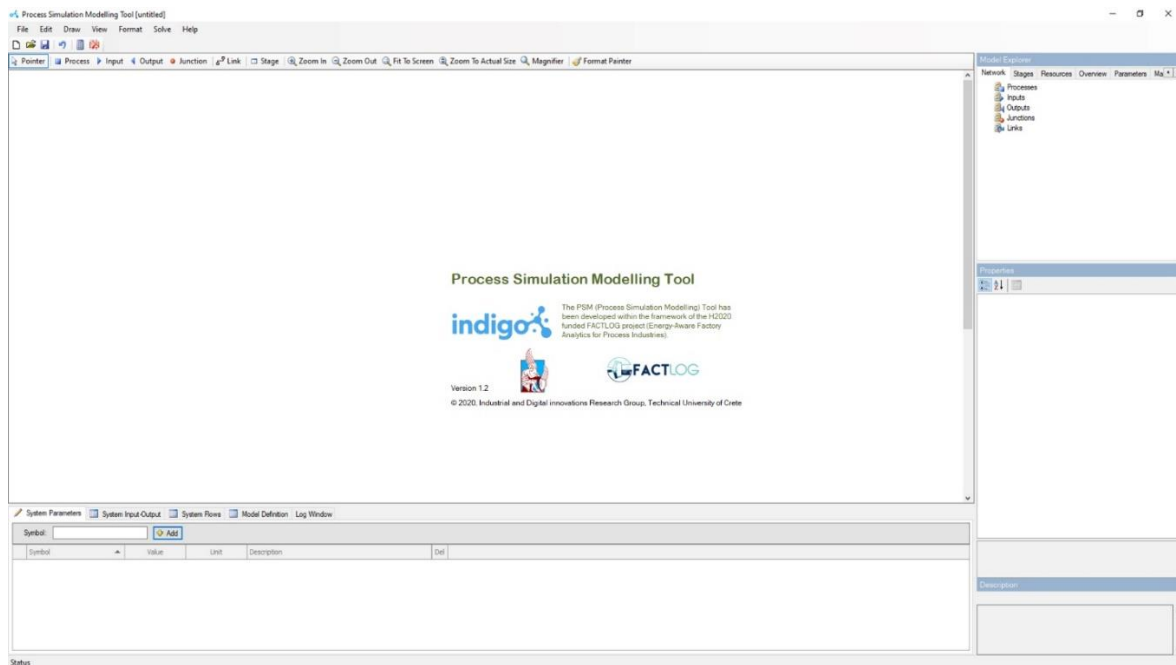


Figure 5: PSM Tool Interface and Splash Screen

PSM Tool is quite straightforward and ease to use since it does not require any programming knowledge or special skills to create a new process model. It is purposefully built this way to allow any user regardless its expertise to use it and build a model no matter of the modelled system’s complexity.

The main user interface is divided in 5 areas: the Component Toolbar (1), the Model Editor/Diagram Area (2), the Model Explorer/Resources Manager (3), the Properties Editor (4) and the Specification Editor (5). These 5 areas can be seen in Figure 6.

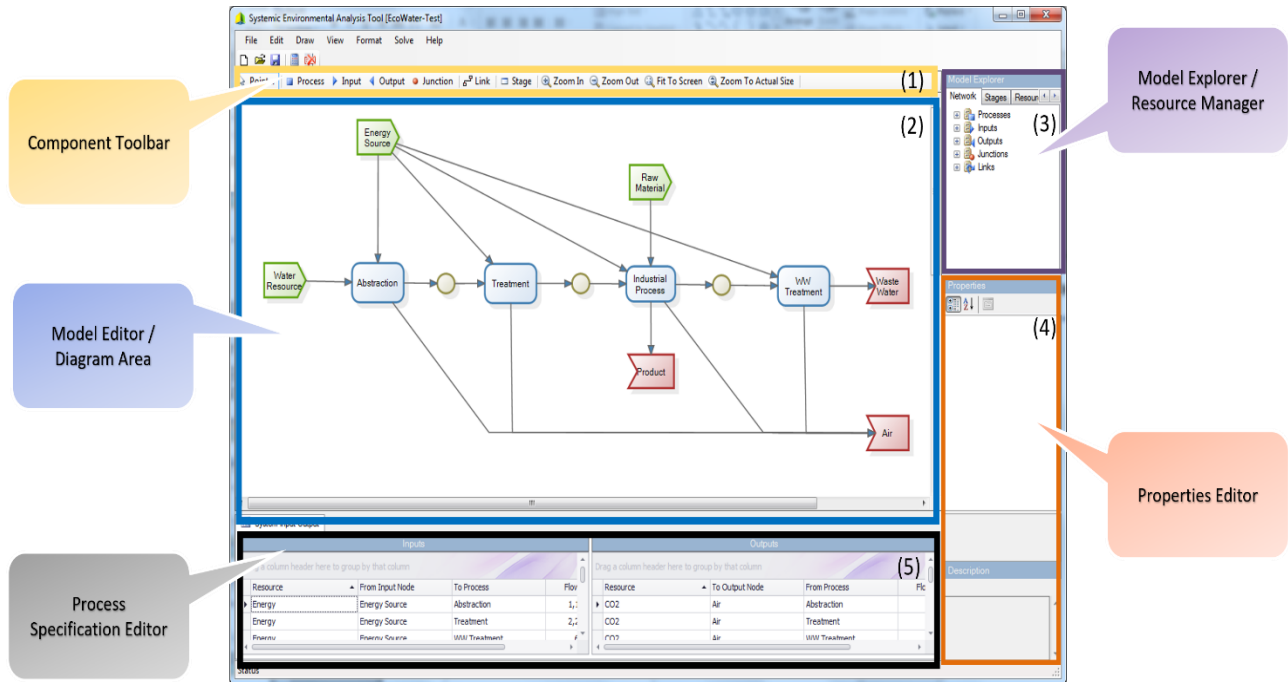


Figure 6: PSM main user interface areas

Following is a detailed description of each of these five areas. This part of the document, together with PSM User Guide, which is available to all partners in FACTLOG’s shared folder, can be used to assist any interested user of the PSM Tool.

3.2.2.1 Component Toolbar

The component toolbar (Figure 7) provides the means for building the graphic representation of the system network and contains six main elements. At any given time, the active state of each drawing mode is shown at the toolbar. When the user double clicks on any of the component toolbar icons and the chosen item can be used to create multiple components. To exit the multiple insertion mode either press the “Esc” key on the keyboard or click on the pointer component with the mouse.

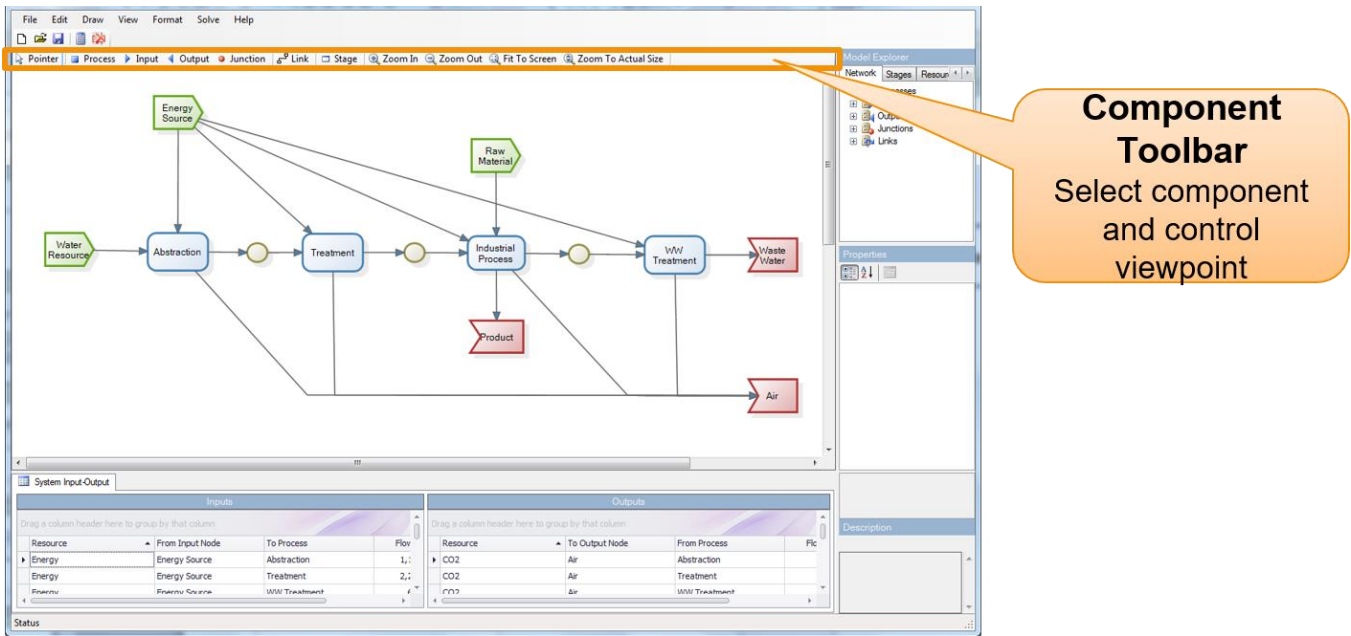


Figure 7: Component Toolbar in PSM Tool interface

The Pointer state mode (Figure 8), is the default state of the cursor. After each step made for creating a system diagram the cursor returns automatically to the “Pointer State Mode”. When this is active, by clicking inside the model editor (3.2.2.2) the user can select any node.

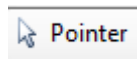


Figure 8: Pointer

The Process state mode (Figure 9) is used to draw new processes. A process is the most important element in the model. Designed either as a blue rectangle or by symbols and figures selected according to each case study (e.g., for TUPRAS refinery chemical engineering symbols have been selected) represents an activity in which input flows are converted into output flows.

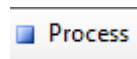


Figure 9: Process

The Input state mode (Figure 10) is used to draw new input nodes. Input nodes have the form of green composite node, serve as the source points of flows towards processes. An input node can only be linked with a process via their right side.

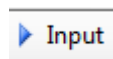


Figure 10: Input

The Output state mode (Figure 11) is used to draw new output nodes. Output nodes have the form of red composite node representing target places of flows from processes. An output node can only be linked with a process via their left side.

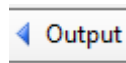


Figure 11: Output

The Junction state mode (Figure 12) is used to draw new junctions. Junction nodes are designed as a circular node. They act as both input and output nodes, or in other words, they serve as connectors between processes.



Figure 12: Junction

The Link state mode (Figure 13) is used to draw links between nodes of the system. Links illustrate the flows from or to between the nodes of the system. Multiple resources can flow within a link.



Figure 13: Process

The Stage state mode (Figure 14) is used to draw new stages. Stages serve as containers for grouping multiple model elements. They can be used in cases where part of a model can be aggregated as an individual unit.



Figure 14: Stage

Component Toolbar also includes some viewpoint control buttons such as the Zoom In, Zoom Out, Fit to Screen and Zoom to Actual Size Buttons that are affecting the way the model is being shown into the Model Editor area of the tool.

3.2.2.2 Model Editor / Diagram Area

The Model Editor, presented in Figure 15 is the most important part of PSM Tool. This is where the system model is created and maintained. A description of each function can be found after the screenshot below.

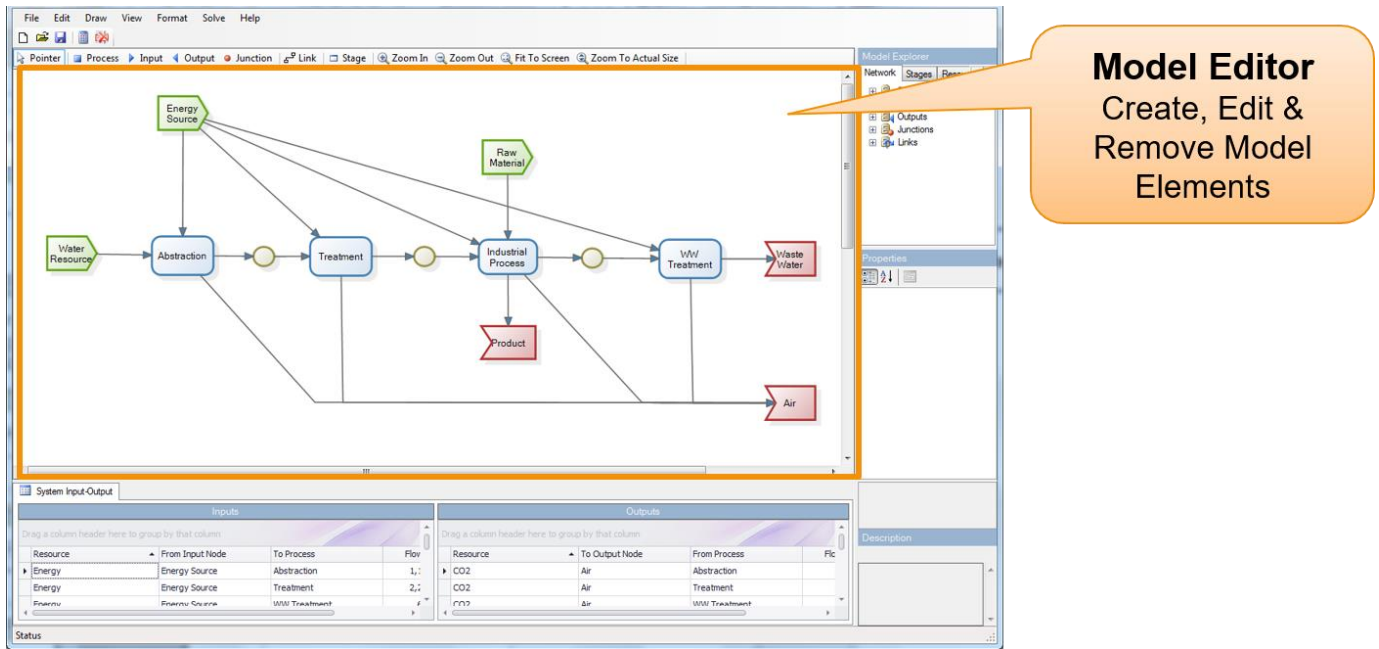


Figure 15: Model Editor / Diagram Area in PSM Tool interface

Inserting Components

To insert a new component, click on the corresponding button in the component toolbar (3.2.2.1). The component is created by clicking anywhere in the diagram area and then with left click you can select it and drag to set the desired size of the created component. After the creation of the component, the draw state reverts to the default “Pointer state mode”. You can insert multiple components of the same type by double-clicking in the appropriate button in the component toolbar. You can then exit the multiple insertion mode by either pressing the “Esc” key or clicking on the pointer button to return to “Pointer state mode”.

Selecting Components

To select a component inside the Model Editor, click on the component while in “Pointer state mode”. To select multiple components, click on the draw area and drag with the mouse a rectangle area that contains all the components need to be selected. Please note that in a multiple selection there is always one “Active” component. The visual difference between an active component and a selected component in a multiple selection is that the active item has white-coloured marker points, whereas the selected items have grey-coloured marker points. The functional difference between an active component and a selected component is that any changes in size will be done with reference to the active component.

Moving and Resizing Components

To move a component, select it with left click and click and drag to move it with your mouse in any direction on the diagram area. To resize a component, click on the appropriate marker point and drag it to the appropriate size. Note that a “Stage node” will be automatically resized to fit its contents.

Formatting Components

To facilitate easier handling of the components in the Model Editor, PSM Tool offers a number of relativity functions. More specifically, by accessing the Format Menu item on the menu toolbar there is the ability to change the location or the size of the selected components relatively to the active component. In order to activate these functions a number of components has to be selected beforehand.

Deleting Components

To delete a single component or a number of components, select the component by clicking on it with the mouse and then press the “Delete” button on your keyboard. You can delete more than one component at the same time by holding the “Ctrl” button while clicking on the components with the mouse (to perform group selection) or by dragging a selection box around the area of interest and then pressing “Delete”. Keep in mind that if a component with links to other components is deleted, then the associated links are removed as well.

3.2.2.3 Model Explorer / Resources Manager

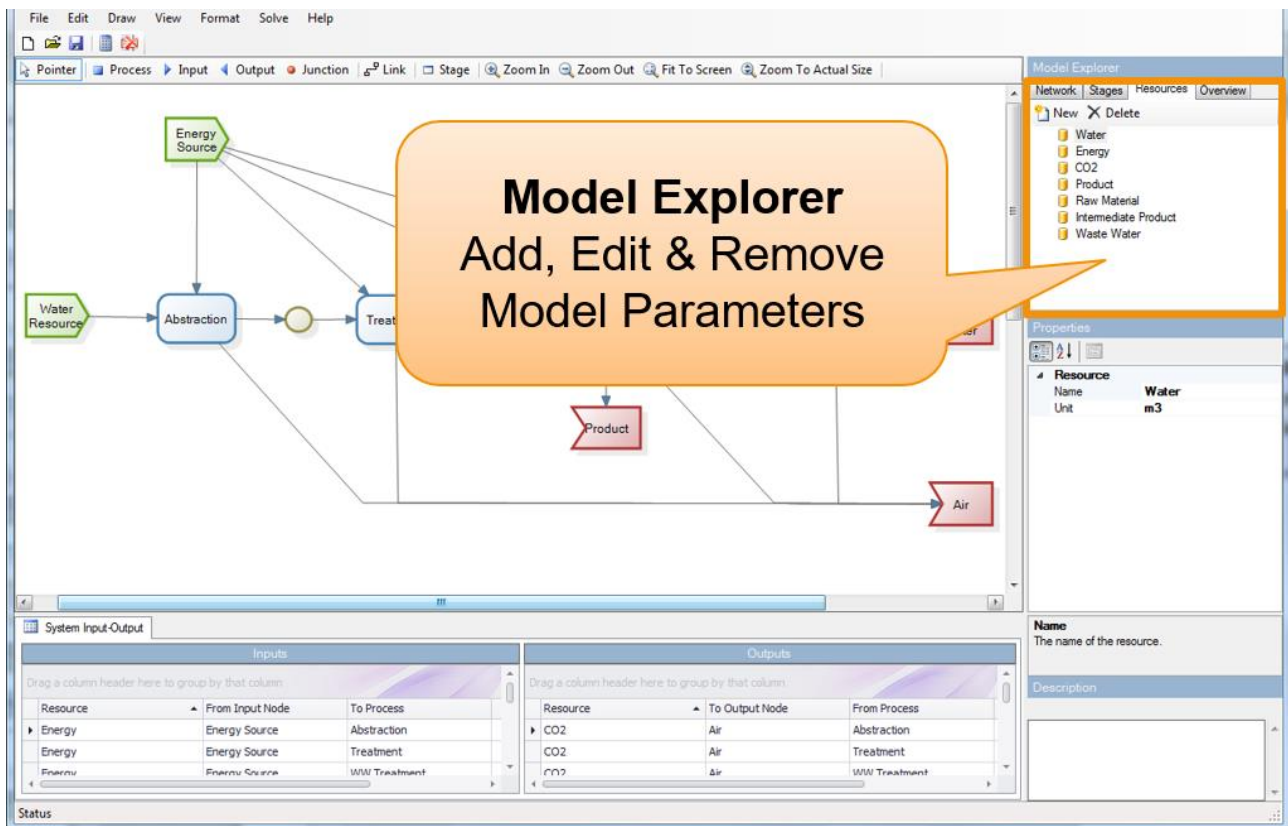


Figure 16: Model Explorer / Resources Manager in PSM Tool interface

The model explorer area (Figure 16) is located in the upper right corner of the PSM Tool. It contains four individual tabs, *Network*, *Stages*, *Resources* and *Overview*, each serving a distinct purpose. The model explorer area is an active window which gathers all the model information. When someone clicks on a process, input, output, junction or link in the created model, the related parameters are presented there. The user has the ability to enter data

and modify the model through the Model Explorer area, utilizing functionality that will be explained in detail below.

Network Tab

The Network Tab (Figure 17) shows the elements that exist in the model, grouped by their component type. When the user clicks on Process 1, for example, the Properties Editor as well as the Specification Editor are activated. The Properties Editor is an active window where the user can give, for example a name the selected component. In the Specification Editor area, you will notice that there are tabs that have a pencil symbol at the beginning, indicating that data can be entered there or a table symbol indicating that information is read-only available.

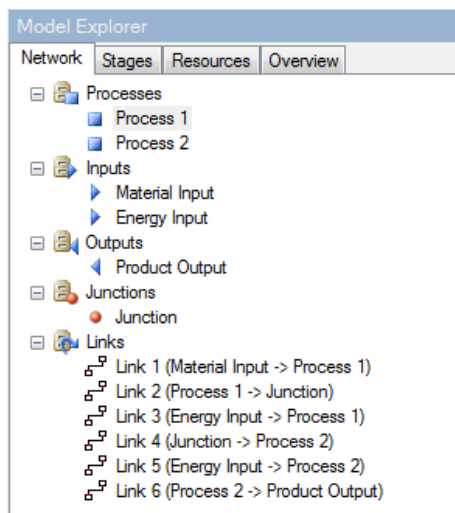


Figure 17: Network Tab

Stages Tab

The Stages Tab contains the processes of the model, along with the input/ output nodes that are part of each stage. An example of the Stages tab's overall layout is shown in Figure 18. As with the Network tab, when the user clicks on a Stage the Properties Editor and the Specification Editor are activated. The user can then add a stage name through the Properties Editor or modify the System Parameters through the Specifications Editor.

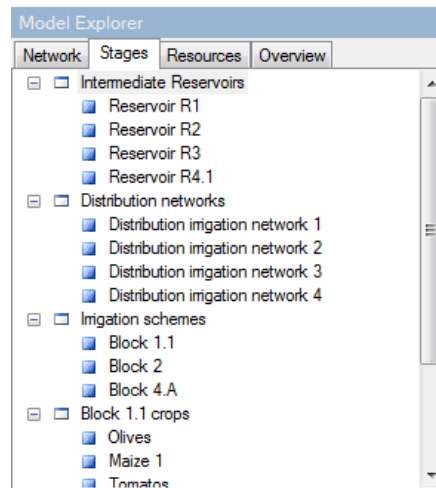


Figure 18: Stages Tab

Resources Tab

The Resources tab is used for creating, editing and deleting resources in the system model. As mentioned before, a *resource* is transferred to and from a *process* via a *flow* (or a *link* in terms of graphic presentation). The user by clicking on the Resources Tab activates a Properties window where the selected resource can be given a name, get a unit definition and even add a short description. The Resources Tab is shown in Figure 19.

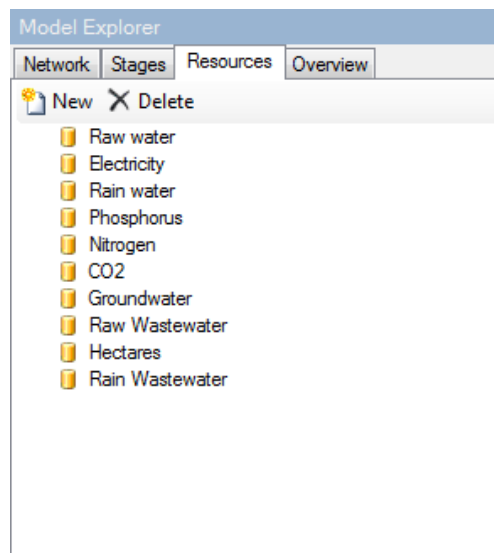


Figure 19: Resources Tab

Overview Tab

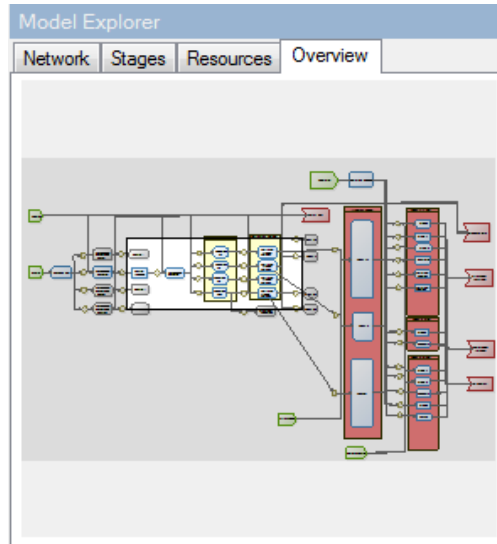


Figure 20: Overview Tab

The Overview Tab displays the Diagram Area current view location within the overall model structure. The user can click and move the *blue window*, representing the current view area, to change the viewpoint inside the model. The overall layout of the overview tab is shown in Figure 20.

3.2.2.4 Properties Editor

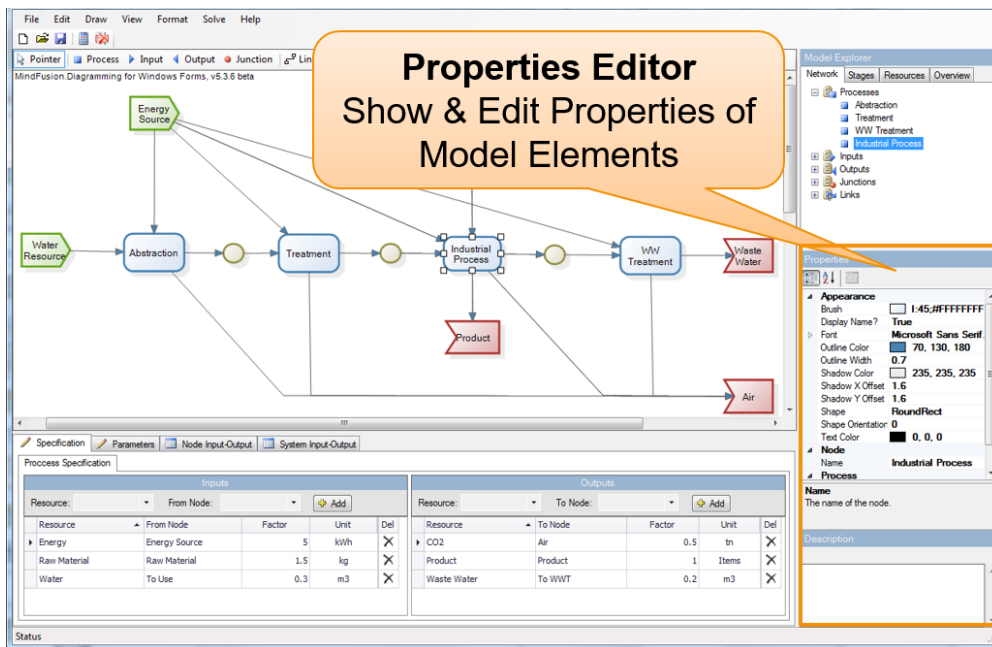


Figure 21: Properties Editor in PSM Tool interface

The Properties area is located in the lower right corner of the PSM tool interface, as seen in Figure 21. As mentioned above, every time a model entity is selected, the *Properties Editor* is activated and displays relevant information. The user can give a name to this entity or a

value through this pane. The term “model entity” refers to a resource or any model component as described above. The properties can be *categorized* or *sorted A to Z* by using the relevant buttons located at the top of the Properties Editor pane. There is also the option through the 3rd button to add pages in order to place the properties for this model.

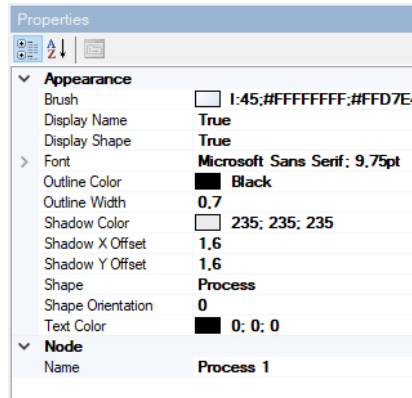


Figure 22: Properties Editor

In Figure 22 the properties that can be adjusted can be seen. They are mostly appearance related but keep in mind that the properties in the pane are being adjusted according to the selection of the model entity.

In Table 1 the complete list of properties and the corresponding model entities that use them are presented.

Property	Description	Model Entity
Name	The name of the model entity	Resource, Process, Junction, Input Node, Output Node, Link, Stage
Colour	The colour of the model entity	Link
Display Name	Show or hide the name of the model entity	Process, Junction, Input Node, Output Node, Link, Stage
Head Colour	The colour of the arrowhead shape of the model entity	Link
Head Shape	The arrowhead shape of the model entity	Link
Head Size	The size of the arrowhead of the model entity	Link

Property	Description	Model Entity
Shadow Colour	The colour used to paint the shadow of the model entity	Process, Junction, Input Node, Output Node, Link, Stage
Shadow X Offset	The horizontal offset of the model entity	Process, Junction, Input Node, Output Node, Link, Stage
Shadow Y Offset	The vertical offset of the model entity	Process, Junction, Input Node, Output Node, Link, Stage
Style	The style of the model entity	Link
Text Colour	The text colour of the model entity name	Process, Junction, Input Node, Output Node, Link, Stage
Width	The width of the model entity	Link
Brush	The brush used to fill the model entity	Process, Junction, Input Node, Output Node, Stage
Font	The font used to display the text of the model entity	Process, Junction, Input Node, Output Node, Stage
Outline Colour	The colour used to draw the outline of the model entity	Process, Junction, Input Node, Output Node, Stage
Outline Width	The outline width of the model entity	Process, Junction, Input Node, Output Node, Stage
Shape	The geometric shape of the model entity	Process, Junction, Input Node, Output Node
Shape Orientation	The shape orientation of the model entity	Process, Junction, Input Node, Output Node
Unit	The measurement unit of the model entity	Resource

Table 1: Model Properties

3.2.2.5 Specification Editor

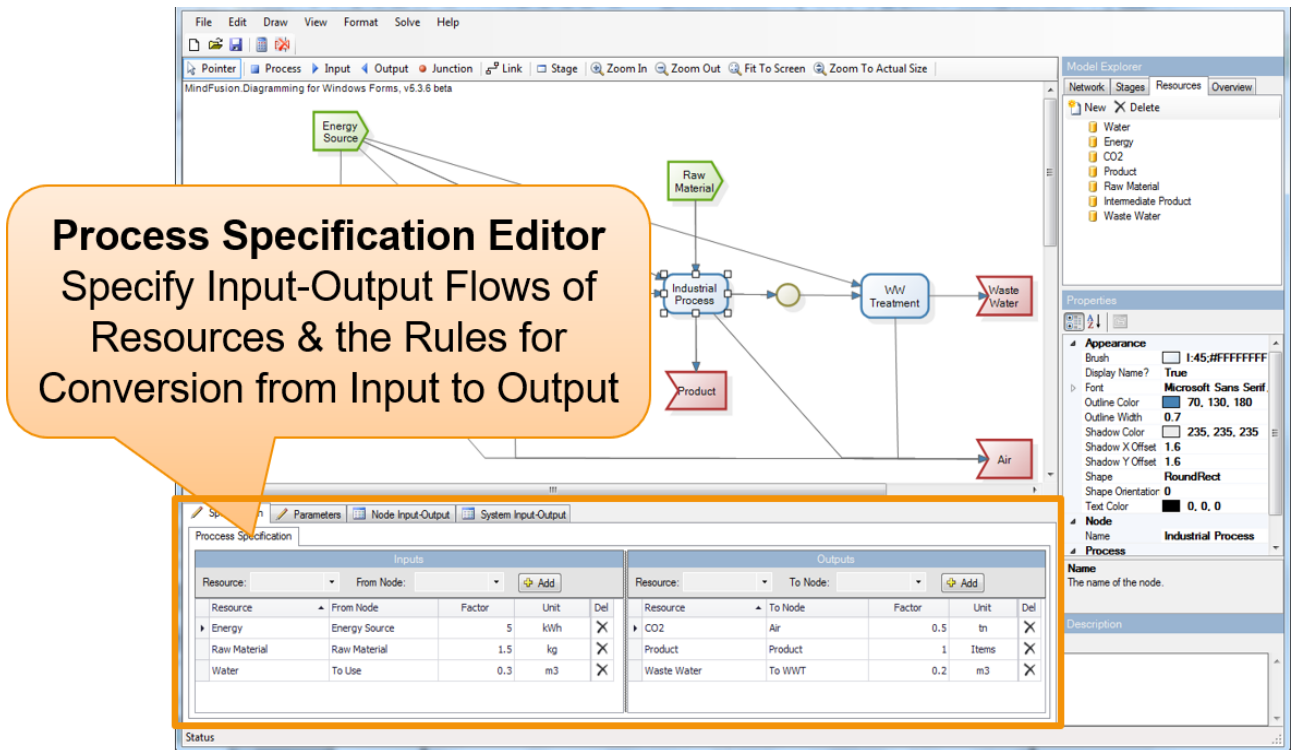


Figure 23: Specification Editor in PSM Tool interface

The Specification Editor (shown in Figure 23) is divided into two distinct tabs which are used for specifying the flows of resources to and from processes as well as viewing the results of calculations. In particular, the two distinct tabs are a) the Editing Tabs and the Presentation Tabs. Editing Tabs are used for actions like managing the factors of processes, declaring the flows as reference flows for the calculation of the system, managing stock definitions for junctions and input-output nodes and declaring process-wide as well as model-wide Parameters. Presentation Tabs are used for displaying calculation results to the user of the Tool. The same notation as earlier is valid also here with the Editing Tabs having an icon pencil and the Presentation Tabs having a table icon in order to easier distinguish their type.

Editing Tabs

The Editing Tabs contains the Specification Tab, the Process Parameters Tab and the System Parameters Tab. Depending on the model entity Specification Tab includes may have different options. Following is a description of the different options that can be found on the Specifications Tab as well as the Process Parameters and System Parameters Tab.

Specification Tab – Process

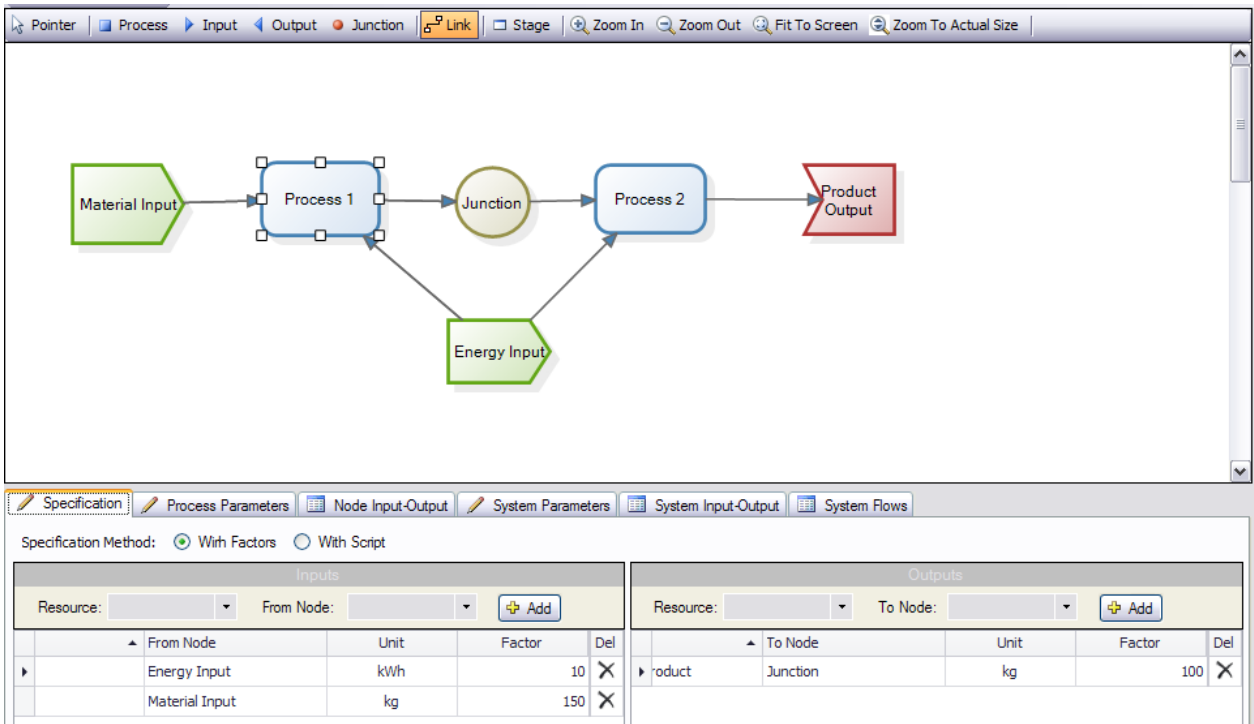


Figure 24: Specification Tab for Process

The Specification tab is the area where the flows are specified. Its features vary according to the selected node. Figure 24 shows the layout of the Specification tab of a Process node selected in the diagram area.

The left and the right parts of the Specification tab comprise a list of incoming and outgoing flows respectively. To add a flow, select a Resource from the drop-down list and the corresponding node from the From Node drop-down list, which follows on the other side of the flow. When Resource is selected Unit field is automatically filled according to the data previously entered in the Model Explorer. The column Factor is by default 0, but the user can enter there a factor keeping in mind always that flows are relative. To successfully calculate the flows of a model, it is obligatory to specify correctly the factors between relative incoming and outgoing flows of each process. For example, as seen in the previous figure to produce 100 kg of intermediate product, 10 kWh of energy and 150 kg of raw material are required. Exploring the Specifications Tab of the process, a verbal description of the process would be: “In Process 1, every **10** kWh of energy and **150** Kg of raw material are **transformed** into **100** Kg of intermediate product”.

Specification Tab – Link

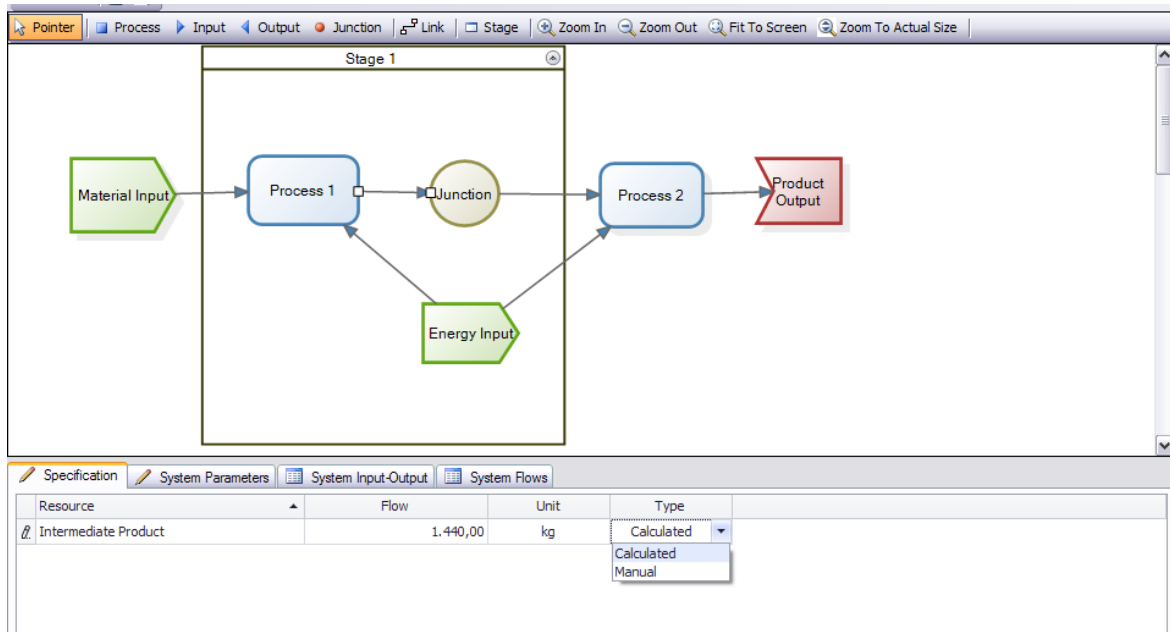


Figure 25: Specification Tab for Link

Figure 25 shows the layout of the Specification tab for a Link. It displays the number of flows that exist in the selected link and is also used to set reference flows in the calculation by declaring them in column Type as manual and thus enabling the manual entry of this value.

Specification Tab – Junction, Input / Output Node

When a *junction*, an *input* or an *output* node is selected, the Specification tab looks like the one in Figure 26. Through the Specification Tab an initial stock can be declared, which refers to the pre-existing quantity of a resource. The final stock is the quantity of the stock after this has been calculated by the system. There is also the option to enable the use of own resources (stock), if required, during the calculation of the model. The Consume Stock Column has 4 alternatives: Never, Always, On Supply, On Demand. Always equals to the stock consumption being consumed in priority to any other input/ output of the set resource. Likewise, the Never option implies that the stock is not consumed during the calculation of the model. The terms On Demand and On Supply are as per the economic terms. When On Demand option is selected then the stock is consumed as set in the demand part of the model. When On-Supply option is selected then the stock is consumed as set in the supply part of the model.

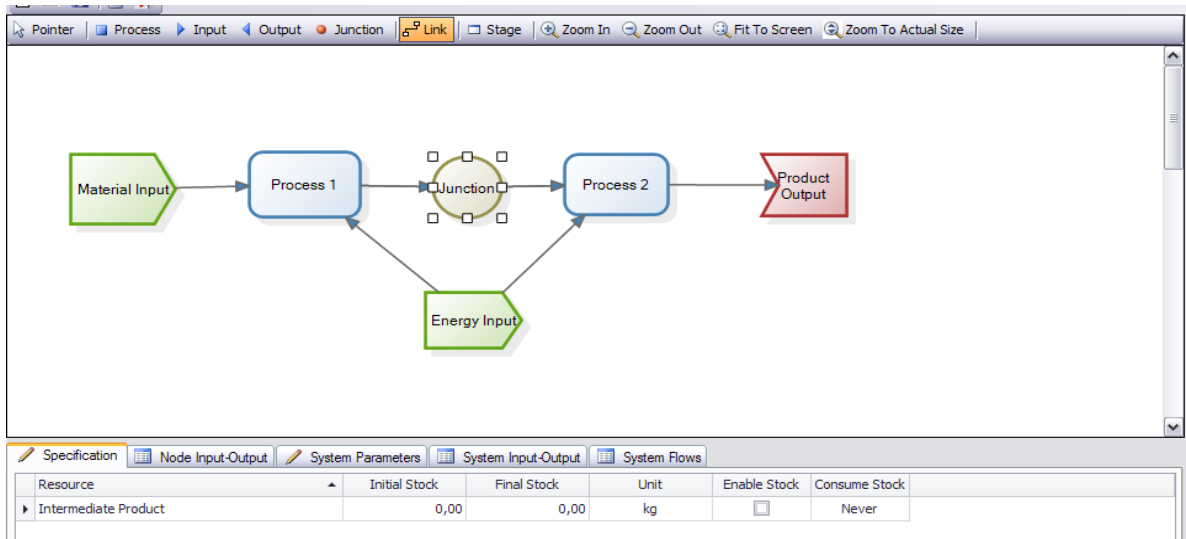


Figure 26: Specification Tab for Junction, Input / Output Node

Specification Tab – Process / System Parameters

PSM Tool provides the opportunity to set Parameters for specific Processes or for the whole Model. In order to create a Process wide Parameter, click on a Process, and select the Process Parameter Tab. Figure 27 shows the layout of the “Process Parameters” tab with a process node selected in the Model Editor. Enter a name at the Symbol area and then click on the Add button. You can then specify the Unit, Value and Description of the Parameter you have just created. The scope of Process Parameters is limited within a process. For example, in the following figure, in which Process 1 is selected, if the user creates a parameter that will be valid specifically for Process 1 and only that. In case the user wants to create a system-wide parameter, then the System Parameter Tab can be used similarly to the Process Parameter Tab, but the created parameters will have a global effect on the whole model.

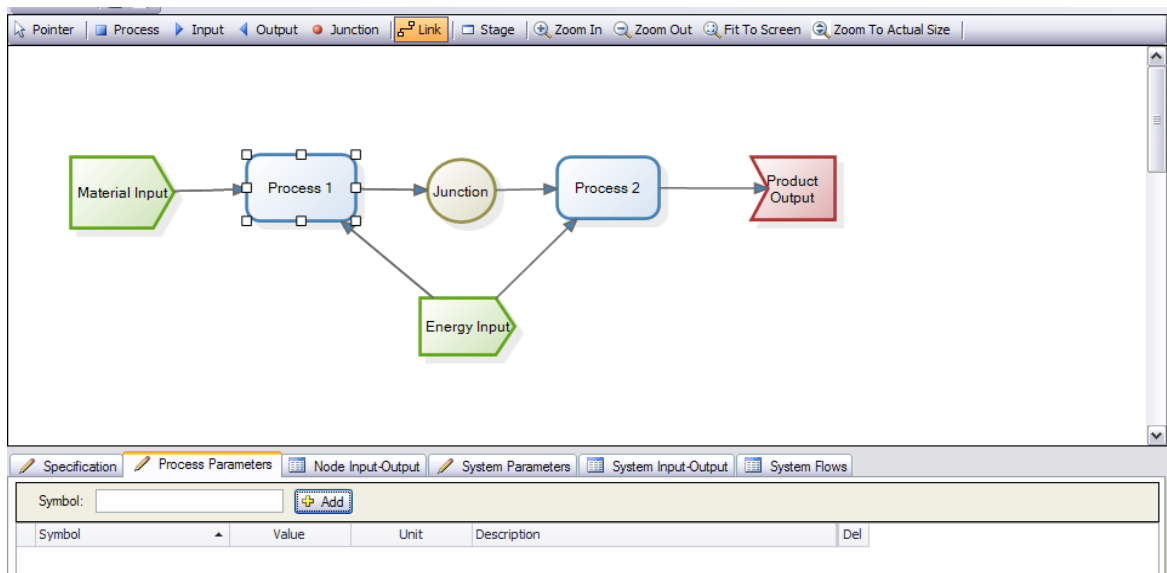


Figure 27: Process Parameters Tab

Presentation Tabs

Presentation tabs include the Node Input-Output Tab, the Stage Input-Output Tab, the System Input-Output Tab and the System Flows Tab. The Input-Output tabs are used to view calculation results for a selected process, as shown in the following figures. More specifically, the Node Input-Output tab shows the incoming and outgoing flows of the process, the node they are connected to as well as their resource. Stage Input-output and System Input-Output display the flows that exist in a selected *Stage* or a *Model* respectively. The System Flows tab displays all the flows of the model. Each column can be sorted either alphabetically or numerically in ascending or descending order by clicking on the specific header column, similar to the File Explorer in modern Operating Systems. Results can also be grouped by column by dragging a column header in the area above it. Figure 28 displays the Node Input-Output Tab. Two tables are available, the Inputs on the left and the Outputs on the right. Similarly, in Figure 29 and Figure 30 the Inputs and Outputs of a selected stage and the whole system are shown respectively.

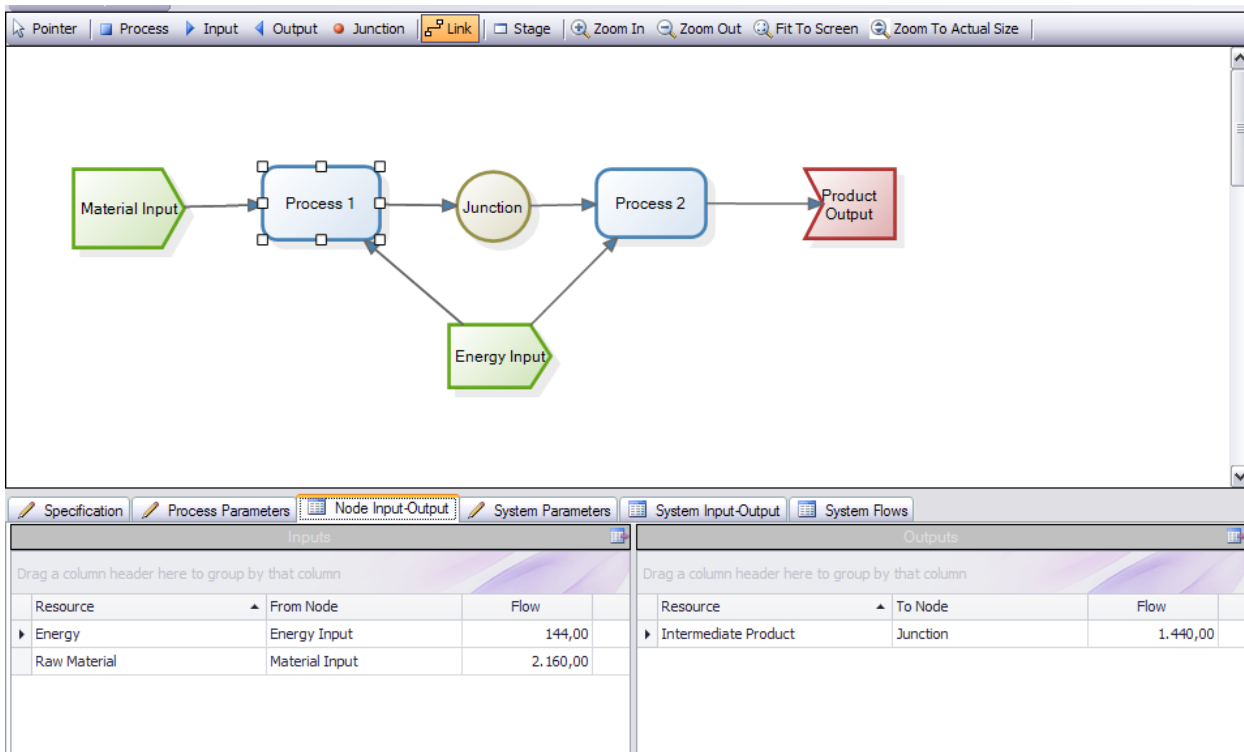


Figure 28: The Node Input-Output Tab

D4.3 Systemic Cognitive Models Prototypes

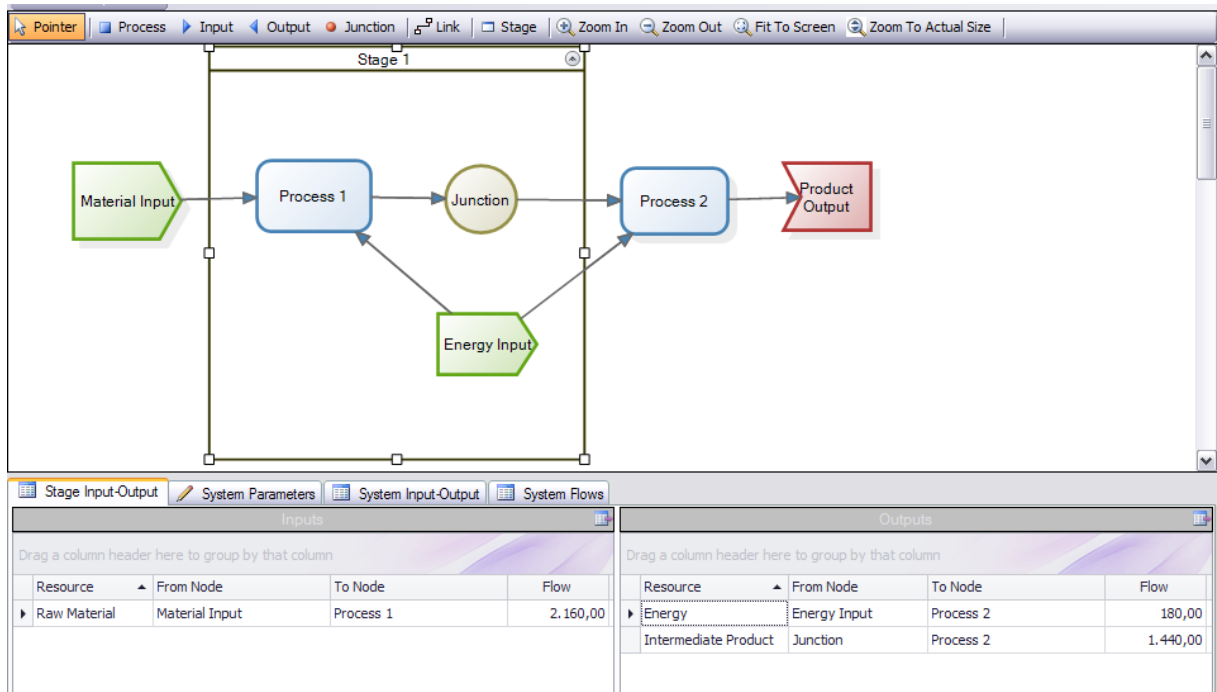


Figure 29: The Stage Input-Output Tab

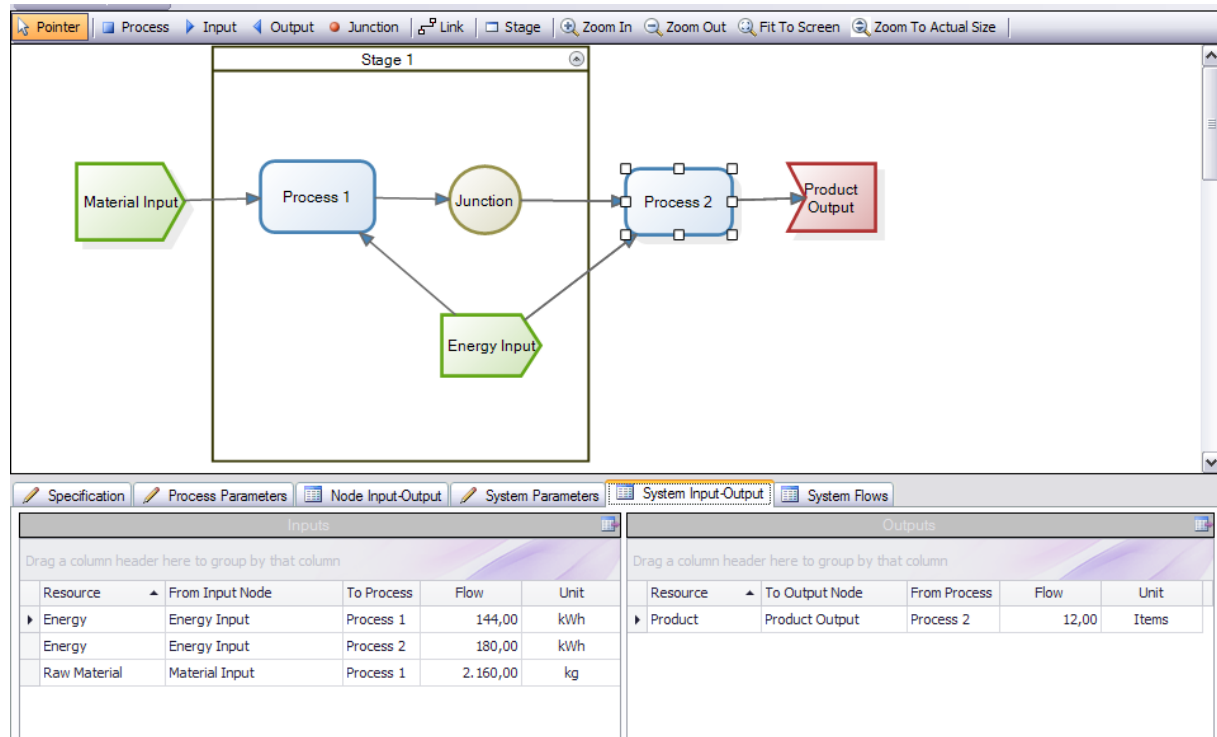


Figure 30: The System Input-Output Tab

3.3 PSM Online Service (HTTP API)

The PSM Online Service has been developed to support the integration requirements of FACTLOG project but at the same time to future-proof and enable further exploitation of the Tool in similar research and innovation actions. It allows any external actor (end-user, integrator, optimisation, analytics, etc) to create instances in complete isolation of the model, modify process parameters and perform simulations and what-if scenarios easily. The PSM Online Service relieves the end-user from the requirement of understanding the model building procedure and the functionalities of PSM Tool.

The actual value of the PSM API is illustrated by presenting and explaining the life-cycle of a model. During FACTLOG, as already mentioned, the two main categories of process industries studied are the continuous and the discrete process industries. Due to the differences in the fundamental principles of operation, not only the models have been developed differently but also the APIs of the Online Service are distinct.

Regarding the discrete cases (BRC and CONTINENTAL pilots), the initial stage is to create the entities and the interactions among them and define the complexities and the dynamic parameters. In order to perform this step static and dynamic data from the pilot are required. In order to perform a simulation, an isolated model of the system is created, which allows the user the flexibility to change parameter values and update the input data files. With the help of the optimisation results (as in discrete case the simulation is run after optimisation service produces the schedule), any number of scenarios can be simulated and the required KPIs can be calculated. It is very easy to change any parameter and re-run the simulation on the same isolated model to compare KPIs with different set-ups or perform what-if hypothesis tests. Finally, if the input data change a mechanism to update them is in place as well as the required KPIs. When the need for simulation is done, the model instance can be removed from the system until a new version is required. In Figure 31 the described life-cycle is summarised in a graph.

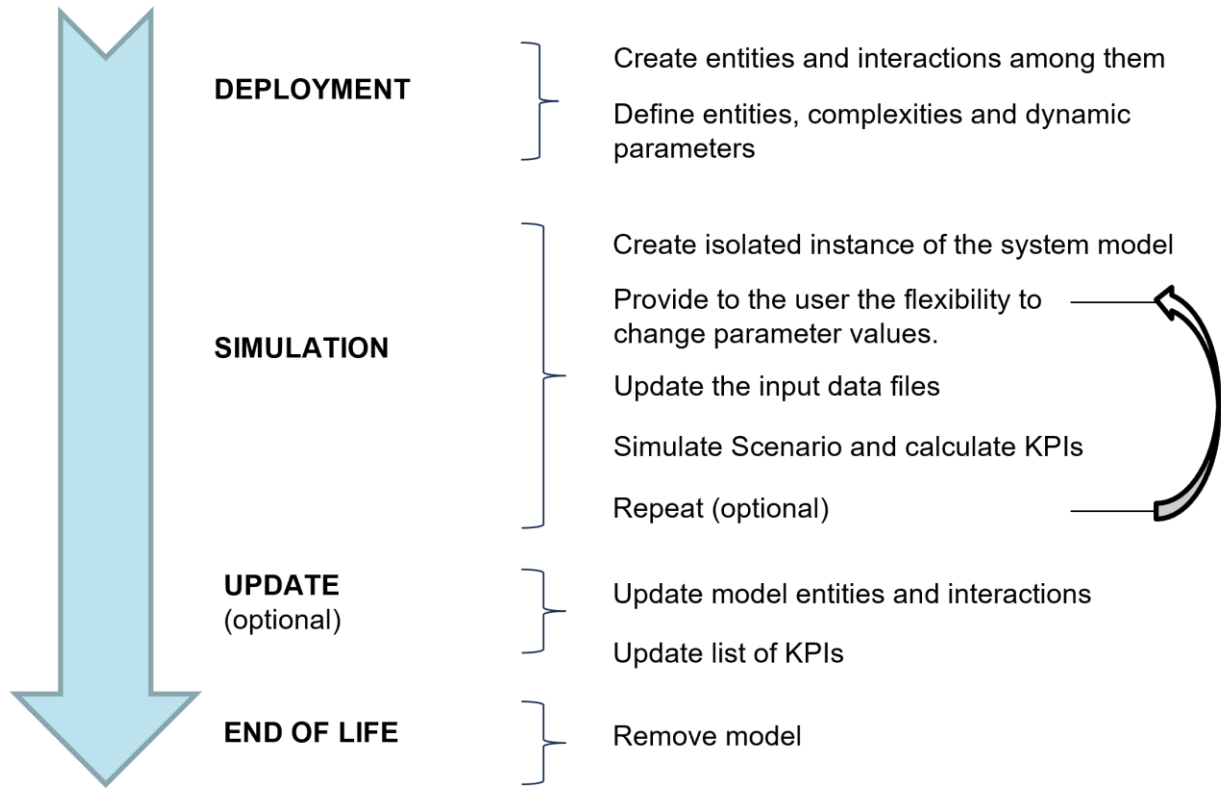


Figure 31: Discrete systems API Lifecycle

The API for the Discrete cases can be accessed from the following URL <https://www.indigo.tuc.gr:8443/DPSM/swagger/index.html> . There two different POST calls can be seen, one named *RunProductionLineScenario*, which corresponds to the CONTINENTAL pilot and another one named *RunParallelLineScenario* which corresponds to the BRC pilot, which can be seen in Figure 32.

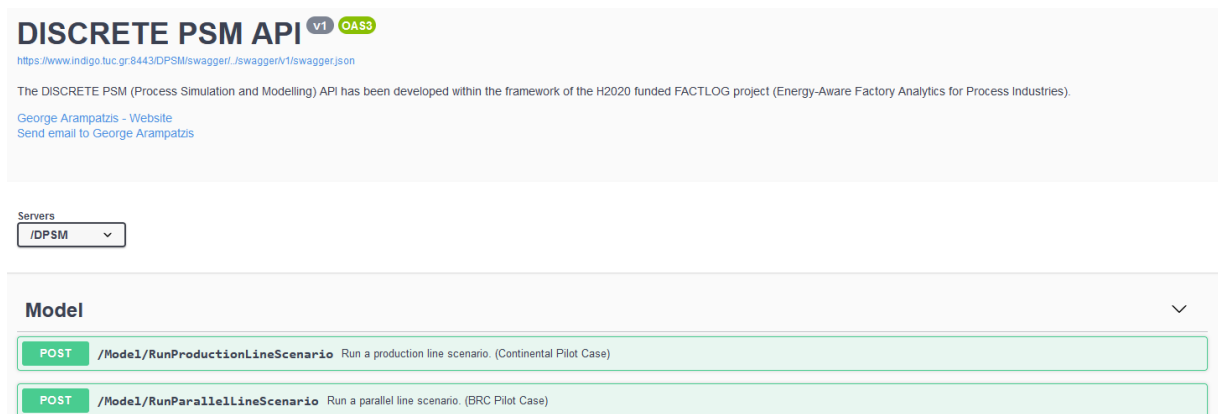


Figure 32: Discrete PSM API

The Discrete PSM HTTP API requires a *.json* input for both BRC and CONTINENTAL pilot cases, where data coming from the pilot case as well as the optimisation results are compiled into a single file. The generic format can be seen in the following *.json* snip while an example of a full *.json* input to the service is presented in Appendix II – BRC Simulation

Request for BRC and Appendix III – CONTINENTAL Simulation Request for CONTINENTAL.

```
{  
  "scheduleParam": [  
    [  
      "string"  
    ]  
  ],  
  "optimizationParam": [  
    [  
      "string"  
    ]  
  ]  
}
```

Figure 33: Discrete API input .json format

Generally speaking, the Continuous Processes HTTP API shares a lot of similarities with the above described but also differences, that's why they are presented separately. Regarding the continuous processes, the model needs to be created with the PSM Tool. Then, it is registered through the Tool to the platform providing the exact representation of the model as produced by the standalone app. Next step is to grab an instance at the specific time frame important for the specification assessment or optimisation effort. The HTTP API provides the flexibility to set parameter values that dynamically affect the process. When the values are set we are able to simulate the results either for the whole model or per unit (depending if we are looking for a specific unit or the whole model values) and then get those calculated values. With these results we are able to calculate KPIs or objective functions and of course depending if the results are satisfactory or not we can repeat the calculations altering the parameter values. When we are done with our calculations regarding the specific instance we can discard it. In case the registered model needs to be updated this functionality is provided too, however that is not going to update any instances already created before the update but only the model itself. Finally, when we have completed every simulation and experimentation and we don't require the model any more we can completely remove it from the system, marking the end of the life-cycle we have been discussing. In Figure 34 this life-cycle described above is being depicted.

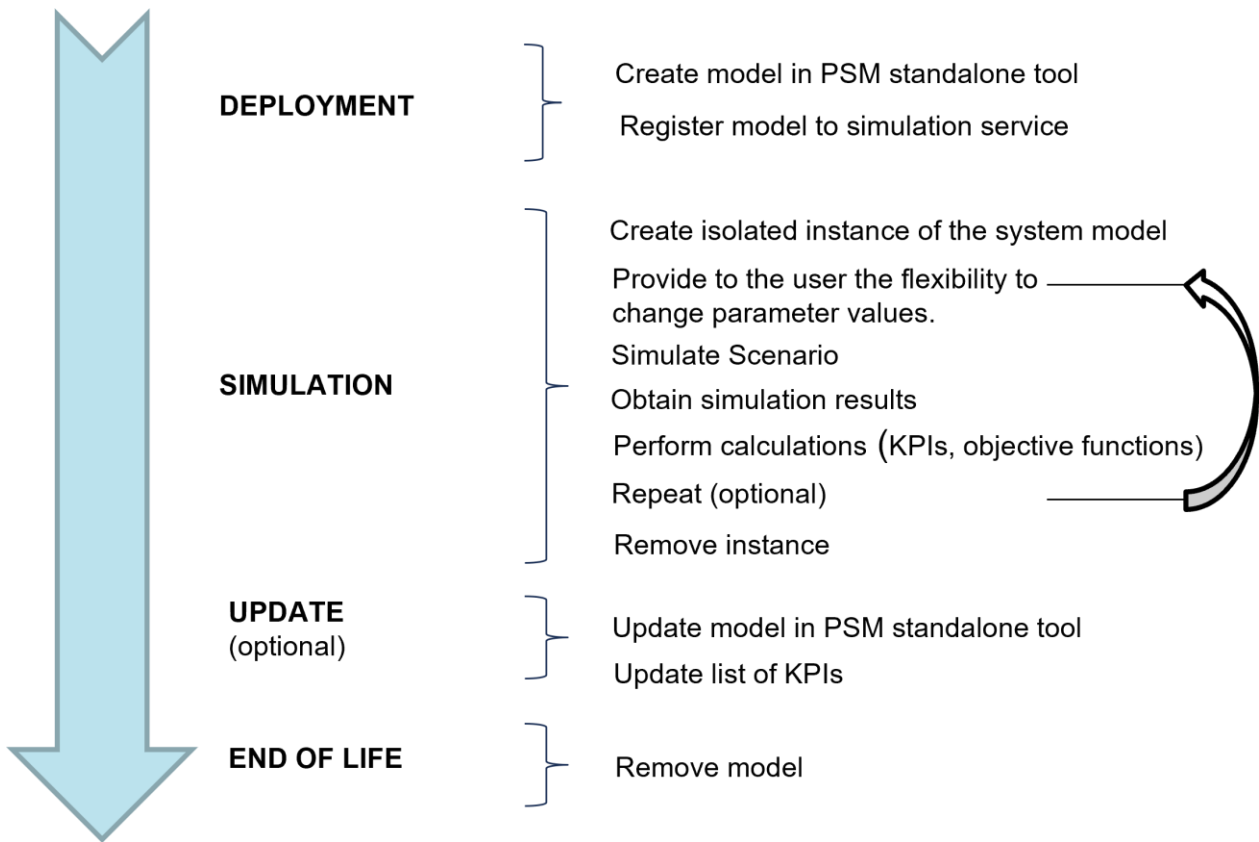


Figure 34: Continuous systems API Lifecycle

The HTTP API for the continuous cases can be accessed from the following URL <https://www.indigo.tuc.gr:8443/PSMApi/swagger/index.html>. This API is considerably more complex than the Discrete HTTP API, as it is not built only for serving the project needs but the PSM Tool in total, covering every functionality available. There are four main categories (seen in Figure 35) of calls; the calls regarding the Instance, the calls regarding the Model, the calls regarding the integration with the Machine Learning Models of JSI developed in T4.4 and the calls related to the creation of the operational scenarios needed for Optimisation.

PSM API v1 OAS3

<https://www.indigo.tuc.gr:8443/PSMApi/swagger/.swagger/v1/swagger.json>

The PSM (Process Simulation and Modelling) API has been developed within the framework of the H2020 funded FACTLOG project (Energy-Aware Factory Analytics for Process Industries).

George Arampatzis - Website
Send email to George Arampatzis

Servers

/PSMApi ▾

Instance ▾

MachineLearningModels ▾

Model ▾

Scenario ▾

Figure 35: PSM API Calls

As it will be too extensive to describe every call here, in Appendix I – Continuous Process Simulation and Modelling API a table describing in detail every call and its functionality can be seen. Indicatively, in Figure 36 the many different calls regarding the manipulation of a model can be seen.

Model ^

GET	/api/model	Get a list with stored models.	▾
GET	/api/model/{id}	Get model specification by model unique identifier.	▾
DELETE	/api/model/{id}	Remove a model specification (model name and definition)	▾
GET	/api/model/definition	Given a model id returns the model definition only.	▾
PUT	/api/model/file/{id}	Upload file and update an existing model.	▾
POST	/api/model/file	Upload file and add a new model.	▾
GET	/api/model/model-parameters	Get all model parameters by specifying an model unique identifier.	▾
GET	/api/model/model-parameter	Get a model parameter by specifying an model unique identifier and parameter name.	▾
PUT	/api/model/model-parameter	Set a specific model parameter value by specifying model unique identifier and parameter name.	▾
GET	/api/model/process-parameters	Get all process parameters by specifying a model model unique identifier, and the process. You must specify either the process ID or NAME. Not both of them.	▾
GET	/api/model/process-parameter	Get a specific process parameter by specifying its name: 1) Model unique identifier, 2) Process unique identifier or process name parameter, 3) Process parameter name. You must specify either the process ID or NAME. Not both of them.	▾
PUT	/api/model/process-parameter	Set a specific process parameter value by specifying: 1) Model unique identifier (Required), 2) Process unique identifier or process name parameter, 3) Process parameter symbol (name), 4) New process parameter value (Required).	▾
GET	/api/model/flow/ByID	Gets the requested flow utilizing the provided IDs by specifying: 1) Model unique identifier, 2) Flow source node ID, 3) Flow target node ID, 4) Resource ID.	▾
PUT	/api/model/flow/ByID	Update the requested flow quantity utilizing entities IDs, by specifying: 1) Model unique identifier, 2) Flow source node ID, 3) Flow target node ID, 4) Resource ID, 5) New flow quantity.	▾
GET	/api/model/flow/ByName	Gets the requested flow utilizing entity names, by specifying: 1) Model unique identifier, 2) Flow source node name, 3) Flow target node name, 4) Resource name.	▾
PUT	/api/model/flow/ByName	Update the requested flow quantity utilizing entities names, by specifying: 1) Model unique identifier, 2) Flow source node name, 3) Flow target node name, 4) Resource name, 5) New flow quantity.	▾

Figure 36: PSM API calls regarding Model

4 Implementation per Pilot Case

4.1 BRC

4.1.1 Case Description

A bay of a steel reinforcing installation in the premises of BRC Reinforcement in Newport, UK is considered as a case study. The processes performed involve cutting and shaping various diameters of steel reinforcing bar using various manual or automatic operations. BRC produces parts of a variety of diameters in the form of simple straight bars, “U” shaped bars up to complicated shape codes of 3D shapes with different numbers of bents. Three types (or families) of final products can be produced: coils, bars and bent bars of different diameters and with different numbers of bents. Coils and bars are produced after a one stage process while bent bars need a 2-stage process (first cutting and then bending – there is no flexibility between the 2 stages). For the previously described processes, four machines are available (in parallel) for coil cutting ($M_1 - M_4$), 2 for Bar Cutting ($M_5 - M_6$) and 3 machines ($M_7 - M_9$) for bar bending. Machines performing a type of process are not all identical per stage as they have different constraints (e.g., maximum size of raw materials that they can process) and production characteristics (e.g. process, speeds). BRC receives from its customers’ orders composed of multiple jobs, each one referring to the production of a specific quantity of a given type, diameter and shape of product (product specifications).

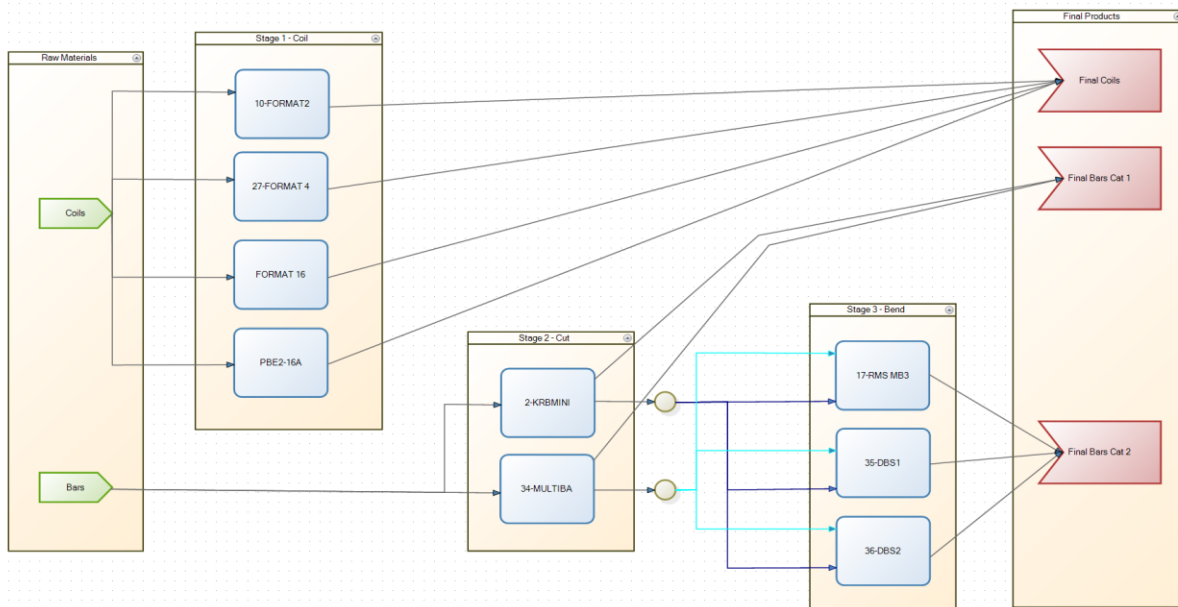


Figure 37: BRC Bay 3 Shopfloor structure

Each machine performs strictly processes of one stage and setup is necessary before changing the operation type performed in it. Moving equipment (cranes) is used for transportation loading and unloading of parts and products between machines and buffers but the respective delays are considered negligible. The possible flows of the described process are presented in Figure 37 as a 3-stage process where stages 1 and 2 are completely independent, while parts receiving bending process (stage 3) must be already cut to the given dimensions (stage 2). Different types of raw materials are initially available and with respect to specifications of products characteristics concerning, size, geometry, weight, type of raw material used etc. can be produced. It is obvious that the production

system considered refers to parallel machines and for this case schedule per machine is necessary in order to model it's behaviour.

The main entities considered are the machines of the three production stages according to the specifications provided. In order to model the behaviour of the machines, the jobs under process must be considered. The orders consist of jobs that refer to batches of products with similar characteristics. The main assumptions regarding process modelling and simulation concern the orders that have to be scheduled, as well as the characteristics (batch sizes, processing times in the alternative machines per stage, setup times etc.) of all jobs. When a job is scheduled, no preemption - interruption is allowed and no job can be processed by more than one machine at the same time (i.e., job splitting is not allowed). In addition, machines cannot process multiple jobs at the same time and if a product is flagged as finished, then it cannot be processed again. Finally, raw material availability in all cases is considered infinite.

4.1.2 Process Modelling and Simulation

As already presented in Chapter 2 a hierarchical modelling approach is followed in the pilots modelled as Discrete cases. According to this the first level of the model, that represents the entities of the system is constructed using Timed Petri nets and is mainly used as a graphical representation. In the case of BRC the main entities considered up to now are the nine machines that have already been described in Figure 37. The implemented Petri net model is shown in Figure 38. In this, sets of machines of each production stage of the system as well as the raw material and final product buffers are graphically grouped together using boxes containing them (this is for demonstration purposes only and does not play any role in the execution of the PN).

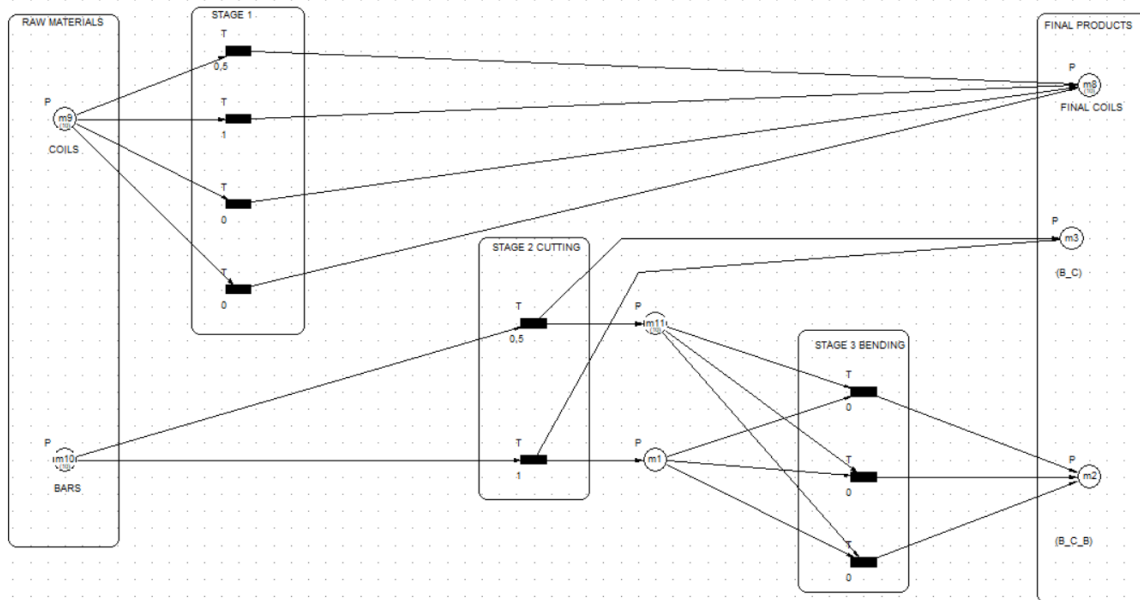


Figure 38: Petri net model of the upper layer of BRC Shopfloor

In the model of Figure 4.2 the places represent the buffers of the systems where raw materials, in process parts and final products are stored (and are of limited capacity) while each transition refers to a machine of the system. From this figure it is obvious that coils receive one cut process in order to become final products while all bars receive a first cutting process while certain of them receive a second bending process according to the customers'

needs. It must be noted that it is not possible raw materials to receive only bending process according to the specifications provided from the pilot industry.

The second layer of the model, that is mainly operated during simulation is implemented using Python programming language and does not have a graphical representation. This happens because the respective layer is non-static and has to be implemented with respect to the number of orders and jobs considered as well as to the number of jobs assigned to each one of the machines. The implementation of such a model with traditional methods like Petri nets would be difficult and non-efficient, while its complexity would be high, and the simulation durations significantly increased compared to discrete event simulation performed using Python. In addition, Python has been used from optimisation service also so the communication between the 2 services and exchange of outputs is much more efficient and direct.

In order to illustrate the advanced complexity of the models implemented using a graphical modelling tool such as Petri nets, Figure 39 is presented. This describes a scenario of BRC Pilot according to which four jobs that refer to coils and four jobs that refer to bars (cutting and bending) are performed.

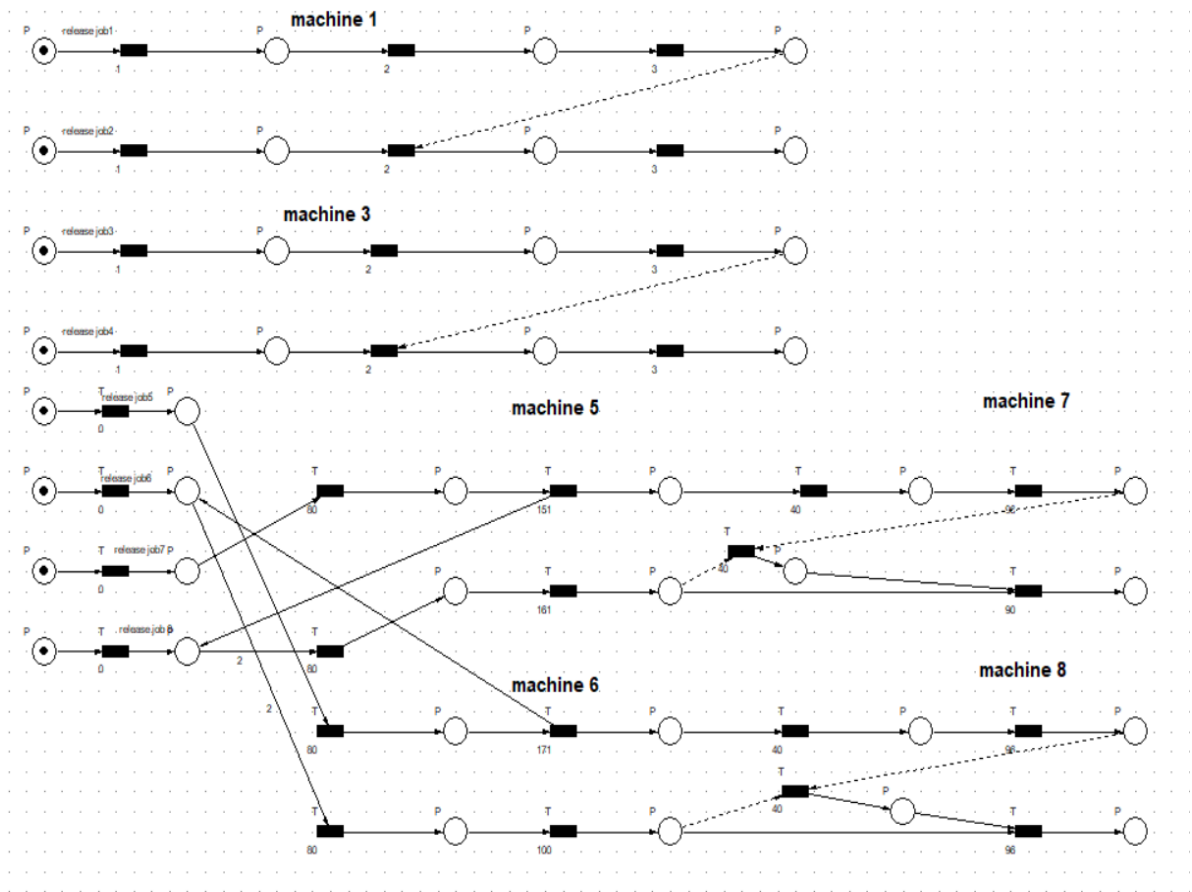


Figure 39: Petri net model of a specific scenario in BRC Pilot

After the calculation of the schedule from optimisation service, the respective subsets of the available machines are defined, and the schedule of each one is transformed in the respective Petri net. The schedule in fact defines the sequence according to which the jobs have to be performed in each machine following the constraints that have to do with machine

availability, raw materials availability and job assignment. The overall node complexity of such a simple case is 42 places and 30 transitions. When in a realistic day in BRC some hundreds of jobs are performed this means that the overall complexity of a Petri net model would be very high while the graphical representation would also be not very helpful especially for most of the users. Every alteration of the schedule would lead to changes of the Petri net models of the entities (machines) used for its process. Only changes in the values of parameters (e.g., processing durations) does not lead to changes in the structure of the implemented Petri net.

The discrete event simulation performed using Python is based in the already described inputs from BRC and associate services. The static data referring to machines of the system are not updated very often but this is also possible as the implemented model is fully parametric. After reading the entities (machines in this case) and their grouping (if they belong to stage 1, 2 or 3) optimisation service output is used to introduce the jobs assigned to each machine as well as the sequence in which jobs are operated. Then the processing and setup times of these jobs are introduced from static data and simulation is ready to start as all the values of the parameters have been defined. A general assumption that is followed is that machine setup for the upcoming (according to machine schedule job) can be performed when a machine becomes idle from it's previous process. For machines of the first two stages (Cut coils and Cut bars) completion of machine setup enables the process of the parts in the respective machine. In the case of machines of stage three, a process can be performed when setup has finished and process of the parts in the previous stage (cut bars) has also finished. So, in this case the maximum of the two respective timestamps is used as initialization of the process in the respective bend machine. When the performance of all jobs in the defined machines has finished, simulation is terminated. The calculation of the already described KPIs is possible using appropriate variables during simulation. Simulation time is calculated as the maximum completion time of all orders. Percentage of machine usage is calculated by adding the setup and process durations of the jobs performed in a machine and by dividing this duration with the simulation time and is used to detect possible bottlenecks (in unbalanced networks of machines) as well as to schedule machine maintenance activities. Finally, completion time per order, that is a very important KPI since it makes possible the distribution of the order in the customer, is calculated as the maximum of the completion times of the jobs comprising an order. Optimisation service calculates extra KPIs (such as machine and order Gantt charts) while the addition of extra KPIs is currently a matter of discussion with BRC.

4.1.3 Results and Outcomes

In this section the results derived from the simulation of a scenario that consists of 43 orders and 300 jobs (each order is composed from 1 – 191 jobs) are presented. Initially BRC static and dynamic data as well as Optimisation service output are introduced in the Process modelling and Simulation service using *.json* files.

Table 2 summarises the aforementioned inputs for the considered scenario. These jobs are assigned from optimisation service in a subset of five different machines (two Bar Cut and three Bar Bend). It must be noted that all 300 jobs are processed in Stage 2 (Cut Bars) and 179 are processed in Stage 3 (Bend Bars). From Table 2 it can be validated that the overall number of jobs processed in the set of five machines is equal to the sum of Stage 2 and Stage 3 jobs.

BRC Simulation Scenario	
Production Orders	43
Jobs	300
Stage 1 Jobs (Cut Coil)	0
Stage 2 Jobs (Cut Bar)	300
Stage 3 Jobs (Bend Bar)	179
Schedule	
Machine	Number of assigned Jobs
35-DBS1	74
36-DBS2	39
2-KRBMINI	123
17-RMS MB3	66
34-MULTIBA	177

Table 2: BRC pilot input description

The format of the process modelling and simulation output *.json* file is presented in Figure 40, while the whole output *.json* file can be found in Appendix IV – BRC Simulation Result.

For these inputs the simulation is terminated after 1881.83-time units, duration exactly equal to the one calculated from optimisation service. The comparison between durations calculated from optimisation service and the one calculated from Process Modelling and Simulation is used in certain cases to validate the calculated results as well as the accuracy of the implemented model. However, Optimisation Service or other users of FACTLOG platform (e.g., users from pilot or from other associate services), may change the values of certain model parameters and repeat the simulation in order to evaluate different possible schedules. This can be used even as part of a heuristic optimisation method which produces alternative schedules in order to calculate an optimal or suboptimal solution (a survey on such algorithms is presented in [15]). According to the calculated results machine usage varies from 51.07 % to 99.12 %. Low machine usage practically means extended machine idleness and loss of production capacity, while high machine utilization (as for machine 35-DBS1 in this example that reaches almost 100%) practically means increased probability of bottleneck in the respective machine in case of random events appearance (such as machine breakdowns). This may lead to tardiness regarding order completion and increase of the overall production time (according to [16] typically mean machine utilization varies from 60-90% for different types of machines), so the study of alternative schedules when machine usage reaches very high rates should be a matter of evaluation. Finally order

completion time is the maximum of the completion times of the jobs composing an order and refers to the time when this order can be transferred to the customer.

```
{
  "simulationTime":
  [
    {
      "time": "1881.83"
    }
  ],
  "machinesUsed":
  [
    {
      "machineName": "2-KRBMINI",
      "usageTime": "961.13",
      "percentageOfSimTime": "51.07"
    },
    ...
    ...
    The rest has been removed due to space limitations.
    ...
    ...
  ],
  "completionTimePerOrder":
  [
    {
      "order": "C202011389SEQ-0585",
      "completionTime": "388.7"
    },
    {
      "order": "C201910830SEQ-0063",
      "completionTime": "122.61"
    },
    ...
    ...
    The rest has been removed due to space limitations.
    ...
    ...
  ]
}
```

Figure 40: Structure of the Process Modelling and Simulation output .json file

4.2 CONTINENTAL

4.2.1 Case Description

CONTINENTAL is among the top worldwide electronic manufacturers, whose products are manufactured in electronic plants such as the plant in Timisoara, a part of which is considered in FACTLOG project. The specific plant is producing electronic products and covers all stages from design to production. CONTINENTAL's customers may define specifications of the product according to their individual needs from design phase.

The production procedure considered in FACTLOG Project is depicted in Figure 41. Such a production setting can be modelled as a multi-stage flow shop scheduling problem (FSSP) with resource constraints (e.g., semi-finished products, raw materials etc.). In particular, two consecutive production lines with a buffer space between them are considered. The first line

is called preassembly line and is composed of five machines that perform different process (one process is performed in each machine). The second production line is composed of 18 machines (again with one dedicated machine for each process) and is called final assembly process. The names and types of the processes performed in each machine are shown in Figure 41.

The main assumptions regarding CONTINENTAL case are the following: all jobs follow the same routing through each line (there is no routing flexibility, but job sequence can change between two lines) and are processed in all machines of a line, no internal buffers are considered between workplaces except the one between the two lines, no parallel workplaces exist in any production stage (all stages are single server stages) and the processes performed are highly automated. Change of process performed in a line is followed by setup of all machines of the line, the duration of parts movement between workplaces is considered negligible and orders refer to batches of products with similar characteristics.

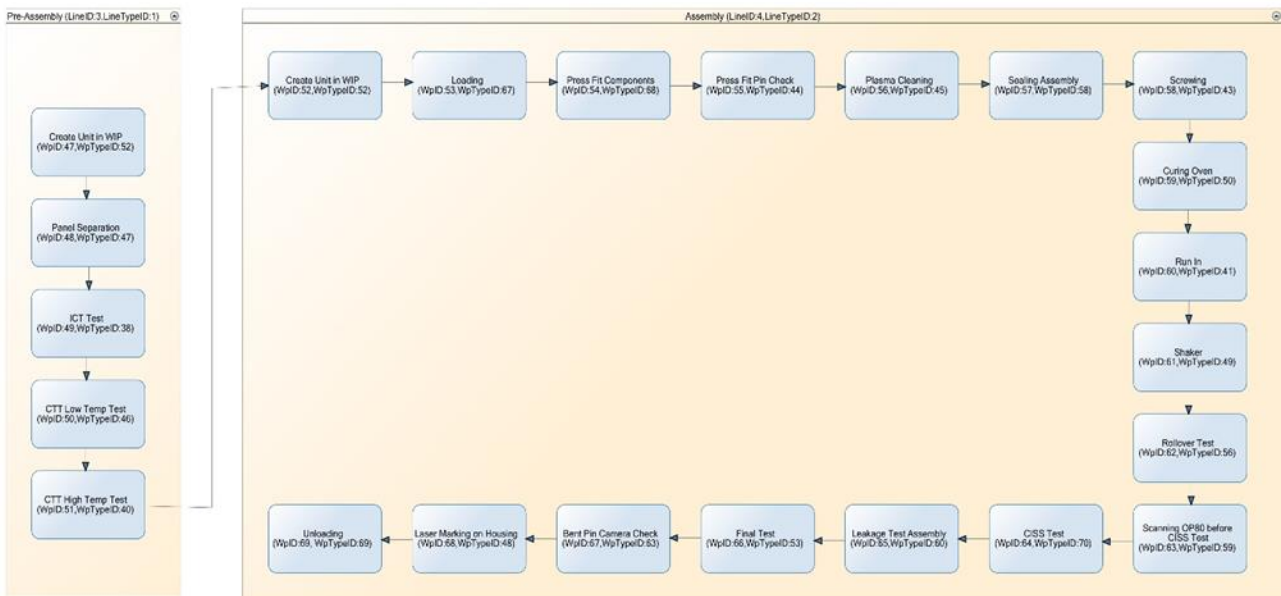


Figure 41: CONTINENTAL Timisoara plant shop floor structure

The main entities considered are the machines of the two lines according to the specifications provided and the buffer space between them. When a job is scheduled, no preemption - interruption is allowed, machines cannot process multiple jobs at the same time and if a product is flagged as finished, it cannot be processed again. Raw material availability in all cases is considered infinite, products belong to product families, machine setup is considered sequence independent (but has common duration of all products of a product family) and ideal processing durations for all types of products in all machines (since no parallel machines exist) are provided by CONTINENTAL. The structure of CONTINENTAL shop floor has common characteristics with the one of BRC but has also significant differences. In particular, since all processes are performed in a single machine stage, schedules provided from optimisation service do not need to be calculated per machine but per line (and are common for all machines of a line).

4.2.2 Process Modelling and Simulation

In the case of CONTINENTAL pilot a hierarchical modelling approach is followed with common characteristics as the one already presented in BRC. According to this the first level of the model, that is static since entities do not change often, is constructed using Timed Petri Nets and is mainly used as a graphical representation. The main entities considered are the 23 (5 + 18) machines illustrated in Figure 41. The implemented Petri Net model is shown in Figure 42. In this, two black boxes have been used to illustrate the limits of the consecutive production lines. The complexity of this model is increased since each part is processed in 23 machines; it is composed from 70 places and 69 transitions even in the upper layer (the respective upper layer from the PN model of BRC was composed from seven places and nine transitions in comparison). The addition of realistic scale dynamic data would make the PN model completely inefficient so for this reason Python programming language has been used to model and simulate the next layer of CONTINENTAL industrial system.

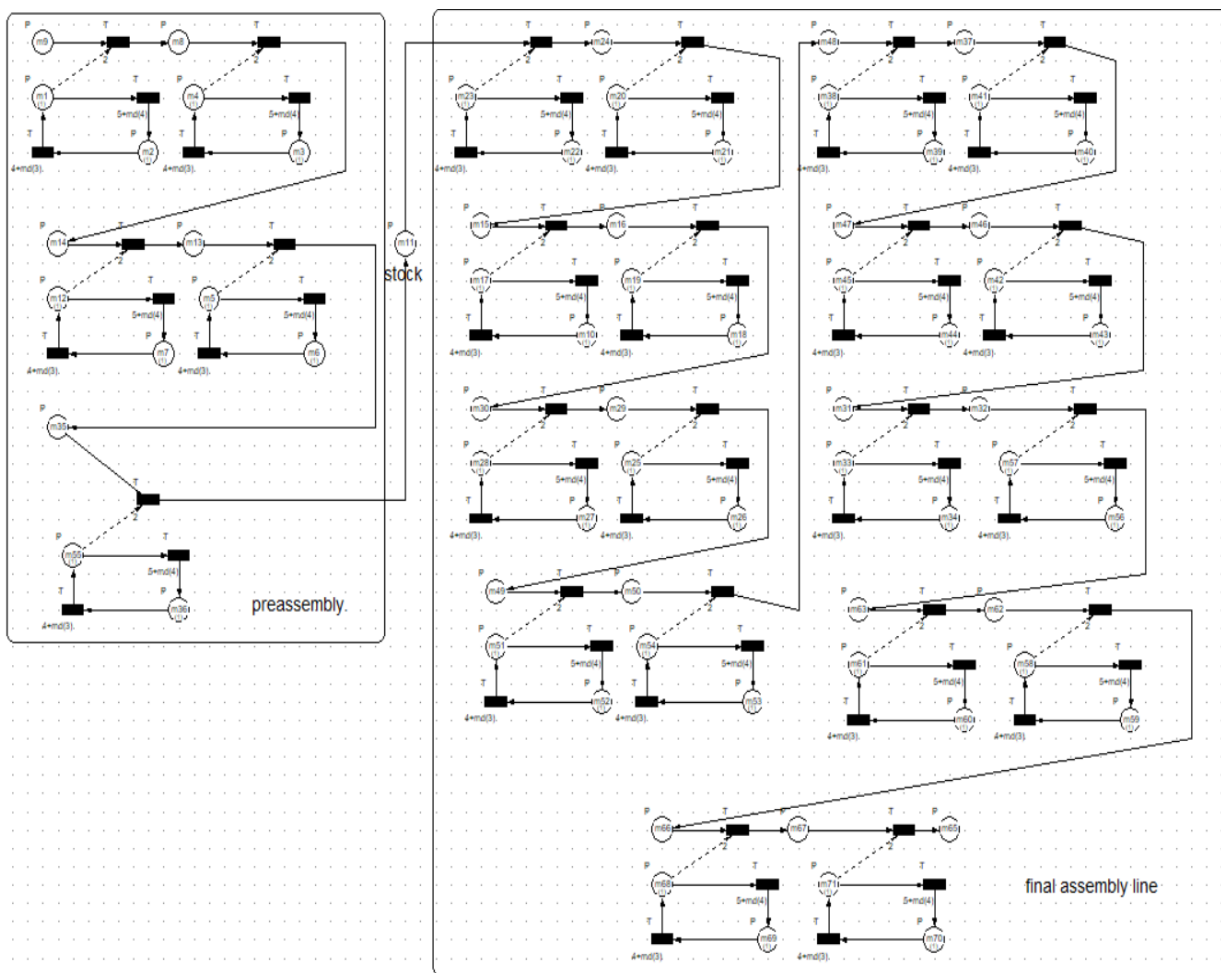


Figure 42: Petri net model of the upper layer of CONTINENTAL Pilot Shopfloor.

After introducing the entities (18+5 machines in this case) optimisation service output is used to read the jobs sequence in each production line. Then the processing and setup times of these jobs are introduced from static data and simulation is ready to start as all the values of the parameters have been defined. A general assumption that is followed is that machine setup for the upcoming (according to machine schedule job) can be performed when a

machine becomes idle from its previous process. The process of an order can start in a machine if this machine has finished processing the previous order according to the schedule, the required setup has taken place and only if the respective raw materials are available after they have been processed from the previous machine (since the model is sequential) or from the previous line if the machine considered is the first machine of final assembly.

So, in this case the maximum of the three described timestamps is used as initialization of the process in the respective machine. When the performance of all jobs in the defined machines is finished, simulation is terminated. The calculation of the already described KPIs is possible using appropriate variables during simulation. Simulation time is calculated as the maximum completion time of all orders. Percentage of machine usage is calculated by adding the setup and process durations of the jobs performed in a machine and by dividing this duration with the simulation time and is used to detect possible bottlenecks (in unbalanced networks of machines) as well as to schedule machine maintenance activities. Optimisation service calculates additional KPIs (such as machine and order Gantt charts) while the addition of extra KPIs is currently a matter of discussion with CONTINENTAL.

4.2.3 Results

In this section the results that are derived from the simulation of a scenario that consists of ten orders (in this case each order consists of one job so the two terms can be used interchangeably) are presented. Initially CONTINENTAL static and dynamic data as well as optimisation service output (schedule per line) are introduced in the Process Modelling and Simulation service using *.json* files. In CONTINENTAL case all machines are used in any process, as shop floor has a sequential structure.

The format of the process modelling and simulation output *.json* file for the CONTINENTAL pilot case is presented in Figure 43, while the whole output *.json* file can be found in Appendix V – CONTINENTAL Simulation Result.

For those inputs, simulation is terminated after 263284-time units, duration exactly equal to the one calculated from optimisation service. If a schedule implementation start date is available, the end date can be calculated in order to compare the results with the orders due dates decided with customers. The results provided from optimisation service can be used in the same way as discussed in BRC case for model verification and alternative scenarios evaluation. Additionally, the percentage of machine usage and order completion time (maximum of the completion times of the jobs belonging to each order) are calculated.

```
{
  "simulationTime":
  [
    {
      "simEnd": "263284"
    },
    {
      "simDuration": "263284"
    }
  ],
  "machinesUsed":
  [
    {
      "workplaceName": "PRA_Create unit in WIP_1",
      "processingTime": "70778",
      "setupTime": "84",

```

```

    "totalTime": "70862",
    "percentageOfTotalTime": "26.91"
  },
  ...
  ...
  The rest has been removed due to space limitations.
  ...
  ...
],
"completionTimePerOrder":
[
  {
    "orderId": "1",
    "orderName": "order rllpwHHCGorXmK8XH03e",
    "completionTime": "199230"
  },
  {
    "orderId": "2",
    "orderName": "order yymynK1etnIPHhy8m4HN",
    "completionTime": "221806"
  },
  ...
  ...
  The rest has been removed due to space limitations.
  ...
  ...
]
}

```

Figure 43: Structure of the Process modelling and Simulation output .json file

Table 3 summarises the percentage of machine usage (for the time horizon of the current scenario) for each of the 23 machines of pre-assembly and final assembly line. From this we can see that the percentage of machine usage varies from ~12 % to ~33 % and none of the machines acts as a bottleneck in CONTINENTAL pilot case. Since the structure of the model is sequential, the existence of a bottleneck would reduce significantly the efficiency of the overall system.

CONTINENTAL Simulation Scenario	
Machine	% of machine usage
PRA_Create unit in WIP_1	26.91
PRA_PANEL SEPERATION_2	25.86
PRA_ICT test_4	21.87
PRA_CTT LOW TEMP TEST_6	17.83
PRA_CTT HIGH TEMP TEST_8	20.66
FA_Create unit in WIP_1	21.55
FA_Loading_2	14.89

CONTINENTAL Simulation Scenario	
FA_Press Fit componets_4	17.87
FA_Pressfit pin check_5	23.02
FA_Plasma Cleaning_7	15.98
FA_SEALING ASSEMBLY_8	19.72
FA_Screwing_10	20.89
FA_Curing oven_12	12.3
FA_Run in_13	23.38
FA_Shaker_15	21.23
FA_Rollover test_17	32.95
FA_Scanning OP80 before CISS test_19	26.78
FA_CISS Test_20	32.27
FA_Lekeage test assembly_22	23
FA_Final test_24	30.4
FA_Bent Pin Camera Check_26	19.24
FA_Laser marking on housing_28	31.22
FA_Unloading_30	21.09

Table 3: Percentage of machine usage

4.3 TUPRAS

4.3.1 Case Description

In FACTLOG project, and specifically on TUPRAS pilot, the case lies around the Izmit refinery. This plant has a combined capacity of 11.3 million tonnes per year and produces a number of petroleum products such as diesel, gasoline, naphtha and LPG. For the pilot case, the focus is the LPG purification process, a process that has to be undertaken in order to keep the final product free of impurities.

In order to understand what the impurities are, it should be mentioned here that Liquefied Petroleum Gas (LPG) is a mixture of hydrocarbon gases; mostly propane (C_3H_8) and butane (C_4H_{10}) with propylene, butylene, and various other hydrocarbons usually also present in smaller concentrations.

The main impurities that have to be removed from the product are C2, C5 and Sulphur. For that, TUPRAS refinery is equipped with Debutanizer / Deethanizer units which remove C2 / C5 and DEA units which remove Sulphur using diethylamine, an amine that absorbs the excessive sulphur content on the LPG mixture.

The most important problem of TUPRAS refinery, comes from the fact the final product is gathered in a common tank regardless of the process that has been produced. If a process is producing off-specs product it will be gathered in the final tank. Samples to test product quality are taken every two to three days and are analysed in the laboratory. This analysis takes around 24 hours and if the quality is off-spec an imminent need to recover the product arises. However, the origin of the off-spec product might be unknown or not able to be detected anymore because the process parameters have changed during the last 24 hours. In order to recover the final product, the purification process must intensify and this results in additional energy consumption.

This is the main target of FACTLOG, to be able to detect faster the off-spec production and also to create an optimal recovery plan in order to have the final product on-specs and by consuming the least additional amount of energy.

4.3.2 Process Modelling and Simulation

As already mentioned in [8], four different levels of detail have been created for the TUPRAS pilot, each one adding more information to the previous one. These different variants have been created to serve different purposes, depending on the user’s needs. They are presented in Figure 44, however, since the main focus of the pilot was around the second level of detail, the rest of the deliverable will refer to that, but a similar modelling and simulation procedure is followed for the other 3 remaining levels of detail.

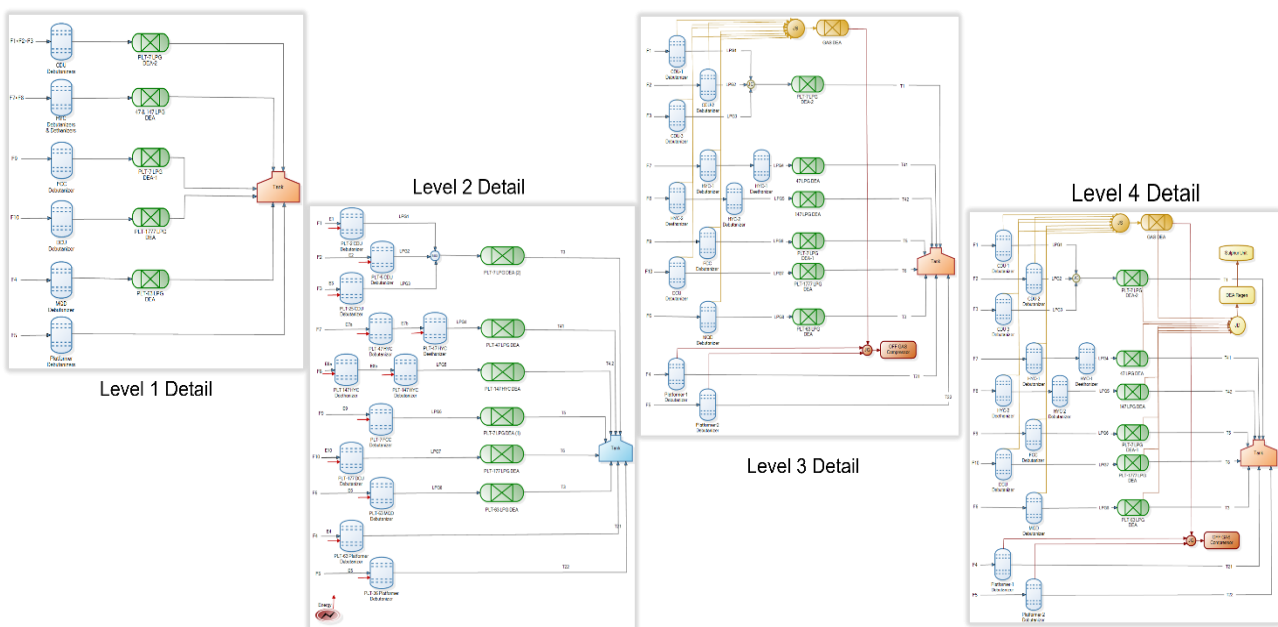


Figure 44: TUPRAS Model Levels of Detail

The second level of detail, even though it represents all the units that participates in the LPG purification process does not consider the top product (gases) extracted from the distillation columns and the DEA regeneration process. As agreed among TUPRAS and the rest of the partners participating on the pilot, in order to solve the main problem of off-spec production detection and recovery with the least amount of energy consumed in a certain time window, Level 2 model is adequate, since Level 3 or 4 will add nothing but complexity to the problem.

Focusing on Level 2, it consists of three main components; 12 Debutanizer / Deethanizer columns, 6 DEA units and the final tank that collects the LPG. It should be mentioned here that there are various configurations on the purification unit; from single debutanizer columns connected directly to the tank to debutanizer followed by deethanizer and then a DEA unit before going into the tank. These various configurations can be seen in Figure 45. We can identify ten incoming LPG streams, deriving from different processes in the plant such as Crude Distillation, Fluid Catalytic Cracking, Hydrocracking, etc. that need to be purified. Equally important are the twelve Energy feeds on the Debutanizer / Deethanizer columns, since energy consumption is of critical importance for the pilot. On the DEA units, energy consumption is not considered as according to TUPRAS it is constant and won't change the solution of the aforementioned problem.

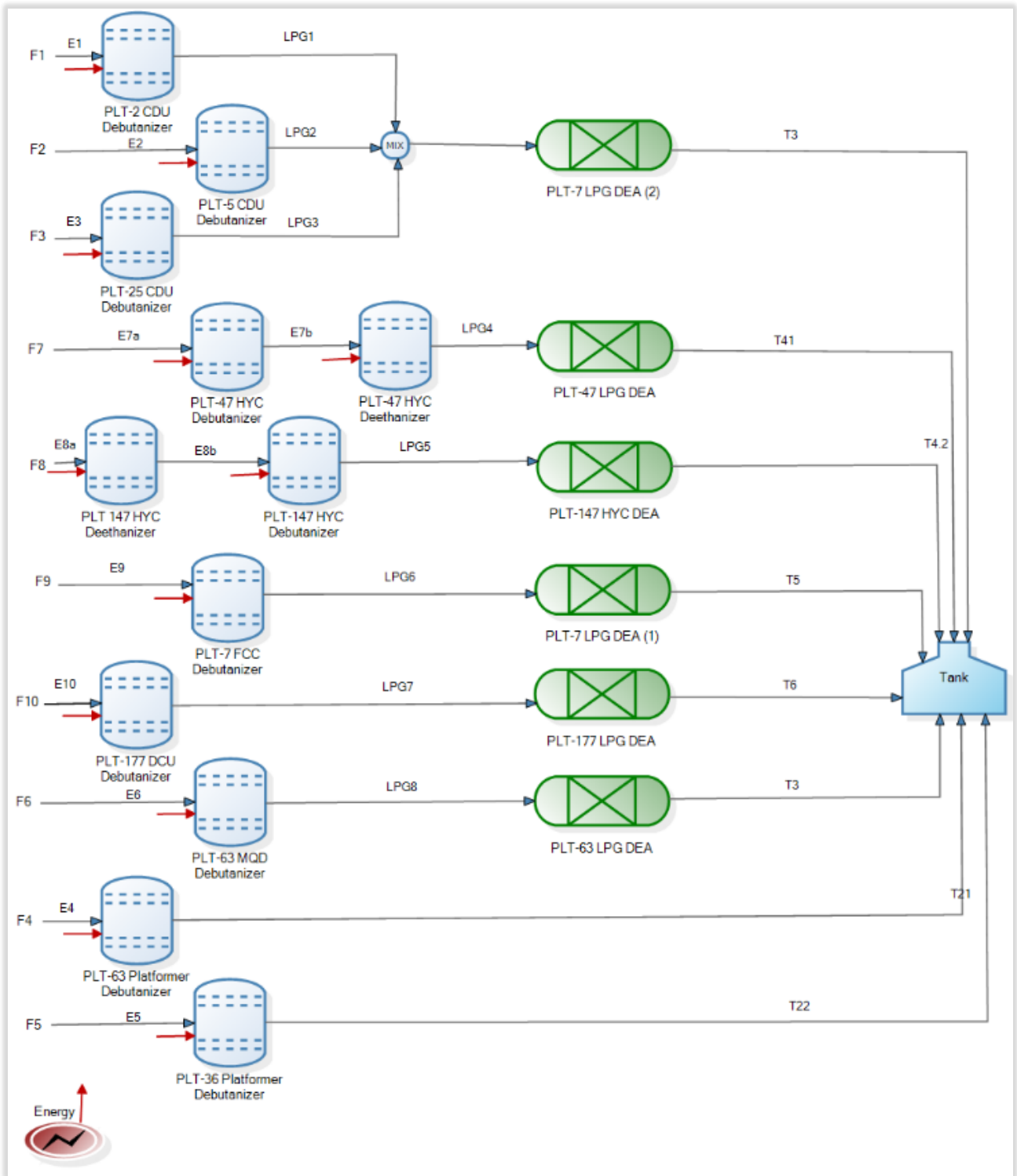


Figure 45: TUPRAS Level 2 model

A detailed description per structural unit of the model can be found below. We will describe the modelling of a debutanizer / deethanizer unit, a DEA unit and the Tank; since these are the three main components. All the other similar components have been modelled in the same way as the one described below.

4.3.2.1 Debutanizer / Deethanizer Model

A debutanizer / deethanizer is a distillation column destined to remove C2 and C5 from the LPG stream. More details on their functionality can be found in [8], so they won't be described here. In this part we are going to explain the information required to create the model in PSM. First of all, it must be mentioned here that TUPRAS and the provided information help a lot to model the LPG purification unit. This information have been discussed in depth in various meeting and analysed to provide us the required data for the model creation.

Regarding the debutanizer / deethanizer units, what must be understood is that the process and the quality of the purified LPG is affected by three parameters that can be changed by the process engineer; the column top temperature, the column top pressure and the reboiler flow (or reflow). These three values affect the C2 and C5 concentrations on the outcome of the debutanizer / deethanizer and the energy consumed in the unit.

If we focus for example on PLT-63 MQD Debutanizer we can see in Figure 46 how these process parameters have been defined in PSM tool along with their measurement units, a short description and an average value that has been calculated from TUPRAS data provided to us.

Symbol	Is Read Only?	Value	Unit	Description	Del
PRESSURE	<input checked="" type="checkbox"/>	9.92	kg/cm ²	Top Pressure (63PIC2033.PV)	X
REFLOW	<input checked="" type="checkbox"/>	49.54	kg/hr	Reboiler Flow (63FIC2085.PV)	X
TEMPERATURE	<input checked="" type="checkbox"/>	57.45	C	Top Temperature (63TI2057A.PV)	X

Figure 46: PLT-63 MQD Debutanizer process parameters

4.3.2.2 DEA Model

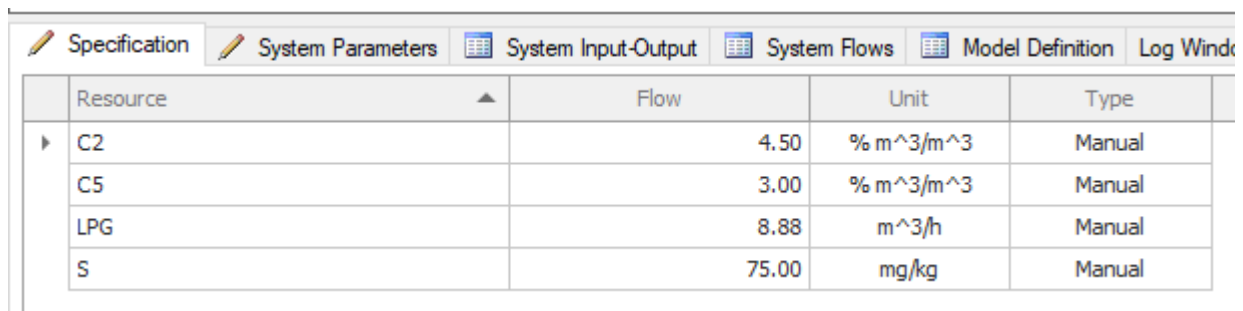
In a similar fashion, DEA units are removing Sulphur from the LPG streams. In these units there is no reboiler flow, as the bottom product is not reheated and circulated back to the unit. The control parameters of a DEA unit are the temperature and the pressure of the unit and these two parameters are affecting the absorption of Sulphur from the diethylamine. In Figure 47 the definition of these parameters on the PSM model can be seen.

Symbol	Is Read Only?	Value	Unit	Description	Del
PRESSURE	<input checked="" type="checkbox"/>	9.81	kg/cm ²	Column Pressure (63TIC2059.PV)	X
TEMPERATURE	<input checked="" type="checkbox"/>	64.55	C	Column Temperature (63PIC2033.PV)	X

Figure 47: PLT-63 LPG DEA Process Parameters

4.3.2.3 Input - Output Flows

One of the most important components of the model is the definition of the input flows, since the final results are calculated based on their composition. There are ten different input flows in the model. Each of these flows corresponds to an LPG stream that enters the purification plant. And even though, data regarding the flow of LPG has been provided there is no prior knowledge regarding the quality of the product. Thus, the C2, C5 and S concentrations have been set to 50% above the threshold. However, this does not affect the outcome of the model in any way, since the debutanizer / deethanizer and DEA units have been defined to call the JSI Machine Learning prediction models in order to retrieve C2, C5 and S based on the corresponding process parameters described above. The reason we manually set these values is in order to be able to compile the model in the PSM Tool. In Figure 48 the composition of F6 input stream can be seen



Resource	Flow	Unit	Type
C2	4.50	% m ³ /m ³	Manual
C5	3.00	% m ³ /m ³	Manual
LPG	8.88	m ³ /h	Manual
S	75.00	mg/kg	Manual

Figure 48: F6 Input Flow Composition

Regarding the output flows, a simplification has been considered regarding the LPG output flow, and this is to consider it equal to the LPG input flow. This simplifies the calculations on the model itself and it is also in line with the problem that has to be solved, that is related to LPG purification and not the lighter or heavier smaller fractions of the input mixture. The C2, C5 and S concentrations are all calculated by JSI and fed to PSM model through the PSM API that has been developed. Any required conversions are performed on the HTTP API wrapper.

4.3.2.4 Tank

The Tank is gathering the final purified product of the unit. There are various Tanks on TUPRAS plant, however every given time only one of them is being filled and only when it is full another one is selected. Based on that, the model depicts only one Tank. The Tank capacity can be changed to match the one currently filling at any given time. Also, in the Tank there are various calculations being performed on the background. Since, all the purified streams are gathered in this output, the final Level as well as the concentration of C2, C5 and S after a given amount of time are calculated there. The user (or the FACTLOG platform) needs to give the initial level of the Tank and the initial values of C2, C5 and S as well as a certain TimeStep and the simulation will produce the final values performing all the required calculations, exploiting the prediction models that have been defined on the backend of the process units. In Figure 49 the parameters of the tank model are presented, with Capacity, ConcC2Init, ConcC5Init, ConcSInit and LevelInit defined beforehand and ConcC2Final, ConcC5Final, ConcSFinal and LevelFinal being calculated by PSM.

Specification Process Parameters Node Input-Output System Parameters System Input-Output System Flows Model Definition						
Symbol: <input type="text"/>		<input type="button" value="Add"/>				
Symbol	Is Read Only?	Value	Unit	Description	Del	
Capacity	<input checked="" type="checkbox"/>	5,000	m ³		X	
ConcC2Final	<input type="checkbox"/>	2.8751174405	% m ³ /m ³		X	
ConcC2Init	<input checked="" type="checkbox"/>	5.2	% m ³ /m ³		X	
ConcC5Final	<input type="checkbox"/>	1.9248865087	% m ³ /m ³		X	
ConcC5Init	<input checked="" type="checkbox"/>	3.5	% m ³ /m ³		X	
ConcSFinal	<input type="checkbox"/>	43.810146542	mg/kg		X	
ConcSInit	<input checked="" type="checkbox"/>	50	mg/kg		X	
LevelFinal	<input type="checkbox"/>	3,289.16	m ³		X	
LevelInit	<input checked="" type="checkbox"/>	1,808	m ³		X	

Figure 49: Tank Process Parameters

4.3.3 Results

In order to provide meaningful results for TUPRAS pilot, as briefly described above a time-dependent calculation has been introduced to PSM. This allows users to calculate the impurities in the final tank and allows the direct comparison with the optimisation service calculations. These calculations have been possible since the models and the corresponding flows among units have all been deduced on a per hour basis. Based on that, the TimeStep parameter being set in the model is exploited and multiplied with the flows to calculate the total amount. In a similar fashion, because the impurities are in percentages volume / volume, having calculated the volume after some time it is easy enough to calculate and the impurities. In Figure 50 the comparison between the simulation and optimisation results is shown (as presented in FACTLOG M12 – M30 review meeting). It can be seen that they are directly comparable and that for the same amount of time in the future (twelve hours), if the plant continued working with the same process parameters both C2 and C5 will end out of spec and more energy would have been spend in comparison to the optimisation result which provides the optimal settings for each process unit, in order to have C2 and C5 on specs and consume the least possible amount of energy.

	Current Plan	Optimized Plan
LPG Quantity (m ³)	3038	3038.766
Sulphur (mg/kg)	47.447	0
C5 (mol/mol %)	2.084	1.980
C2 (mol/mol %)	3.105	2.940
Energy (kJ)	46279368	40489776

Figure 50: Comparison of Simulation and Optimisation Results

In addition to the results that can be seen in the FACTLOG platform upon calling the simulation service, through PSM Tool (or the HTTP API) there are various ways to retrieve the results of the simulation, either unit by unit, by resource or cumulatively. The output through the HTTP API is a structured document on *.json* format whereas through the PSM Tool they can also be visualized as seen in Figure 51 and Figure 52 or exported in tabular (see Figure 53) or *.txt* format.

From Node	Flow	Unit
Resource: C2 (Total = 29.09)		
PLT-7 LPG DEA (2)	3.48	% m ³ /m ³
PLT-47 LPG DEA	2.14	% m ³ /m ³
PLT-147 HYC DEA	2.98	% m ³ /m ³
PLT-7 LPG DEA (1)	4.21	% m ³ /m ³
PLT-177 LPG DEA	3.17	% m ³ /m ³
PLT-63 LPG DEA	4.71	% m ³ /m ³
PLT-63 Platformer Debutani...	4.73	% m ³ /m ³
PLT-36 Platformer Debutani...	3.66	% m ³ /m ³
Resource: C5 (Total = 1.97)		
Resource: LPG (Total = 123.43)		
Resource: S (Total = 329.56)		

Figure 51: Results grouped per Resources

To Node	Flow	Unit
Resource: Energy (Total = 4,043,537.00)		
PLT-2 CDU Debutanizer	164,365.00	kJ/h
PLT-25 CDU Debutanizer	316,891.00	kJ/h
PLT-5 CDU Debutanizer	316,891.00	kJ/h
PLT-36 Platformer Debutani...	681,228.00	kJ/h
PLT-63 Platformer Debutani...	316,891.00	kJ/h
PLT-63 MQD Debutanizer	316,891.00	kJ/h
PLT-177 DCU Debutanizer	316,891.00	kJ/h
PLT-7 FCC Debutanizer	316,891.00	kJ/h
PLT 147 HYC Deethanizer	316,891.00	kJ/h
PLT-147 HYC Debutanizer	316,891.00	kJ/h
PLT-47 HYC Debutanizer	316,891.00	kJ/h
PLT-47 HYC Deethanizer	345,925.00	kJ/h

Figure 52: Total Energy required

Resource	From Node	Unit	Name	Flow	Type
C2	PLT-7 LPG DEA (2)	% m ³ /m ³	C2 In 1	3.48	Calculated
C5	PLT-7 LPG DEA (2)	% m ³ /m ³	C5 In 1	0.02	Calculated
LPG	PLT-7 LPG DEA (2)	m ³ /h	LPG In 1	28.44	Calculated
S	PLT-7 LPG DEA (2)	mg/kg	S In 1	24.52	Calculated
C2	PLT-47 LPG DEA	% m ³ /m ³	C2 In 2	2.14	Calculated
C5	PLT-47 LPG DEA	% m ³ /m ³	C5 In 2	0.05	Calculated
S	PLT-47 LPG DEA	mg/kg	S In 2	30.83	Calculated
LPG	PLT-47 LPG DEA	m ³ /h	LPG In 2	5.69	Calculated
C2	PLT-147 HYC DEA	% m ³ /m ³	C2 In 3	2.98	Calculated
C5	PLT-147 HYC DEA	% m ³ /m ³	C5 In 3	0.24	Calculated
LPG	PLT-147 HYC DEA	m ³ /h	LPG In 3	15.45	Calculated
S	PLT-147 HYC DEA	mg/kg	S In 3	33.67	Calculated
C2	PLT-7 LPG DEA (1)	% m ³ /m ³	C2 In 4	4.21	Calculated
C5	PLT-7 LPG DEA (1)	% m ³ /m ³	C5 In 4	0.03	Calculated
LPG	PLT-7 LPG DEA (1)	m ³ /h	LPG In 4	25.83	Calculated
S	PLT-7 LPG DEA (1)	mg/kg	S In 4	23.56	Calculated
C2	PLT-177 LPG DEA	% m ³ /m ³	C2 In 5	3.17	Calculated
C5	PLT-177 LPG DEA	% m ³ /m ³	C5 In 5	1.02	Calculated
LPG	PLT-177 LPG DEA	m ³ /h	LPG In 5	16.22	Calculated
S	PLT-177 LPG DEA	mg/kg	S In 5	21.74	Calculated
C2	PLT-63 LPG DEA	% m ³ /m ³	C2 In 6	4.71	Calculated
C5	PLT-63 LPG DEA	% m ³ /m ³	C5 In 6	0.03	Calculated
LPG	PLT-63 LPG DEA	m ³ /h	LPG In 6	8.88	Calculated
S	PLT-63 LPG DEA	mg/kg	S In 6	45.24	Calculated
C2	PLT-63 Platformer Debutanizer	% m ³ /m ³	C2 In 7	4.73	Calculated
C5	PLT-63 Platformer Debutanizer	% m ³ /m ³	C5 In 7	0.05	Calculated
LPG	PLT-63 Platformer Debutanizer	m ³ /h	LPG In 7	15.14	Calculated
S	PLT-63 Platformer Debutanizer	mg/kg	S In 7	75.00	Calculated
C2	PLT-36 Platformer Debutanizer	% m ³ /m ³	C2 In 8	3.66	Calculated
C5	PLT-36 Platformer Debutanizer	% m ³ /m ³	C5 In 8	0.51	Calculated
LPG	PLT-36 Platformer Debutanizer	m ³ /h	LPG In 8	7.78	Calculated
S	PLT-36 Platformer Debutanizer	mg/kg	S In 8	75.00	Calculated

Figure 53: Results in Tabular Format

References

- [1] "FACTLOG Deliverable D4.2 (2022) - Knowledge Graph Modeling".
- [2] "<https://joinanalytics.net/why-are-knowledge-graphs-transforming-the-industry/>," [Online].
- [3] "FACTLOG Deliverable D5.2 (2022) - Robust and energy-aware planning and scheduling".
- [4] "Maksym Figat, Cezary Zieliński, Parameterised robotic system meta-model expressed by Hierarchical Petri nets, Robotics and Autonomous Systems, Volume 150, 2022, 103987, ISSN 0921-8890, <https://doi.org/10.1016/j.robot.2021.103987>".
- [5] "A. Desrochers, A. and R. Al-Jaar, "Applications of Petri Nets in Manufacturing Systems - Modeling, Control and Performance Analysis", IEEE Press, 1995."
- [6] "J.M. Proth, and N. Sauer, "Scheduling of piecewise constant product flows: a Petri net approach", European Journal of Operational Research, vol. 106, pp. 45 – 56, 1998."
- [7] "M. Kuchárik, and Z. Balogh, "Student Learning Simulation Process with Petri Nets", In: Patnaik S., Jain V. (eds) Recent Developments in Intelligent Computing, Communication and Devices. Advances in Intelligent Systems and Computing, vol.752, Springer".
- [8] "FACTLOG Deliverable D4.1 (2021) - Process Modelling Methodology".
- [9] "R.J. Brooks, A.M. Tobias, Choosing the best model: Level of detail, complexity, and model performance, Mathematical and Computer Modelling, Volume 24, Issue 4, 1996, Pages 1-14, ISSN 0895-7177".
- [10] "Z. Balogh, and M. Turčáni, "Possibilities of Modelling Web-Based Education Using IF-THEN Rules and Fuzzy Petri Nets in LMS", in Informatics Engineering and Information Science: International Conference, ICIEIS 2011, Kuala Lumpur, Malaysia".
- [11] "<https://pandas.pydata.org/>," [Online].
- [12] "<https://numpy.org/>," [Online].
- [13] "G. Arampatzis, A. Angelis-Dimakis, M. Blind, D. Assimacopoulos, A web-based toolbox to support the systemic eco-efficiency assessment in water use systems, Journal of Cleaner Production 138 (2016) 181–194, Sustainable consumption and production".

- [14] “A. Angelis-Dimakis, G. Arampatzis, D. Assimacopoulos, Systemic eco-efficiency assessment of meso-level water use systems, *Journal of Cleaner Production* 138 (2) (2016) 195–207”.
- [15] “Soltani, Nasim & Soleimani Neysiani, Behzad & Barekatin, Behrang. (2017). Heuristic Algorithms for Task Scheduling in Cloud Computing: A Survey. *International Journal of Computer Network and Information Security*. 9. 16-22. 10.5815/ijcnis.2017.08.03.”.
- [16] “Brinker, R.W.; Kinard, J.; Rummer, Robert; Lanford, B. 2002. Machine rates for selected forest harvesting machines. In: *Machine Rates for Selected Forest Harvesting Maines*, 32 p.”.

Appendix I – Continuous Process Simulation and Modelling API

API Functions	HTTP Method	Description of Function
<i>Instance Management</i>		
Solve (instanceID)	GET	Calculate all flows of the specified instance model.
Get Instance (instanceID)	GET	Get a model instance specifications by providing the instanceID.
Update Instance (instanceID)	PUT	Update an existing instance by giving the instanceID.
Delete Instance (instanceID)	DELETE	Remove an instance using the instanceID.
Get Instance Definition (instanceID)	GET	Given an instanceID, returns the model instance definition.
Create Instance (modelID)	POST	Create a new instance from an existing model by specifying the modelID.
Get Model Parameters of Instance (instanceID)	GET	Get all model parameters by specifying an instanceID.
Get Model Parameter of Instance (instanceID, parameterName)	GET	Get a model parameter by specifying the instanceID and parameterName.
Set Model Parameter of Instance (instanceID, parameterName, parameterValue)	PUT	Set a specific model parameter value by specifying the instanceID, the parameterName and the parameterValue.
Get Process Parameters of Instance (instanceID, processID/processName)	GET	Get all process parameters by specifying the instanceID and the processID or processName.
Get Process Parameter of Instance (instanceID, processID/processName, parameterName)	GET	Get a specific process parameter by specifying the instanceID, the processID or processName and the parameterName.

API Functions	HTTP Method	Description of Function
Set Process Parameter of Instance (instanceID, processID/processName, parameterName, parameterValue)	PUT	Set a specific process parameter value by specifying the instanceID, the processID or processName, the parameterName and the parameterValue.
Get Instance Flow by ID (instanceID, sourceID, targetID, resourceID)	GET	Gets a flow between two nodes utilizing the provided instanceID, sourceID of source node, targetID of target node and ResourceID.
Set Instance Flow by ID (instanceID, sourceID, targetID, resourceID, resourceFlow)	PUT	Sets a flow between two nodes utilizing the provided instanceID, sourceID of source node, targetID of target node, ResourceID and resourceFlow.
Get Instance Flow by Name (instanceID, sourceID, targetID, resourceName)	GET	Gets a flow between two nodes utilizing the provided instanceID, sourceID of source node, targetID of target node and resourceName.
Set Instance Flow by Name (instanceID, sourceID, targetID, resourceID, resourceFlow)	PUT	Sets a flow between two nodes utilizing the provided instanceID, sourceID of source node, targetID of target node, ResourceID and resourceName.
<i>Machine Learning Models Management</i>		
Get Single Debutanizer (lpg, pressure, temperature, reflow, sulfur)	POST	Calls JSI service to get predictions regarding a single debutanizer providing LPG input flow, pressure, temperature, reflow and sulfur concentration. Receives predictions regarding LPG flow, C2 concentration, C5 concentration, Sulfur and Energy requirements.
Get Pair Debutanizer (lpg, c2,sulfur, pressure, temperature, reflow)	POST	Calls JSI service to get predictions regarding a pair debutanizer providing LPG input flow, c2

API Functions	HTTP Method	Description of Function
		concentration, sulfur concentration, pressure, temperature and reflow Receives predictions regarding LPG flow, C2 concentration, C5 concentration, Sulfur and Energy requirements.
Get Pair Deethanizer (lpg, c5,sulfur, pressure, temperature, reflow)	POST	Calls JSI service to get predictions regarding a pair deethanizer providing LPG input flow, c5 concentration, sulfur concentration, pressure, temperature and reflow Receives predictions regarding LPG flow, C2 concentration, C5 concentration, Sulfur and Energy requirements.
Get DEA (lpg, c2, c5, pressure, temperature, reflow, sulfur)	POST	Calls JSI service to get predictions regarding a DEA unit providing LPG input flow, c2 concentration, c5 concentration, pressure, temperature reflow and sulfur concentration. Receives predictions regarding LPG flow, C2 concentration, C5 concentration and Sulfur.
<i>Model Management</i>		
List Models ()	GET	Get a list with all stored models.
Get Model (modelID)	GET	Get a model specifications by providing the modelID.
Delete Model (modelID)	DELETE	Remove an model using the modelID.
Get Model Definition (modelID)	GET	Given an modelID, returns the model definition.
Update Model (modelID)	PUT	Given the modelID you can update a model by providing its new

API Functions	HTTP Method	Description of Function
		configuration in PSM format.
Insert Model (modelID)	POST	Insert a new model by providing its new configuration in PSM format.
Get Model Parameters of Model (modelID)	GET	Get all model parameters by specifying a modelID.
Get Model Parameter of Model (modelID, parameterName)	GET	Get a model parameter by specifying the modelID and parameterName.
Set Model Parameter of Model (modelID, parameterName, parameterValue)	PUT	Set a specific model parameter value by specifying the instanceID, the parameterName and the parameterValue.
Get Process Parameters of Model (modelID, processID/processName)	GET	Get all process parameters by specifying the modelID and the processID or processName.
Get Process Parameter of Model (modelID, processID/processName, parameterName)	GET	Get a specific process parameter by specifying the modelID, the processID or processName and the parameterName.
Set Process Parameter of Model (modelID, processID/processName, parameterName, parameterValue)	PUT	Set a specific process parameter value by specifying the modelID, the processID or processName, the parameterName and the parameterValue.
Get Model Flow by ID (modelID, sourceID, targetID, resourceID)	GET	Gets a flow between two nodes utilizing the provided modelID, sourceID of source node, targetID of target node and ResourceID.
Set Model Flow by ID (modelID, sourceID, targetID, resourceID, resourceFlow)	PUT	Sets a flow between two nodes utilizing the provided modelID, sourceID of source node, targetID of target node, ResourceID and resourceFlow.

API Functions	HTTP Method	Description of Function
Get Model Flow by Name (modelID, sourceID, targetID, resourceName)	GET	Gets a flow between two nodes utilizing the provided modelID, sourceID of source node, targetID of target node and resourceName.
Set Model Flow by Name (modelID, sourceID, targetID, resourceID, resourceFlow)	PUT	Sets a flow between two nodes utilizing the provided modelID, sourceID of source node, targetID of target node, ResourceID and resourceName.
<i>Scenario Management</i>		
Get Model Scenarios (modelID)	GET	Get all scenarios for the specified modelID.
Remove Model Scenarios (modelID)	DELETE	Remove all scenarios that are registered to the specific modelID.
Get Scenario Parameters (modelID)	GET	Get all scenarios parameters for the specified modelID.
Get Scenarios per Node (modelID, nodeID)	GET	Get all node scenarios for the specified modelID and nodeID.
Remove Scenarios per Node (modelID, nodeID)	DELETE	Remove all scenarios that are registered to a specific model node according to modelID and nodeID.
Get Scenario Parameters per Node (modelID, nodeID)	GET	Get the scenarios parameters for the specified modelID and nodeID.

Table 4: PSM API functionalities

Appendix II – BRC Simulation Request

```

{
  "scheduleParam": {
    "route": "multi-stage-flowshop",
    "data": {
      "machines": [
        {
          "machineId": "10-FORMAT2",
          "machineType": "COIL",
          "setupTime": "00:01:07",
          "status": 1
        },
        {
          "machineId": "17-RMS MB3",
          "machineType": "BEND",
          "setupTime": "00:03:38",
          "status": 1
        }
      ],
      ...
      ...
      The rest has been removed due to space limitations.
      ...
      ...
    ],
    "orders": [
      {
        "id": "C201919720SEQ-0109",
        "jobs": [
          {
            "jobId": "4201200511002458",
            "processStage": 2,
            "processingTimes": [
              {
                "machineId": "34-MULTIBA",
                "duration": "00:06:14"
              },
              {
                "machineId": "2-KRBMINI",
                "duration": "00:18:08"
              },
              {
                "machineId": "36-DBS2",
                "duration": "01:54:28"
              },
              {
                "machineId": "35-DBS1",
                "duration": "00:31:44"
              }
            ]
          }
        ]
      },
      {
        "jobId": "5201200511002459",
        "processStage": 2,
        "processingTimes": [
          {
            "machineId": "34-MULTIBA",
            "duration": "00:06:03"
          },
          {
            "machineId": "2-KRBMINI",
            "duration": "00:17:36"
          }
        ]
      }
    ]
  }
}

```

```

        "machineId": "36-DBS2",
        "duration": "01:51:06"
      },
      {
        "machineId": "35-DBS1",
        "duration": "00:30:48"
      }
    ]
  },
  ...
  ...
  The rest has been removed due to space limitations.
  ...
  ...
],
"dueDate": "016:30:00"
}
],
"lambda": "0.5"
}
},
"optimizationParam": {
  "ObjectiveValues": {
    "Makespan": 224.42,
    "TotalLateness": 0,
    "TotalTardiness": 0
  },
  "OutputMachine": {
    "35-DBS1": {
      "jobID": [
        "8201200511002450",
        "4201200511002443",
        "3201200511002448",
        "8201200511002447",
        "5201200511002459",
        "4201200511002449",
        "3201200511002442"
      ]
    },
    "36-DBS2": {
      "jobID": [
        "4201200511002458"
      ]
    },
    ...
    ...
    The rest has been removed due to space limitations.
    ...
    ...
  },
  "OutputOrder": {},
  "OutputJob": {
    "3201200511002442": {
      "Machines": [
        "34-MULTIBA",
        "35-DBS1"
      ],
      "parentID": "C201919720SEQ-0109",
      "startTime": [
        15.17,
        190.99
      ],
      "completionTime": [
        22.55,
        224.42
      ]
    }
  }
}

```

D4.3 Systemic Cognitive Models Prototypes

```
    ]
  },
  "3201200511002448": {
    "Machines": [
      "34-MULTIBA",
      "35-DBS1"
    ],
    "parentID": "C201919720SEQ-0109",
    "startTime": [
      61.08,
      67.54
    ],
    "completionTime": [
      67.54,
      96.3
    ]
  }
}
...
...
The rest has been removed due to space limitations.
...
...
}
```

Appendix III – CONTINENTAL Simulation Request

```

{
  "InitTime": 0,
  "ScheduleParam": {
    "route": "prod-and-maint-sched",
    "data": {
      "staticData": {
        "WorkplaceTypes": [
          {
            "Id": 390,
            "Code": "CISS Test",
            "Description": "CISS Test"
          },
          {
            "Id": 310,
            "Code": "ICT test",
            "Description": "ICT test"
          }
        ]
      }
    }
  }
},
{
  "InitTime": 0,
  "ScheduleParam": {
    "route": "prod-and-maint-sched",
    "data": {
      "staticData": {
        "WorkplaceTypes": [
          {
            "Id": 390,
            "Code": "CISS Test",
            "Description": "CISS Test"
          },
          {
            "Id": 310,
            "Code": "ICT test",
            "Description": "ICT test"
          }
        ]
      }
    }
  }
},
...
...
The rest has been removed due to space limitations.
...
...
],
"LineTypes": [
  {
    "Id": 24,
    "Code": "FA",
    "Description": "FA"
  },
  {
    "Id": 29,
    "Code": "PRA",
    "Description": "PRA"
  }
],
"Lines": [
  {
    "Id": 24,
    "Code": "FA",
    "Description": "FA",
    "LineTypesId": 24
  }
]

```

```

        {
            "Id": 29,
            "Code": "PRA",
            "Description": "PRA",
            "LineTypesId": 29
        }
    ],
    "Workplaces": [
        {
            "Id": 990,
            "Code": "FA_Laser marking on housing_28",
            "WorkplaceTypesId": 335,
            "LinesId": 24,
            "SequenceInLine": 28
        },
        {
            "Id": 1058,
            "Code": "PRA_CTT HIGH TEMP TEST_8",
            "WorkplaceTypesId": 317,
            "LinesId": 29,
            "SequenceInLine": 8
        },
        ...
        ...
        The rest has been removed due to space limitations.
        ...
    ],
    "ProductFamilies": [
        {
            "Id": 190,
            "Name": "pcb assy VW40MEB D_10-1",
            "Description": "pcb assy VW40MEB D_10-1"
        },
        {
            "Id": 194,
            "Name": "ACU VW40 MY19 S30 12 06 0 0 0 LOW",
            "Description": "ACU VW40 MY19 S30 12 06 0 0 0 LOW"
        },
        ...
        ...
        The rest has been removed due to space limitations.
        ...
    ],
    "Products": [
        {
            "Id": 238,
            "Name": "A3C0808590100",
            "Description": "A3C0808590100",
            "EfficiencyRate": 100,
            "ProductFamiliesId": 190,
            "SourceLineTypesId": 29,
            "EndLineTypesId": 29
        },
    ],

```



```

    {
      "Id": 204,
      "Name": "A3C0864060200",
      "Description": "A3C0864060200",
      "EfficiencyRate": 99.97983870967742,
      "ProductFamiliesId": 214,
      "SourceLineTypesId": 24,
      "EndLineTypesId": 24
    },
    ...
    The rest has been removed due to space limitations.
    ...
  ],
  "Resources": [],
  "ResourceBOM": [],
  "ProductBOM": [
    {
      "Multiplicity": 1,
      "WorkplacesId": 898,
      "ProductsId": 210,
      "RequiredProductsId": 223
    },
    {
      "Multiplicity": 1,
      "WorkplacesId": 898,
      "ProductsId": 204,
      "RequiredProductsId": 238
    },
    ...
    The rest has been removed due to space limitations.
    ...
  ],
  "SetupTimes": [
    {
      "ProductFamiliesId": 206,
      "SetupTime": 865,
      "LineTypesId": 29
    },
    {
      "ProductFamiliesId": 199,
      "SetupTime": 1123,
      "LineTypesId": 24
    },
    ...
    The rest has been removed due to space limitations.
    ...
  ],
  "ProcessingTimes": [
    {
      "WorkplaceTypesId": 328,
      "ProductsId": 261,
      "IdealProcessingTime": 416,
      "RealProcessingTime": null
    },
  ],

```

```

        {
            "WorkplaceTypesId": 359,
            "ProductsId": 210,
            "IdealProcessingTime": 141,
            "RealProcessingTime": null
        },
        ...
        ...
        The rest has been removed due to space limitations.
        ...
        ...
    ]
},
"dynamicData": {
    "ProductionOrders": [
        {
            "Id": 633,
            "Name": "262333943",
            "Quantity": 1248,
            "DueDate": "2022-06-11T00:00:00.000",
            "Priority": 1,
            "MaxQuantity": 1248,
            "ProductsId": 256
        },
        {
            "Id": 272,
            "Name": "260813891",
            "Quantity": 650,
            "DueDate": "2022-06-07T00:00:00.000",
            "Priority": 1,
            "MaxQuantity": 650,
            "ProductsId": 216
        },
        ...
        ...
        The rest has been removed due to space limitations.
        ...
        ...
    ],
    "ScheduledMaintenanceActivities": []
}
},
"OptimizationParam": {
    "uuid": "b20a8a42-afdb-490c-b024-a9c7f7bdeaae",
    "data": {
        "Metrics": {
            "Makespan": 777132,
            "TotalTardiness": 0,
            "AvgWorkStationIdleTime": 365566.73913043475,
            "TotalWorkStatIdleTime": 8408035
        },
        "Schedule": [
            {
                "End": 1723,
                "Start": 946,
                "LineID": 29,
                "OrderId": 96,
                "ProductId": 261,
                "WorkplaceID": 1034,
                "WorkplaceTypeID": 343
            },
            {

```



```

"InitTime": 0,
"ScheduleParam": {
  "route": "prod-and-maint-sched",
  "data": {
    "staticData": {
      "WorkplaceTypes": [
        {
          "Id": 390,
          "Code": "CISS Test",
          "Description": "CISS Test"
        },
        {
          "Id": 310,
          "Code": "ICT test",
          "Description": "ICT test"
        },
        ...
        ...
        The rest has been removed due to space limitations.
        ...
        ...
      ],
      "LineTypes": [
        {
          "Id": 24,
          "Code": "FA",
          "Description": "FA"
        },
        {
          "Id": 29,
          "Code": "PRA",
          "Description": "PRA"
        }
      ],
      "Lines": [
        {
          "Id": 24,
          "Code": "FA",
          "Description": "FA",
          "LineTypesId": 24
        },
        {
          "Id": 29,
          "Code": "PRA",
          "Description": "PRA",
          "LineTypesId": 29
        }
      ],
      "Workplaces": [
        {
          "Id": 990,
          "Code": "FA_Laser marking on housing_28",
          "WorkplaceTypesId": 335,
          "LinesId": 24,
          "SequenceInLine": 28
        },
        {
          "Id": 1058,
          "Code": "PRA_CTT HIGH TEMP TEST_8",
          "WorkplaceTypesId": 317,
          "LinesId": 29,
          "SequenceInLine": 8
        },
        ...
        ...
      ]
    }
  }
}

```

```

    The rest has been removed due to space limitations.
    ...
    ...
  ],
  "ProductFamilies": [
    {
      "Id": 190,
      "Name": "pcb assy VW40MEB D_10-1",
      "Description": "pcb assy VW40MEB D_10-1"
    },
    {
      "Id": 194,
      "Name": "ACU VW40 MY19 S30 12 06 0 0 0 LOW",
      "Description": "ACU VW40 MY19 S30 12 06 0 0 0 LOW"
    },
    ...
    ...
    The rest has been removed due to space limitations.
    ...
    ...
  ],
  "Products": [
    {
      "Id": 238,
      "Name": "A3C0808590100",
      "Description": "A3C0808590100",
      "EfficiencyRate": 100,
      "ProductFamiliesId": 190,
      "SourceLineTypesId": 29,
      "EndLineTypesId": 29
    },
    {
      "Id": 204,
      "Name": "A3C0864060200",
      "Description": "A3C0864060200",
      "EfficiencyRate": 99.97983870967742,
      "ProductFamiliesId": 214,
      "SourceLineTypesId": 24,
      "EndLineTypesId": 24
    },
    ...
    ...
    The rest has been removed due to space limitations.
    ...
    ...
  ],
  "Resources": [],
  "ResourceBOM": [],
  "ProductBOM": [
    {
      "Multiplicity": 1,
      "WorkplacesId": 898,
      "ProductsId": 210,
      "RequiredProductsId": 223
    },
    {
      "Multiplicity": 1,
      "WorkplacesId": 898,
      "ProductsId": 204,
      "RequiredProductsId": 238
    },
    ...
    ...
  ]

```

```

    The rest has been removed due to space limitations.
    ...
    ...
  ],
  "SetupTimes": [
    {
      "ProductFamiliesId": 206,
      "SetupTime": 865,
      "LineTypesId": 29
    },
    {
      "ProductFamiliesId": 199,
      "SetupTime": 1123,
      "LineTypesId": 24
    },
    ...
    ...
    The rest has been removed due to space limitations.
    ...
    ...

  ],
  "ProcessingTimes": [
    {
      "WorkplaceTypesId": 328,
      "ProductsId": 261,
      "IdealProcessingTime": 416,
      "RealProcessingTime": null
    },
    {
      "WorkplaceTypesId": 359,
      "ProductsId": 210,
      "IdealProcessingTime": 141,
      "RealProcessingTime": null
    },
    ...
    ...
    The rest has been removed due to space limitations.
    ...
    ...
  ]
},
"dynamicData": {
  "ProductionOrders": [
    {
      "Id": 633,
      "Name": "262333943",
      "Quantity": 1248,
      "DueDate": "2022-06-11T00:00:00.000",
      "Priority": 1,
      "MaxQuantity": 1248,
      "ProductsId": 256
    },
    {
      "Id": 272,
      "Name": "260813891",
      "Quantity": 650,
      "DueDate": "2022-06-07T00:00:00.000",
      "Priority": 1,
      "MaxQuantity": 650,
      "ProductsId": 216
    },
    ...
    ...
    The rest has been removed due to space limitations.
  ]
}

```

```

...
...
    ],
    "ScheduledMaintenanceActivities": []
  }
}
},
"OptimizationParam": {
  "uuid": "b20a8a42-afdb-490c-b024-a9c7f7bdeaae",
  "data": {
    "Metrics": {
      "Makespan": 777132,
      "TotalTardiness": 0,
      "AvgWorkStationIdleTime": 365566.73913043475,
      "TotalWorkStatIdleTime": 8408035
    },
    "Schedule": [
      {
        "End": 1723,
        "Start": 946,
        "LineID": 29,
        "OrderId": 96,
        "ProductId": 261,
        "WorkplaceID": 1034,
        "WorkplaceTypeID": 343
      },
      {
        "End": 6332,
        "Start": 3483,
        "LineID": 29,
        "OrderId": 96,
        "ProductId": 261,
        "WorkplaceID": 1085,
        "WorkplaceTypeID": 333
      },
      ...
      ...
      The rest has been removed due to space limitations.
      ...
      ...
    ]
  },
  "produced_at": 1654528214522
}
}

],
"ProductFamilies": [
  {
    "Id": 190,
    "Name": "pcb assy VW40MEB D_10-1",
    "Description": "pcb assy VW40MEB D_10-1"
  },
  {
    "Id": 194,
    "Name": "ACU VW40 MY19 S30 12 06 0 0 0 LOW",
    "Description": "ACU VW40 MY19 S30 12 06 0 0 0 LOW"
  },
  ...
  ...
  The rest has been removed due to space limitations.
  ...
  ...
]

```

```

"Products": [
  {
    "Id": 238,
    "Name": "A3C0808590100",
    "Description": "A3C0808590100",
    "EfficiencyRate": 100,
    "ProductFamiliesId": 190,
    "SourceLineTypesId": 29,
    "EndLineTypesId": 29
  },
  {
    "Id": 204,
    "Name": "A3C0864060200",
    "Description": "A3C0864060200",
    "EfficiencyRate": 99.97983870967742,
    "ProductFamiliesId": 214,
    "SourceLineTypesId": 24,
    "EndLineTypesId": 24
  },
  ...
  ...
  The rest has been removed due to space limitations.
  ...
  ...
],
"Resources": [],
"ResourceBOM": [],
"ProductBOM": [
  {
    "Multiplicity": 1,
    "WorkplacesId": 898,
    "ProductsId": 210,
    "RequiredProductsId": 223
  },
  {
    "Multiplicity": 1,
    "WorkplacesId": 898,
    "ProductsId": 204,
    "RequiredProductsId": 238
  },
  ...
  ...
  The rest has been removed due to space limitations.
  ...
  ...
],
"SetupTimes": [
  {
    "ProductFamiliesId": 206,
    "SetupTime": 865,
    "LineTypesId": 29
  },
  {
    "ProductFamiliesId": 199,
    "SetupTime": 1123,
    "LineTypesId": 24
  },
  ...
  ...
  The rest has been removed due to space limitations.
  ...
  ...

```



```

    ],
    "ProcessingTimes": [
      {
        "WorkplaceTypesId": 328,
        "ProductsId": 261,
        "IdealProcessingTime": 416,
        "RealProcessingTime": null
      },
      {
        "WorkplaceTypesId": 359,
        "ProductsId": 210,
        "IdealProcessingTime": 141,
        "RealProcessingTime": null
      },
      ...
    ]
    ...
    The rest has been removed due to space limitations.
    ...
  ],
  "dynamicData": {
    "ProductionOrders": [
      {
        "Id": 633,
        "Name": "262333943",
        "Quantity": 1248,
        "DueDate": "2022-06-11T00:00:00.000",
        "Priority": 1,
        "MaxQuantity": 1248,
        "ProductsId": 256
      },
      {
        "Id": 272,
        "Name": "260813891",
        "Quantity": 650,
        "DueDate": "2022-06-07T00:00:00.000",
        "Priority": 1,
        "MaxQuantity": 650,
        "ProductsId": 216
      },
      ...
    ]
    ...
    The rest has been removed due to space limitations.
    ...
  ],
  "ScheduledMaintenanceActivities": []
}
}
},
"OptimizationParam": {
  "uuid": "b20a8a42-afdb-490c-b024-a9c7f7bdeaae",
  "data": {
    "Metrics": {
      "Makespan": 777132,
      "TotalTardiness": 0,
      "AvgWorkStationIdleTime": 365566.73913043475,
      "TotalWorkStatIdleTime": 8408035
    },
    "Schedule": [
      {
        "End": 1723,
        "Start": 946,
        "LineID": 29,

```

```
    "OrderId": 96,  
    "ProductId": 261,  
    "WorkplaceID": 1034,  
    "WorkplaceTypeID": 343  
  },  
  {  
    "End": 6332,  
    "Start": 3483,  
    "LineID": 29,  
    "OrderId": 96,  
    "ProductId": 261,  
    "WorkplaceID": 1085,  
    "WorkplaceTypeID": 333  
  },  
  ...  
  ...  
  The rest has been removed due to space limitations.  
  ...  
  ...  
] }  
"produced_at": 1654528214522  
}
```

Appendix IV – BRC Simulation Result

```

{
  "simulationTime":
  [
    {
      "time": "1881.83"
    }
  ],
  "machinesUsed":
  [
    {
      "machineName": "2-KRBMINI",
      "usageTime": "961.13",
      "percentageOfSimTime": "51.07"
    },
    {
      "machineName": "34-MULTIBA",
      "usageTime": "1117.25",
      "percentageOfSimTime": "59.37"
    },
    {
      "machineName": "17-RMS MB3",
      "usageTime": "1858.24",
      "percentageOfSimTime": "98.75"
    },
    {
      "machineName": "35-DBS1",
      "usageTime": "1865.28",
      "percentageOfSimTime": "99.12"
    },
    {
      "machineName": "36-DBS2",
      "usageTime": "1821.32",
      "percentageOfSimTime": "96.78"
    }
  ],
  "completionTimePerOrder":
  [
    {
      "order": "C202011389SEQ-0585",
      "completionTime": "388.7"
    },
    {
      "order": "C201910830SEQ-0063",
      "completionTime": "122.61"
    },
    {
      "order": "C20211762SEQ-0001",
      "completionTime": "898.49"
    },
    {
      "order": "C20211366SEQ-0020",
      "completionTime": "800.71"
    },
    {
      "order": "C202016982SEQ-0029",
      "completionTime": "181.04"
    },
    {
      "order": "C202016982SEQ-0027",
      "completionTime": "870.52"
    }
  ],
}

```

```
{
  "order": "C202018775SEQ-0023",
  "completionTime": "699.23"
},
{
  "order": "C202016982SEQ-0026",
  "completionTime": "356.65"
},
{
  "order": "C20202176SEQ-0175",
  "completionTime": "367.73"
},
{
  "order": "C202011389SEQ-0660",
  "completionTime": "925.91"
},
{
  "order": "C202017929SEQ-0053",
  "completionTime": "773.73"
},
{
  "order": "C202014734SEQ-0018",
  "completionTime": "454.2"
},
{
  "order": "C202014734SEQ-0009",
  "completionTime": "643.16"
},
{
  "order": "C202016982SEQ-0028",
  "completionTime": "482.65"
},
{
  "order": "C20211161SEQ-0001",
  "completionTime": "649.99"
},
{
  "order": "C20211508SEQ-0001",
  "completionTime": "974.28"
},
{
  "order": "C20211366SEQ-0012",
  "completionTime": "539.66"
},
{
  "order": "C202011389SEQ-0663",
  "completionTime": "658.82"
},
{
  "order": "C202011389SEQ-0664",
  "completionTime": "946.33"
},
{
  "order": "C20202176SEQ-0188",
  "completionTime": "791.15"
},
{
  "order": "C20209431SEQ-0442",
  "completionTime": "689.87"
},
{
  "order": "C202011389SEQ-0634",
  "completionTime": "1092.72"
},
{
  "order": "C202013207SEQ-0135",
```

```
"completionTime": "912.39"
},
{
  "order": "C202011389SEQ-0662",
  "completionTime": "931.13"
},
{
  "order": "C202014734SEQ-0008",
  "completionTime": "744.22"
},
{
  "order": "C202013207SEQ-0131",
  "completionTime": "849.12"
},
{
  "order": "C202011389SEQ-0648",
  "completionTime": "1113.13"
},
{
  "order": "C202011389SEQ-0647",
  "completionTime": "1106.57"
},
{
  "order": "C202011389SEQ-0659",
  "completionTime": "934.21"
},
{
  "order": "C20212166SEQ-0009",
  "completionTime": "869.04"
},
{
  "order": "C201919770SEQ-0311",
  "completionTime": "959.65"
},
{
  "order": "C202017929SEQ-0050",
  "completionTime": "901.49"
},
{
  "order": "C202011389SEQ-0632",
  "completionTime": "1085.06"
},
{
  "order": "C202012020SEQ-0094",
  "completionTime": "961.13"
},
{
  "order": "C202011389SEQ-0633",
  "completionTime": "1094.78"
},
{
  "order": "C202018775SEQ-0024",
  "completionTime": "745.38"
},
{
  "order": "C202011389SEQ-0640",
  "completionTime": "329.56"
},
{
  "order": "C202017929SEQ-0052",
  "completionTime": "543.42"
},
{
  "order": "C20211539SEQ-0010",
  "completionTime": "663.79"
},
}
```

D4.3 Systemic Cognitive Models Prototypes

```
{
  "order": "C202012020SEQ-0091",
  "completionTime": "1108.82"
},
{
  "order": "C202011389SEQ-0635",
  "completionTime": "1054.54"
},
{
  "order": "C202012020SEQ-0092",
  "completionTime": "1881.83"
},
{
  "order": "C202017929SEQ-0049",
  "completionTime": "1034.46"
}
]
```

Appendix V – CONTINENTAL Simulation Result

```

{
  "simulationTime":
  [
    {
      "simEnd": "263284"
    },
    {
      "simDuration": "263284"
    }
  ],
  "machinesUsed":
  [
    {
      "workplaceName": "PRA_Create unit in WIP_1",
      "processingTime": "70778",
      "setupTime": "84",
      "totalTime": "70862",
      "percentageOfTotalTime": "26.91"
    },
    {
      "workplaceName": "PRA_PANEL SEPERATION_2",
      "processingTime": "68014",
      "setupTime": "84",
      "totalTime": "68098",
      "percentageOfTotalTime": "25.86"
    },
    {
      "workplaceName": "PRA_ICT test_4",
      "processingTime": "57484",
      "setupTime": "84",
      "totalTime": "57568",
      "percentageOfTotalTime": "21.87"
    },
    {
      "workplaceName": "PRA_CTT LOW TEMP TEST_6",
      "processingTime": "46851",
      "setupTime": "84",
      "totalTime": "46935",
      "percentageOfTotalTime": "17.83"
    },
    {
      "workplaceName": "PRA_CTT HIGH TEMP TEST_8",
      "processingTime": "54303",
      "setupTime": "84",
      "totalTime": "54387",
      "percentageOfTotalTime": "20.66"
    },
    {
      "workplaceName": "FA_Create unit in WIP_1",
      "processingTime": "56616",
      "setupTime": "130",
      "totalTime": "56746",
      "percentageOfTotalTime": "21.55"
    },
    {
      "workplaceName": "FA>Loading_2",
      "processingTime": "39068",
      "setupTime": "130",
      "totalTime": "39198",
      "percentageOfTotalTime": "14.89"
    }
  ],
}

```

```

    "workplaceName": "FA_Press Fit componets_4",
    "processingTime": "46924",
    "setupTime": "130",
    "totalTime": "47054",
    "percentageOfTotalTime": "17.87"
  },
  {
    "workplaceName": "FA_Pressfit pin check_5",
    "processingTime": "60476",
    "setupTime": "130",
    "totalTime": "60606",
    "percentageOfTotalTime": "23.02"
  },
  {
    "workplaceName": "FA_Plasma Cleaning_7",
    "processingTime": "41945",
    "setupTime": "130",
    "totalTime": "42075",
    "percentageOfTotalTime": "15.98"
  },
  {
    "workplaceName": "FA_SEALING ASSEMBLY_8",
    "processingTime": "51787",
    "setupTime": "130",
    "totalTime": "51917",
    "percentageOfTotalTime": "19.72"
  },
  {
    "workplaceName": "FA_Screwing_10",
    "processingTime": "54871",
    "setupTime": "130",
    "totalTime": "55001",
    "percentageOfTotalTime": "20.89"
  },
  {
    "workplaceName": "FA_Curing oven_12",
    "processingTime": "32267",
    "setupTime": "130",
    "totalTime": "32397",
    "percentageOfTotalTime": "12.3"
  },
  {
    "workplaceName": "FA_Run in_13",
    "processingTime": "61418",
    "setupTime": "130",
    "totalTime": "61548",
    "percentageOfTotalTime": "23.38"
  },
  {
    "workplaceName": "FA_Shaker_15",
    "processingTime": "55777",
    "setupTime": "130",
    "totalTime": "55907",
    "percentageOfTotalTime": "21.23"
  },
  {
    "workplaceName": "FA_Rollover test_17",
    "processingTime": "86609",
    "setupTime": "130",
    "totalTime": "86739",
    "percentageOfTotalTime": "32.95"
  },
  {
    "workplaceName": "FA_Scanning OP80 before CISS test_19",
    "processingTime": "70386",
    "setupTime": "130",

```



```

    "totalTime": "70516",
    "percentageOfTotalTime": "26.78"
  },
  {
    "workplaceName": "FA_CISS Test_20",
    "processingTime": "84829",
    "setupTime": "130",
    "totalTime": "84959",
    "percentageOfTotalTime": "32.27"
  },
  {
    "workplaceName": "FA_Lekeage test assembly_22",
    "processingTime": "61219",
    "setupTime": "130",
    "totalTime": "61349",
    "percentageOfTotalTime": "23.3"
  },
  {
    "workplaceName": "FA_Final test_24",
    "processingTime": "79906",
    "setupTime": "130",
    "totalTime": "80036",
    "percentageOfTotalTime": "30.4"
  },
  {
    "workplaceName": "FA_Bent Pin Camera Check_26",
    "processingTime": "50538",
    "setupTime": "130",
    "totalTime": "50668",
    "percentageOfTotalTime": "19.24"
  },
  {
    "workplaceName": "FA_Laser marking on housing_28",
    "processingTime": "82078",
    "setupTime": "130",
    "totalTime": "82208",
    "percentageOfTotalTime": "31.22"
  },
  {
    "workplaceName": "FA_Unloading_30",
    "processingTime": "55386",
    "setupTime": "130",
    "totalTime": "55516",
    "percentageOfTotalTime": "21.09"
  }
],
"completionTimePerOrder":
[
  {
    "orderId": "1",
    "orderName": "order rlpwHHCGorXmK8XH03e",
    "completionTime": "199230"
  },
  {
    "orderId": "2",
    "orderName": "order ymynK1etnIPHhy8m4HN",
    "completionTime": "221806"
  },
  {
    "orderId": "3",
    "orderName": "order yUHJp0eENolKk0EBw4uh",
    "completionTime": "232888"
  },
  {
    "orderId": "4",
    "orderName": "order tIU3O98FI771FdrNG21n",

```

```
    "completionTime": "263284"
  },
  {
    "orderId": "5",
    "orderName": "order 4wkhbl24o4A19Fh4R9eL",
    "completionTime": "171030"
  },
  {
    "orderId": "6",
    "orderName": "order jbRQBa8ObWXGjTHKvk0n",
    "completionTime": "240969"
  },
  {
    "orderId": "7",
    "orderName": "order QC8xJYQe8NGQ8Fo85EX0",
    "completionTime": "229531"
  },
  {
    "orderId": "8",
    "orderName": "order VlbPYhzcDkQDqQ1TWJMS",
    "completionTime": "258533"
  },
  {
    "orderId": "9",
    "orderName": "order aDKSvdsImoim4q9y9nIE",
    "completionTime": "87541"
  },
  {
    "orderId": "10",
    "orderName": "order 5HU79rkq8X1XqHEmHIQh",
    "completionTime": "240174"
  }
]
}
```