

# ENERGY-AWARE FACTORY ANALYTICS FOR PROCESS INDUSTRIES

Deliverable D4.1  
**Process Modelling Methodology**

**Version**  
Version 1.0

**Lead Partner**  
TUC

**Date**  
31/05/2021

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

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

## Dissemination Level

X	<b>PU</b>	Public
	<b>CO</b>	Confidential, restricted under conditions set out in the Grant Agreement
	<b>CI</b>	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

Process Simulation Modelling is an overall methodological scheme, suitable for modelling and simulating most or even all types of process industries, as well as suitable for providing the capacities and services involved in the process.

In the context of FACTLOG, Process Simulation Model denotes a generic model with all related methods, algorithms, mechanisms, services and tools it directly uses, integrated into an overall modelling application or platform. In any specialized model, these methods, algorithms and mechanisms do not change. Process Simulation Modelling interconnects and interoperates with external AI tools, Optimisation tools, Analytics tools, etc.

To model any system, its state space needs to be defined, i.e., the variables that govern the behaviour of the system with respect to the metrics being estimated. If the variables continuously change over time, it is a continuous system. If the system state instantaneously changes at discrete points in time, instead of continuously, it is a discrete system.

The baseline for system modelling will be the process modelling methodology, whereby the system modelled is analysed into machines/processes, production stages, resource stocks/warehouses, inputs, outputs and material/energy flows. All model entities are organised into a hierarchical inheritance registry that provides prototype reconfigurable building blocks for building any industrial system model.

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

- 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 adaptive, in the sense that (a) the models support/facilitate the physical system's adaptation to new goals/targets/conditions/possibilities, and (b) the models can adapt closely following any structural, methodological or operational changes made to the physical systems.

The Process Simulation and Modelling Tool is a tool developed in the context of Task 4.1, in order 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 industry. An Application Programming Interface has also been created, to assist the integration of Process Simulation Modelling module into the Digital Twins Platform of FACTLOG.

FACTLOG has five interesting and different industrial pilot cases. This document presents the outputs from the transformation of use cases, defined in WP1, into Process Models. A deep analysis of the pilot cases and the corresponding process models are provided.

## Revision History

Revision	Date	Description	Organisation
0.1	12/04/2021	Deliverable Structure; Table of Contents	TUC
0.2	07/05/2020	TUPRAS Pilot Case Process Modelling	TUC
0.3	11/05/2021	Process Simulation and Modelling Tool	TUC
0.5	14/05/2021	PSM Tool API	TUC
0.6	17/05/2021	Continuous Process Modelling	TUC
0.6	17/05/2021	Discrete Process Modelling	TUC
0.6	19/05/2021	BRC Pilot Case Process Modelling	TUC
0.7	20/05/2021	JEMS Pilot Case Process Modelling	JSI
0.7	22/05/2021	Continental Pilot Case Process Modelling	TUC
0.8	24/05/2021	Python API for discrete modelling	TUC
0.8	25/05/2021	1 <sup>st</sup> Major Revision (preparation for internal review)	TUC
0.9	28/05/2021	PIACENZA Pilot Case Process Modelling	UNIFI
1.0	29/05/2021	Final version after reviewing from UNIFI and JSI	TUC, UNIFI, JSI

## Contributors

Organisation	Author	E-Mail
TUC	George Arampatzis	<a href="mailto:garampatzis@pem.tuc.gr">garampatzis@pem.tuc.gr</a>
TUC	George Tsinarakis	<a href="mailto:tsinar@dpem.tuc.gr">tsinar@dpem.tuc.gr</a>
TUC	Nikolaos Sarantinoudis	<a href="mailto:nsarantinoudis@isc.tuc.gr">nsarantinoudis@isc.tuc.gr</a>
JSI	Aljaž Košmerlj	<a href="mailto:aljaz.kosmerlj@ijs.si">aljaz.kosmerlj@ijs.si</a>
UNIPi	Pavlos Eirnakis	<a href="mailto:pavlose@gmail.com">pavlose@gmail.com</a>
UNIPi	Konstantinos Kaparis	<a href="mailto:k.kaparis@uom.edu.gr">k.kaparis@uom.edu.gr</a>
AUEB	Yiannis Mourtos	<a href="mailto:mourtos@aueb.gr">mourtos@aueb.gr</a>

# Table of Contents

<b>Executive Summary</b>	<b>3</b>
<b>Revision History</b>	<b>4</b>
<b>1 Introduction</b>	<b>10</b>
1.1 Purpose and Scope	10
1.2 Relation with other Deliverables	10
1.3 Structure of the Document	10
<b>2 Process Modelling and Simulation</b>	<b>12</b>
2.1 The role of Process Modelling and Simulation	12
2.2 Interaction with other Modules of FACTLOG Platform	13
2.3 Process Modelling and Simulation Concepts	14
2.3.1 Process Model	14
2.3.2 Dynamic Process Model	15
2.3.3 Adaptive Process Model	15
2.3.4 Hierarchical Entities Registry	15
2.3.5 Simulation	16
2.4 Modelling and Simulation of Continuous Processes	16
2.4.1 Problem Definition	17
2.4.2 Identification of Controlling Factors or Mechanisms	17
2.4.3 Problem Data Evaluation	17
2.4.4 Model Development	17
2.4.5 Solution Procedure Selection	18
2.5 Modelling and Simulation of Discrete Processes	18
2.5.1 Introduction	18
2.5.2 Types of Events	19
2.5.3 Discrete Process Modelling Concepts	19
2.5.4 Discrete Process Modelling Levels	20

2.5.5	Input Data.....	21
2.5.6	Modelling Procedure .....	22
2.5.7	Output – Model.....	23
2.6	Overview of Process Simulation and Modelling Procedure .....	23
<b>3</b>	<b>Process Simulation Modelling Tool .....</b>	<b>26</b>
3.1	PSM Stand-Alone Tool .....	26
3.2	Continuous Processes Modelling in PSM Tool.....	26
3.3	Discrete Process Modelling .....	29
<b>4</b>	<b>Process Modelling and Simulation Service API.....</b>	<b>31</b>
4.1	Continuous Process Modelling API .....	31
4.2	Discrete Process Modelling .....	37
<b>5</b>	<b>Process Models in FACTLOG Pilot Cases.....</b>	<b>39</b>
5.1	Continuous Processes Pilot Cases.....	39
5.1.1	Oil Refineries: Pilot Case by TUPRAS .....	39
5.1.2	Waste-to-Fuel Transformer Plants: Pilot Case by JEMS .....	47
5.2	Discrete Pilot Cases .....	49
5.2.1	Steel Manufacturing: Pilot Case by BRC .....	49
5.2.2	Automotive Manufacturing: Pilot Case by CONTINENTAL.....	58
5.2.3	Textile Industry: Pilot Case by PIACENZA .....	64
	<b>References .....</b>	<b>66</b>



## List of Figures

Figure 1: Flow Chart of Process Simulation and Modelling procedure .....	25
Figure 2: PSM Tool Interface and Splash Screen .....	26
Figure 3: PSM Tool User Interface Elements.....	28
Figure 4: Transition Firing Example .....	30
Figure 5: PSM Tool API main interface.....	34
Figure 6: PSM Tool API Services for Models.....	34
Figure 7: PSM Tool API Services for Instances .....	35
Figure 8: PSM Tool API schemas.....	35
Figure 9: Life-Cycle of the API use .....	36
Figure 10: Basic Example of a Timed Petri Net .....	37
Figure 11: Output of Python software .....	38
Figure 12: Initial and final graphs of a Petri Net .....	38
Figure 13: Process Model progressive level of detail.....	40
Figure 14: Level 1 process model of TUPRAS LPG purification plant .....	42
Figure 15: Level 2 process model of TUPRAS LPG purification plant .....	43
Figure 16: Level 3 process model of TUPRAS LPG purification plant .....	44
Figure 17: Level 4 process model of TUPRAS LPG purification plant .....	45
Figure 18: JEMS plant pipeline split into the four main stages of the process .....	48
Figure 19: Stream Story model for stage 1 - Drying and Mixing .....	49
Figure 20: Production process in Bay 3 .....	50
Figure 21: Basic Timed PN model of a process.....	51
Figure 22: Timed PN model of a process with machine setup. ....	51
Figure 23: Timed PN model of a machine that performs 4 types of process.....	52
Figure 24: Timed PN model of a machine that performs 4 types of process with different release dates. ....	53
Figure 25: Timed PN model of a machine that performs 4 types of process with different release dates following a defined sequence. ....	53
Figure 26: Initial state (scenario) of the Timed PN model of a machine that performs 4 types of process with different release dates following a defined sequence. ....	54
Figure 27: Final state of the Timed PN model of a machine that performs 4 types of process with different release dates following a defined sequence .....	55
Figure 28: Timed PN model of a machine with breakdowns (unreliable machine) that performs 4 types of process with different release dates following a defined sequence....	55
Figure 29: Timed PN model of two interacting machines that perform two types of process .....	56
Figure 30: Implementation of the machine sequence constraints in Timed PN structure. ....	56
Figure 31: Timed Petri net model of a complicated production scenario in Bay 3.....	58
Figure 32: Continental production line stages.....	60



Figure 33: Basic Timed PN model of a process.....	60
Figure 34: Timed PN model of a process with machine setup .....	61
Figure 35: Timed PN model of a machine that performs 4 types of process.....	61
Figure 36: Timed PN model of a machine with breakdowns (unreliable machine) that performs 4 types of processes.....	62
Figure 37: Timed Petri net model of two consecutive workplaces (from the same production line) that perform 4 types of processes .....	63
Figure 38: Overall System (Preassembly and final Assembly lines) Timed Petri net model for the scenario defined in Table 4.....	64

## List of Tables

Table 1: PSM Tool API Services.....	33
Table 2: LPG Specifications that the final product must meet.....	40
Table 3: Job sequence in each machine as defined from optimisation module. ....	57
Table 4: Description of a scenario composed from a 4 Job sequence for each production line .....	63

# 1 Introduction

## 1.1 Purpose and Scope

The objective of Task 4.1 “Industrial Systems Adaptive Dynamic Process Modelling” is the development of methodologies and tools for producing industrial dynamic process models. It is the first Task of WP4 “Knowledge Graph and Process Modelling” and aims to analyse the requirements of the Use Cases defined in WP1 and transform them into process models using state of the art standards for system modelling and analysis, thus supporting the specification, design, verification and validation of the modelled system.

This document corresponds to the Deliverable 4.1 “Process Modelling Methodology” and summarizes the work undertaken for the development of:

- process modelling methodologies, suitable to the FACTLOG industrial pilot cases;
- a prototype Process Simulation Modelling Tool to facilitate the creation of industrial process models and support the simulation of the corresponding industrial systems;
- an Application Programming Interface, to assist the integration of Process Simulation Modelling module into the Digital Twins Platform of FACTLOG;
- detailed Process Models for all FACTLOG industrial pilot cases.

## 1.2 Relation with other Deliverables

This deliverable builds on top of the descriptions of the pilot use-cases in D1.1 Reference Scenarios, KPIs and Datasets. Though short summaries of the pilots are given in Chapter 5, a reader should consult D1.1 for their full descriptions.

This document serves as a base for a lot of present and future work in the project. It contains the requirements specification for all the next steps within WP4 and is as such the base for all its future deliverables: D4.2: Knowledge Graphs Modelling, D4.3: Systemic Cognitive Models Prototypes, D4.4: KG-based analytics for process optimisation. It also provides information for system integration deliverables (D6.3 – D6.5) as well as to system installation and testing deliverables (D7.1, D7.2).

## 1.3 Structure of the Document

The document is structured as follows:

- **Chapter 2** provides an overview of the main concepts behind industrial process modelling and simulation, by identifying the purpose, the main approaches and characteristics. It also provides an analysis on the role of process modelling and simulation in the cognition process as well as the interaction with other FACTLOG modules. Finally, it details the methodologies for modelling continuous and discrete systems.

- **Chapter 3** focuses on the description of the prototype Process Simulation Modelling Tool. The main concepts and a description of the operational environment are first presented followed by the application in continuous and discrete systems.
- **Chapter 4** presents the Application Programming Interface for the Process Modelling and Simulation module of FACTLOG Platform, focusing on the key concepts behind that, its services and functionalities as well as the whole lifecycle of the services provided.
- **Chapter 5** focuses on the Process Models for all FACTLOG industrial pilot cases. It provides the description and the explanation of the developed process models, according to the principles described in previous chapters.

## 2 Process Modelling and Simulation

### 2.1 The role of Process Modelling and Simulation

In the context of FACTLOG, **Process Simulation Model (PSM)** denotes a generic model with all related methods, algorithms, mechanisms, services and tools it directly uses, integrated into an overall modelling application or platform. In any specialized (e.g., FACTLOG use cases) model, these methods, algorithms and mechanisms do not change. It is just the process model (digital) representation per se that changes. PSM interconnects and interoperates with external AI tools, Optimisation tools, Analytics tools, etc. These may also change from subclass to subclass etc. according to specific application needs. However, PSM itself does not change from application case to case.

This digital model core may have two components: (a) a knowledge engineering component in the form of a Knowledge Graph Model, that represents formal system knowledge and data in the form of rules, relations, associations and predicates; (b) a dynamic operational model of the production system, usually (but not always) in the form of some **Dynamic Petri Net** or **Continuous Process-Flow Model**.

Typically, the structure of a model represents all the entities involved in its operation as well as a set of possible states in which the physical system can be found after the appearance of certain sequences of events. Events define the mechanism for change of state of the system (and the respective model) and refer to interactions between entities of the system and between the system and the external environment. A significant difference between these categories of events is that it is much more difficult to control or eliminate the events in the external behaviour and their effects in the system under study (such events are called as uncontrollable in the literature and in many cases the results of their appearance may take extra time to be observed). In addition to the structure of the model, its dynamic state must be defined. This represents at any time instant the active state and enables the set of possible upcoming events (active state enables certain possible upcoming events).

Ideally, the role of the central (production system) modelling component can be defined in relation to the cognition process, as follows:

1. Create a **dynamic digital shadow**<sup>1</sup> of the physical system, providing its accurate image of its operation at any time, updated by getting real-time monitoring data from the analytics platform. The dynamic image can be also used by man-machine interfaces for system monitoring, control and management.
2. Accurately model all production **processes** as well as all entities along the product's value chain and all inputs and outputs, providing a comprehensive **systemic view** and **operational statistics**.

---

<sup>1</sup> A Digital Shadow is a detailed and accurate enough digital image (i.e. a model) of the real physical system that is being continuously updated along the time dimension. Thus, the digital shadow is live, i.e. it follows closely in time the activity, flows and state of the real system and its components.

3. Use KPIs and business rules and benchmarks to **assess** the performance of the entire system and each component; also compare estimated and real-time data for **self-assessment**.
4. Provide **systemic knowledge** (entities, relations, material flows, process states, performance) and operational data to the knowledge engineering component (e.g. Knowledge Graph) and the machine-learning tools, while using the stored knowledge to make the model more intelligent.
5. Provide support for root-cause analysis, risk analysis and hypothesis testing to **AI problem inference** tools.
6. Provide use case models and data to support **optimisation** algorithms and tools.
7. Provide base model and support for demand-supply and other **prediction** tools or in-build routines.
8. Support system **adaptability** by building, running and assessing system adaptation scenarios.

## 2.2 Interaction with other Modules of FACTLOG Platform

The PSM interacts with four (4) other components: (1) Knowledge Graph Model (KGM), (2) KGM-based analytics (3) AI tools for problem detection, (4) Optimisation toolkit.

1. **KGM** responds to knowledge & data queries submitted to it by other modules, while PSM implements the triggers that initiate most of the corresponding actions. Since many model entities have in any case dynamic behaviour, they have rules (and/or they can even query the KGM) and can generate events (and thus, actions too). Therefore, PSM makes a first evaluation of process/system states/status, which can create a system event that in turn triggers an action.
2. The PSM will use the **analytics services** to estimate KPIs for assessing (a) the performance of the current system (or single processes), either at regular intervals or on request), and/or (b) the performance of a simulated scenario (what-if analysis). The Analytics tool will provide services that PSM can use, communicating directly with it. However, for analytics to be computed, data from the model assessed (current/shadow model or scenario) must be passed to the analytics tool.
3. The **AI tools** for problem inference are expected to have a close relationship with PSM. These tools export several services/functions or I/O channels. Given that the AI tools detect an anomaly, PSM can either verify or refine (or both) this AI inference. PSM can have on-line data about past system and process states, flows and events and can thus back-trace (from the point an anomaly is first detected) in system space and time, e.g., to detect possible causes and synergies (secondary effects) of the anomaly and contributions to it, which are not localized in a single input stream (or in the current time slice). Also, PSM can perform simulation, running a scenario of the current system in isolation from the current shadow twin, in order to test some hypothesis about the anomaly (e.g., “a sub-standard quantity of some resource

causes the anomaly by affecting another process upstream” or “the anomaly observed at this point will affect other processes downstream effecting X impact on the production”, etc.). Such scenarios consist of two parts, a modified system model and an assumption (another scenario) about its probable inputs. The main use for PSM simulation will be for assessing scenarios about potential physical system changes that might be used for adapting the production system to a changing internal or external environment.

4. The **Optimisation toolkit**, for production process or production line optimisation, is expected to have a close relationship to PSM. The PSM – Optimisation Tools relationship can be expected to have the following use scenario: The Optimisation Tool asks PSM for some model & data, to be used for running its optimisation algorithm; PSM provides model & data; the Optimisation Tool provides the policy that optimises the KPIs considered. Optionally, the Optimisation Tool communicates result to PSM and asks for evaluation & confirmation. In addition, the process model can be used to evaluate a subset of feasible alternative schedules (suboptimal) through simulation in order to provide the final selection to the Decision makers.

In conclusion, the PSM is designed as an “active” module, which means that it would be self-aware and fully capable of: (a) processing and filtering its input; (b) understanding its state (or the modelled system’s state) and the individual states of its entities; (c) assessing the modelled system’s performance.

## 2.3 Process Modelling and Simulation Concepts

**Process Simulation Modelling** is an overall methodological scheme, suitable for modelling and simulating most or even all types of process industries, as well as suitable for providing the capacities (behavioural vectors) and services involved in the process. In the modelling and simulation procedure, there are three basic elements:

- **(Process) Model:** It is the system specification, defining the static and dynamic structure and behaviour needed to generate data comparable to data from the real world. Generally, a model is a set of instructions, rules, equations, constraints, for generating input/output behaviour.
- **Simulator:** A kind of agent (any computation system) that executes the instructions of the model, generating the behaviour.
- **Experimental Frame:** The specification of the conditions under which the system is observed, allowing the experimentation and validation of the model.

### 2.3.1 Process Model

To model any system, its state space needs to be defined, i.e., the variables that govern the behaviour of the system with respect to the metrics being estimated. If the variables continuously change with time, it is called a **continuous system**. If the system state instantaneously changes at discrete points in time, instead of continuously, it is called a **discrete system**.

The main objectives of a process model, both discrete and continuous, are to develop a scientific understanding through quantitative expression of knowledge of a system by

displaying what we know but may also show up things that we do not know. It is also ideal for testing the effects of changes in a system, make predictions and help in decision-making, including both tactical decisions by managers as well as strategic decisions by planners.

The choice of the best model is the problem of choosing the appropriate level of detail and it is considered as one of the most difficult aspects of the modelling process. A model that is too simple will be unrealistic and its results will be of little use. Considerable resources are required to build a complex model and so, if the model is too complex, constraints on resources may make it inefficient. It is much harder to understand the relationships contained in a complex model and this makes the interpretation of the results more difficult, leading to incorrect conclusions being drawn. A complex model is probably more likely to contain errors, as it is harder to verify that it is working as intended [1]. A similar process has been followed in the creation of the process models for FACTLOG project and is described in detail in Chapter 5, where each individual pilot case is analysed. Following the above guidelines, it is possible to create process models according to the needs and requirements of the application [2].

### 2.3.2 Dynamic Process Model

Process models developed in the context of FACTLOG are **Dynamic Models**. This means that key flows, as well as key process control settings (i.e. variable parameters) are continuously updated in near-real-time, in connection to a real-time monitoring system (sensor network), maintaining a digital Shadow of the physical System. Flows and settings that are not monitored must be estimable from monitored flows with sufficient accuracy. On the basis of these flows and settings, the model can identify (or “understand”) key system events, estimate the operating state of system processes and the overall status of the modelled system at any time, also enabling the prediction of the next probable state change event.

### 2.3.3 Adaptive Process Model

There are two entirely different approaches to adaptation that can make a production system model ‘**adaptive**’:

- a) the model supports/facilitates the **physical system’s adaptation** to new goals/targets/conditions/possibilities;
- b) the **model itself is adaptable**, i.e. it can adapt closely following any structural, methodological or operational changes made to the physical system.

### 2.3.4 Hierarchical Entities Registry

The **PSM Tool**, developed in the context of FACTLOG (Chapter 3), makes possible the representation of the static structure of the system, its dynamic behaviour as well as the flow of materials and information through it. All model entities are organised into a **Hierarchical Inheritance Registry** that provides prototype reconfigurable **building blocks** for building any industrial system model.

The hierarchical structure is developed storing all model entities (e.g. processes, stages, agents, actors, stocks, I/O nodes) that form the primary building blocks of all models registered into the FACTLOG platform through the **Process Modelling API** (Chapter 4). Models registered include the generic model, and any specific system models created during



the life-time of the project or afterwards (in a FACTLOG application's lifetime). Entities of these models are stored as a hierarchy of increasingly more specialized entities. All Registry objects can be re-used as is or as parents for creating new objects of that class by using a build-in inheritance mechanism. In this manner, descendant entities (and descendants of other types of registry objects) can be created by using single or multiple inheritance of properties and relations and/or semantic overloading.

### 2.3.5 Simulation

**Simulation** uses the implemented model of the system, in order to conduct experiments, also referred to as **scenarios**, between others for purposes of understanding the behaviour of the system, testing the effect of changes in the system, making predictions and aiding decision making (including tactical decisions by managers and strategic decisions by planners) as well as for evaluating alternative strategies for the operation of the system.

In the context of FACTLOG, a simulation corresponds to a complete virtual run of a production system's model (a scenario). Throughout the simulation, model flows, process state changes (optionally other system events also, e.g. contingencies, maintenance) are estimated and system status statistics are calculated (synthesized) and aggregated to produce KPIs for assessment. The virtual run fully simulates a physical system's typical operation, by simulating its inputs and using the processes' transition functions to estimate flows and outputs.

Simulations are performed on either:

- a) **system change scenarios**, build as part of the system adaptation support mechanism, described previously;
- b) **contingency scenarios** (hypothesis tests), build in response to feedback from the AI tools; (c) **optimisation models**, either used by optimisation algorithms or tools

Thus, the **simulation methodology** should be a combination of **continuous** (or, equivalently, time-step) and **event-driven**.

## 2.4 Modelling and Simulation of Continuous Processes

A general systematic approach to model continuous processes has emerged from the numerous models which have been set up and used in process engineering. This systematic approach endeavours to provide good engineering practice in process modelling. This section contains the basic principles of model building, outlining the elements and procedures of a systematic approach by addressing the following issues:

- The concept of a process system and the modelling goal, as well as the effect of the goals on the process model.
- The general notion of a model, different types of models and mathematical models.
- The description of the steps in a modelling procedure illustrated by simple examples and an explanation of the iterative nature of the model building process.

- The necessary modelling ingredients, namely the necessary elements of a process model: the assumptions, the model equations and any initial and boundary conditions which are put into a systematic format.

A **continuous process system** is usually a system in which physical and/or chemical processes are taking place. The system to be modelled could be seen as the whole process plant, its environment, part of the plant, an operating unit or an item of equipment. Hence, to define the system we need to specify its boundaries, its inputs and outputs and the physicochemical processes taking place within the system. Process systems are conventionally specified in terms of a **flowsheet** which defines the boundaries together with inputs and outputs. Information is normally available about the internal structure of the system in terms of the operating units and their connections. The modelling goal specifies the intended use of the model. The modelling goal has a major impact on the level of detail and on the mathematical form of the model which will be built.

Like other engineering tasks, good practice requires models to be constructed following a well-defined sequence of steps. These steps are introduced below and further described in [3].

#### 2.4.1 Problem Definition

This step refines the description of the process system with the modelling goal. Moreover, it fixes the degree of detail relevant to the modelling goal and specifies:

- inputs and outputs
- hierarchy level relevant to the model
- the type of spatial distribution
- the necessary range and accuracy of the model

#### 2.4.2 Identification of Controlling Factors or Mechanisms

The next step is to investigate the physio-chemical processes and phenomena taking place in the system relevant to the modelling goal. These are termed controlling factors or mechanisms.

#### 2.4.3 Problem Data Evaluation

Most models of real process systems are of grey-box type models (models that combine both first principles and data), therefore, we almost always need to use either measured process data directly or estimated parameter values in our models.

#### 2.4.4 Model Development

The models are usually based on **first principles**, corresponding to conservation balances (e.g. heat and mass balances), chemical reactions, etc. A model building sub-procedure can be used to obtain a syntactically and semantically correct process model. In cases where it is complicated to create the equations **data driven models** (statistical or machine learning models) can be created based on historical data. Ideally the combination of historical data and some principal equations can create the most accurate models, which are called **hybrid models**.

### 2.4.5 Solution Procedure Selection

Having set up a mathematical model, we must identify its mathematical form and find or implement a solution procedure. In all cases, we must ensure that the model is well posed such that the "excess" variables or degrees of freedom are satisfied.

## 2.5 Modelling and Simulation of Discrete Processes

### 2.5.1 Introduction

A **Discrete Event System** is a system where state changes (events) happen at discrete instances in time, and events take zero time to happen. It is assumed that nothing (i.e., nothing interesting) happens between two consecutive events, that is, no state change takes place in the system between the events (in contrast to continuous systems where state changes are continuous). In such systems, the production procedure of finished products with given specifications is countable in any time instant and in any part of the system and can be considered as a type of manufacturing process that involves raw materials, resources, parts, components, and sub-assemblies.

Systems that can be viewed as Discrete Event Systems can be modelled using **Discrete event simulation** (DES), an iterative method used to model real world systems that can be decomposed into a set of logically separated processes that autonomously progress through time. In any time instant, there is a number of pending events that form a list. Among them, the event that is realized is the one with the minimum occurrence time. For this reason, a fundamental feature of discrete event simulation is the simulation clock that keeps the current time of the system and compares it to the occurrence times of the pending events. Each event occurs on a specific process and is assigned a logical time (a timestamp). The result of this event can be an outcome passed to other processes as typically there are interactions between the entities of the modelled system. Depending on the outcome, new events might be generated and processed at some specified future logical time while others might be removed from the list.

Typical areas where discrete event simulation is used are the following:

- Design and evaluation of new manufacturing processes. Discrete event simulation enables the comparison of alternative types of equipment as well as the simulation of the designed procedure in order to evaluate equipment regarding its ability to operate in a certain way.
- Performance study and improvement of existing manufacturing processes. In this case parts of equipment that constrain the efficiency of the whole system can be detected and ways to overcome such issues can be tested. Change of specific parameters allows the definition of scenarios that need to be studied.
- Establishment of optimum operational policies. This can be done with the implementation of policies received from the use of other methods such as algorithms to support production planning and scheduling. For this concept, it must be noted that depending on the characteristics of the input optimisation scheduling algorithms can be computationally expensive and, in such cases, simulation of certain scenarios can be used to compare policies by perturbing a given solution (a.k.a. running a sensitivity analysis) and improve the quality of the proposed strategies.

### 2.5.2 Types of Events

There are different types of events that take place in a system and change its state. The event space can be classified further:

- **External input events:** events that are triggered from outside the system e.g., a customer arrival at the queue that may lead to changes (for example rescheduling because of high priority of the specific customer).
- **External output events:** events that are generated as output from the system, e.g., order departure. Such events play an active role regarding resource usage and the activation of certain constraints.
- **Internal events:** changes in state variables that do not affect the system environment, e.g., start service, machine breakdown, change of the speed of a machine etc.

The contexts simulated are usually viewed as queuing networks where individual entities pass through a series of discrete events one by one at discrete intervals, between which they have to wait in queues due to the constrained availability of resources. DES allow decision makers to conduct “**what if**” analyses by changing the operational scenarios and rules, to predict the possible impacts resulting from a variety of policy alternatives before truly translated into practice without any disturbances of the real system.

### 2.5.3 Discrete Process Modelling Concepts

The Methodology followed for modelling of discrete processes is a modular one as this serves the already mentioned use of hierarchical models. A module is a fundamental subsystem with a set of input and output connection arcs defining interactions with other modules and a set of internal discrete and continuous relations that define the module’s internal hybrid state.

Modelling in an industrial application typically concerns a number of common concepts. The most important of these are the following:

- Definition, Description and representation of entities (such as machines, people and equipment). This part is the initial part of the modelling process as it is the initial step for the definition of the structure of the model.
- Definition of ways of interactions between entities. In this step the models of the specific entities are appropriately interconnected in order to represent the flows of information, materials and possibly other resources in the system. This step implements the structure of the overall system model using as structural components the individual models of the entities composing it.
- Definition and representation of the events that occur in the system and change its state as well as of their characteristics. Generally speaking, events are of different categories, as certain of them are controllable (for example change of process performed in a machine) while others are not fully or at all controllable (for example

machine breakdowns) and change the state of system components not in the direction of improving its performance.

- Definition of the quantitative parameters of the system. Typically, such parameters refer to time and quantities (for example resources participating in a process). In the case of controllable events, the time instants at which they will occur are specified (or the delays associated to their occurrence), while in the case of uncontrollable events other parameters should be specified (for example the distribution followed, consequences in specific entities such as machines etc.).
- Definition and description of the variables that are necessary in order to represent the state of the system as well as to measure certain KPIs and areas of interest and ways to calculate them or record them during the simulations must be specified. This part is very important also for the performance evaluation of the system under the scenarios considered.
- Representation of the initial state of the system. This refers to the initial inventory (of raw materials, semi-processed parts and products) as well as to the orders that must be processed.
- Implementation in the structure of the model of the applied policies and strategies such as the schedules calculated from optimisation algorithms. With respect to the individual characteristics of the system under study, orders should be fed to the respective machines or lines in given sequences.

#### 2.5.4 Discrete Process Modelling Levels

According to the approach followed concerning the discrete process models, the following parts-levels are considered for representation and simulation purposes.

- **Material processing level** represents the main part of the model. In this the processes performed in order to transform raw materials in final products are represented. The entities composing the production system, such as machines, moving equipment, etc. are represented and interactions between them are defined. Also, resources of the system are represented and their use through the system is defined.
- **Control level** is implemented by adding extra characteristics to the model. In this part of the model certain quantitative parameters are defined (such as buffer capacities, process conditions etc.) and ancillary structures are added in the model in order to control its behaviour. These parameters and structures allow for the constraint of parts of model's behaviour and accordingly their state space (possible reachable states during simulation) within given limits. Typically, through the implementation of ancillary model structures sets of mutually exclusive resource states are defined.
- **Decision Making** is implemented in the model through the definition of quantitative parameters and the addition of certain structures in the model. Typical such decisions concern priorities between the occurrence of certain events (represented in Petri nets (PNs) as transition firing) when conflicts are met, the implementation of the schedule calculated from optimisation module and resource allocation through time.

### 2.5.5 Input Data

For the implementation of the model structure and the definition of its parameters in order to enable simulation, several input data are required. These data are generally of two types: static data, that refer to characteristics of the production system that do not change, such as machine types, machines, resources, etc.; and dynamic data, that refer to the specific scenario considered, such as jobs, products, processes, moving and setup times, maintenance activities, etc. Certain types of data have to be provided to the PSM module by the information system of each pilot under study (orders, release dates, time horizon, etc.) and other from modules as Analytics (data connected with breakdowns, setup and processing times) and Optimisation (that calculates the schedule). The exact types of the input data are defined per case as the structure of the production lines have different characteristics and the exact optimisation problem is slightly different in each case. Typically, the following superset of types of parameters is necessary in order to construct a model and simulate specific production scenarios (in certain problems, parts of these data can be omitted if events as the transfer of parts are considered of negligible duration or if certain characteristics are studied):

- Flow charts to study the connection between machines and cranes (resource allocation) and create the **modular models of system entities as well as their connections** according to this.
- Catalogue of all the resources of the production line as well as the possible use of each one for the production of final products.
- Initial inventory for a case. This refers to the initial levels of internal buffers of the system as well as parts in the machines. This practically is defined as the initial marking of the net (initial token distribution) and is used to define the initial state of the system together with the states of the machines.
- Production orders and jobs. In many cases a production order is composed of different jobs that need to be processed in different machines and produce parts with different characteristics. Product quantities, due dates, priorities between orders, if they can be split in suborders, product special features and resources used for each should be known.
- Machine families. Machines with similar characteristics that can be used for the performance of certain tasks.
- Product families. Products belonging to a family may have common characteristics as certain processes received, common setup times, machines used for the performance of specific processes etc.
- The number and type of different processes performed in each machine (schedule) in order to define the exact structure of its model. This is received from the optimisation module and can be different for every machine or common for every production line with respect to the specific characteristics of the case considered. The proposed schedule might be different if the objective function changes.



- Types, multiplicities of machines and processes performed in each case. Also, possible constraints that refer to specific machines must be known especially for use during scheduling.
- Delay associated to the performance of each type of process in each machine. When a process cannot be performed in a machine the respective delay should not be filled.
- Data concerning maintenance activities. These activities can be considered to be of two types, the scheduled ones and the non-scheduled ones that refer to machine breakdowns. In both cases, the time between two consecutive events or distribution followed if such data exist and are not considered negligible should be known in order to simulate a specific scenario as well as the delay representing the duration of such an event.
- Possible priorities between processes in each machine when this is not specified by a certain schedule.
- Delays representing machine setup when the type of processes performed changes. According to the problem considered these delays can be sequence dependent, non-sequence dependent, common for product families or different for each type of process performed.
- Delays representing loading/unloading and transfer of parts from buffers to machines and vice versa. According to the case these data can be part specific or independent of the characteristics of the parts. In the first case optimisation has to consider these data in detail while in the second one they can be represented as lag times.
- Maximum Capacities of places (especially of the ones representing internal buffers) as well as maximum availability of certain resources.

### 2.5.6 Modelling Procedure

The modelling procedure follows a well-defined sequence of steps in order to produce the Timed Petri net model of the scenario considered for study. Each step calls for specific input data and is used to define parameters (in the structure or concerning quantitative characteristics of the model implemented).

- In the first step the numbers and types of the machines considered have to be defined (static data). These will be the entities that are appropriately combined (with restrictions concerning their types) to form the overall model of the system. If for a given scenario certain machines are not used then the overall model can be simplified in order to represent only the active machines and reduce its overall complexity and use of resources.
- Then the jobs considered for optimisation in the overall system as well as their release dates and the time horizon considered must be fed into the system. If an order is composed of a single job then the two terms can be used interchangeably. In different cases connection between orders and jobs must be defined for use in other modules and operations (e.g., Logistics)



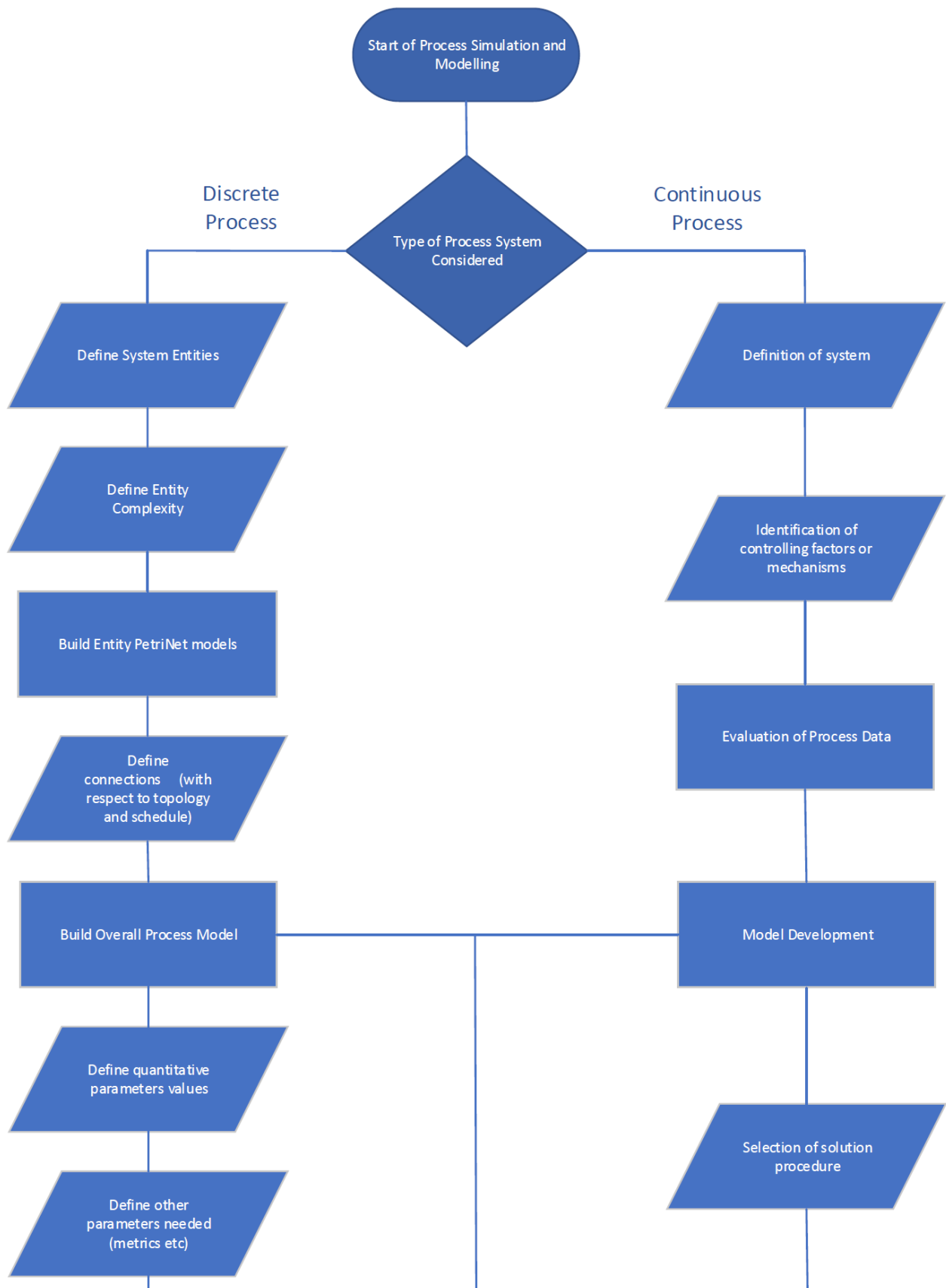
- The output of the optimisation module will be input for the simulation model. More precisely the number of jobs assigned in each machine as well as their sequence are determined. These data are necessary in order to define the exact structure and complexity of the model of each machine.
- The overall system PN is obtained via synthesis of the component models by considering the system's topology and its constraints. The overall system properties may be calculated with respect to the properties of the fundamental modules models used.
- The values of the quantitative parameters of the entities composing the model must be added as well. Examples of such parameters are processing times, setup times (and, if available, movement durations) as provided to PSM Module.
- Finally, release dates of the jobs can be added to the PN structure of the model (assuming they are not equal to zero). This expands the modelling power of the presented method and increases the scenarios that can be considered for simulation.

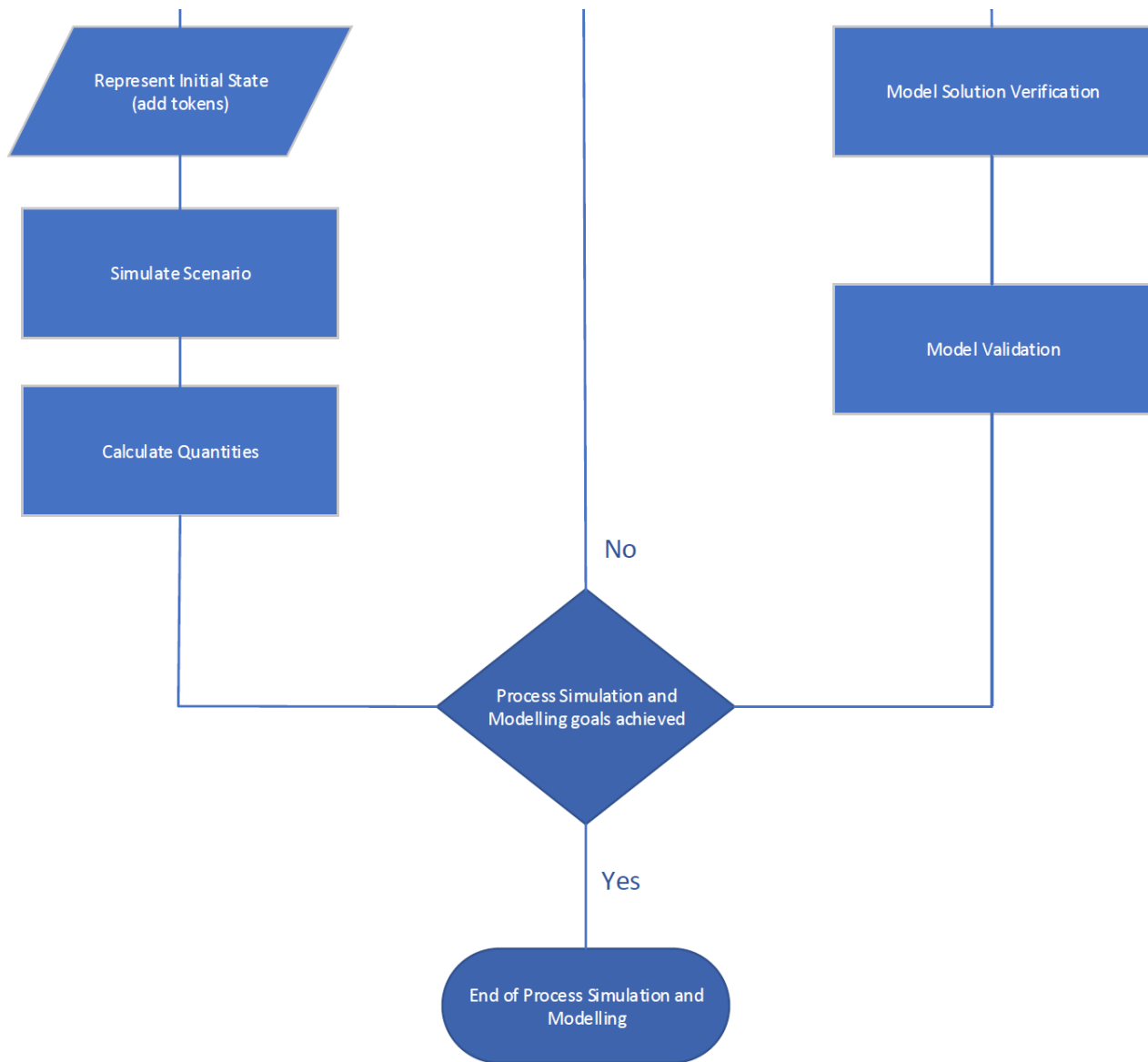
### 2.5.7 Output – Model

Following the previously described procedure, the Timed Petri net model of the scenario under study is constructed. This representation is a first output of the PSM module. Changes of specific parameters can be used for sensitivity analysis and for comparison of different scenarios through simulation. Also, calculation of metrics related to the use of resources of the system is possible with respect to the needs of other modules. Finally, the completion times of the jobs can be compared to the due dates given from the customers in order to specify possible problems regarding the delivery of products.

## 2.6 Overview of Process Simulation and Modelling Procedure

The following Figure 1 illustrates the required steps for building a process model, either continuous or discrete.





**Figure 1: Flow Chart of Process Simulation and Modelling procedure**

### 3 Process Simulation Modelling Tool

#### 3.1 PSM Stand-Alone Tool

The **Process Simulation and Modelling Tool (PSM Tool)** is a tool developed by the Technical University of Crete in order to address the process modelling and simulation requirements of the FACTLOG project. PSM allows the user not only to create a process industry model (both continuous and discrete) but also to simulate the operation of such an industry. At the moment the process modelling, and simulation of continuous processes can be entirely designed and simulated with the PSM Tool while for discrete processes an assortment of 3<sup>rd</sup> party tools and custom software is being tested currently and is scheduled to be integrated into the PSM tool under the work in Task 4.3.

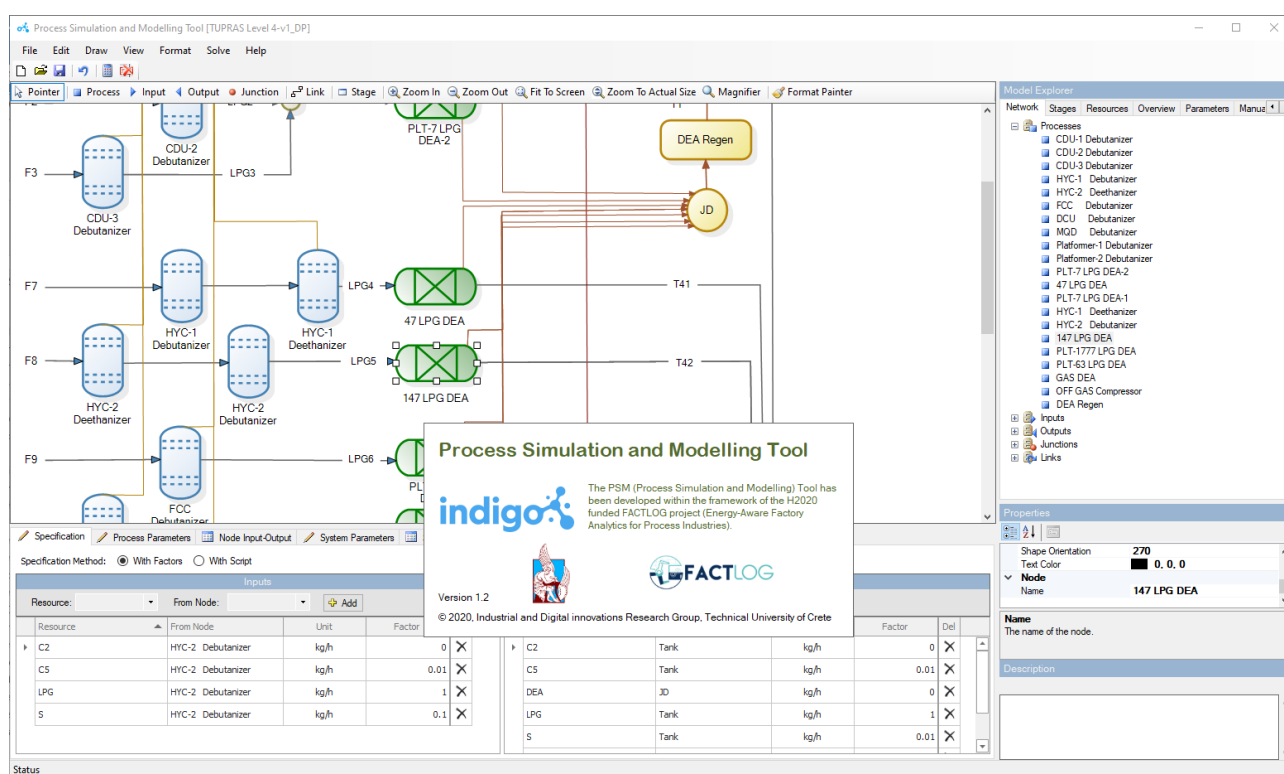


Figure 2: PSM Tool Interface and Splash Screen

#### 3.2 Continuous Processes Modelling in PSM Tool

PSM Tools 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.

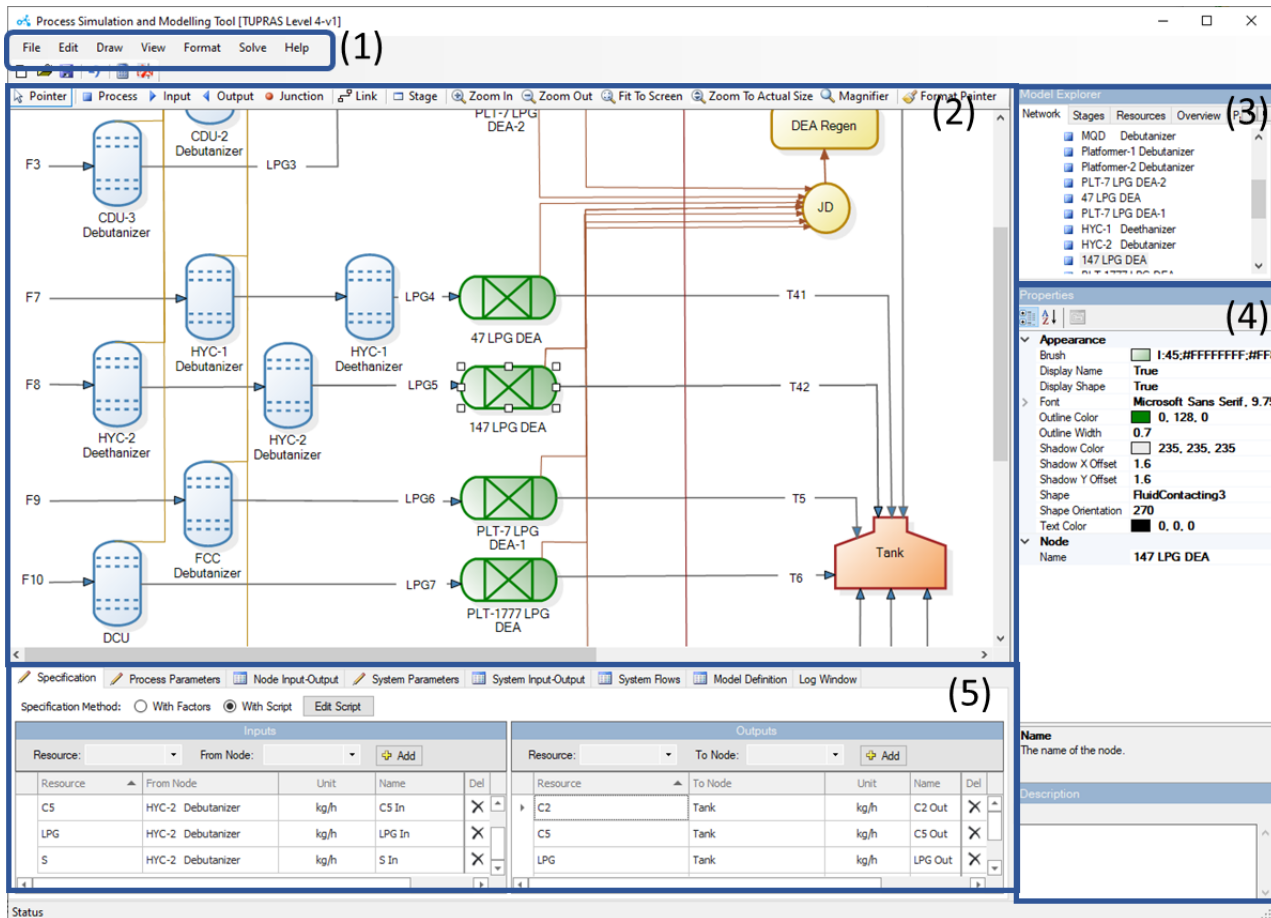
All model entities are organised into a **Hierarchical Inheritance Registry** that provides prototype reconfigurable **building blocks** for building any industrial system model:

- 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**. They are connected with **links** and can be grouped into stages.
- **Processes** represent an activity or task, which is entered by all the required materials (input) and, as a result, generates new or modified materials (output). In this way, processes link material consumption to production. Process specifications can summarize underlying activities in terms of simple relations between input and output flows or algorithms for whose computation programs are used.
- **Places** may be interpreted as stores for resources within the network. They are distinguished as:
  - **Input Nodes**: The initial sources of resources flowing towards processes.
  - **Output Nodes**: The target sinks of resources flowing from processes.
  - **Junctions**: They connect processes, acting as an output node for one process and input node for the other process.
- **Links** 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 can be used in cases where part of a model can be aggregated as an individual unit.

The main **functionalities** of the PSM Tool can be organised in four pillars:

- a) model design, namely the representation of stages and processes taking place in the industries studied in FACTLOG project,
- b) process mapping, which includes the specification of resource flows to and from each process as well as the relations between input/output flows for each resource,
- c) calculation of the resources flow, with the simulation of the production process of the product according to the specified parameters and constraints, and
- d) presentation and reporting of the results at different levels of the model; from the entire system down to the links between the different processes.

The main PSM **user interface elements** (Figure 3) are the following:



**Figure 3: PSM Tool User Interface Elements**

- (1) **Menu Toolbar:** It provides access to the most important functions of the program and consists of seven sub-menus.
- (2) **Graphical Model Editor:** It is the main screen of PSM and allows the user to create, edit and remove model elements. The Graphical Model Editor toolbar provides access to all the model editing operations (inserting new process, input/output node, link, junction or stage).
- (3) **Model Explorer:** It is located in the upper right corner of PSM. The model explorer area is an active window, which basically gathers all the model information. It contains four individual tabs, each serving a distinct purpose:
  - The “Network” tab, presenting the existing elements of the model, categorized by their type.
  - The “Stages” tab, presenting the processes and the input/ output nodes of the model, categorized by stage.
  - The “Resources” tab, presenting all the resources in the model.
  - The “Overview” tab, showing an overview of the system model.
- (4) **Properties Editor:** It presents all the properties of the model element that is selected either in the graphical model editor or in the model explorer.

(5) **Model Specification Area:** It includes three different categories of tabs:

- “Model Specification” tabs
- “Results” tabs and
- “Model Definition” tab.

### 3.3 Discrete Process Modelling

The main tool that will be used for discrete process modelling purposes are Petri nets and specific extensions with given characteristics that improve the modelling power of the initial formalism.

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 [4], [5], [6]. 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). Popular fields of Petri nets include, among others, production and manufacturing systems, project management, computer networks, software development and engineering, traffic monitoring and control, power systems and robotic tasks. However, except the typical engineering applications Petri nets have been used for studying chemical and biochemical processes, medical and healthcare tasks and cognitive, educational and learning procedures.

Ordinary Petri Nets (OPNs) are bipartite directed graphs formally defined as the 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.

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 [2], [7].

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 [5], [6].



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

- **Discrete Places** (  $\bigcirc$  ) describe system states (represent conditions e.g. for control purposes) or resource availability.
- **Discrete Transitions** (  $\blacksquare$  ) represent events or actions that change system states.
- **Tokens** (  $\bullet$  ) 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 (  $\text{---}\text{---}\text{---}\rightarrow$  ). 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 number of tokens at least equal to the weight of the arc connecting place to transition and fire by removing tokens equal to these weights from all the input places and adding tokens to all the output to the transition 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 number of tokens equal to the weight of the respective arc connecting input place with the transition, and “produces” to each one of the output places number of tokens equal to the respective arc connecting transition to these places. An example of transition firing is shown in Figure 4.

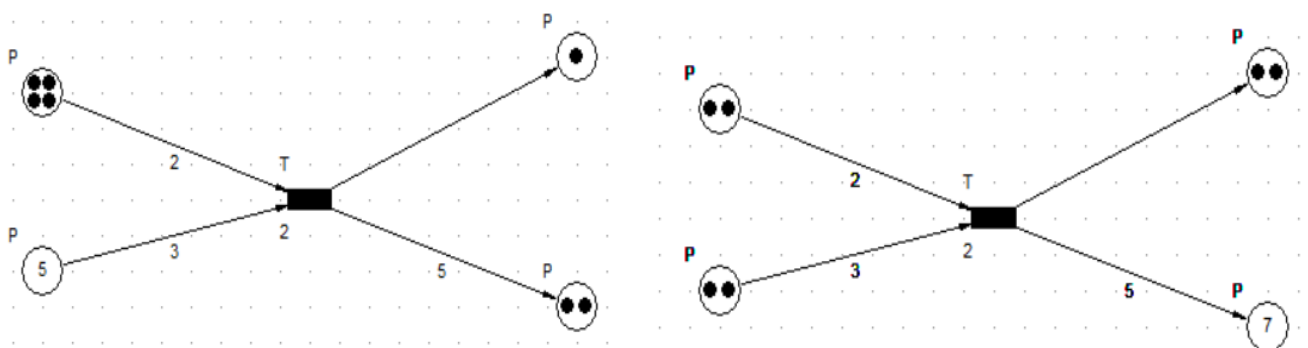


Figure 4: Transition Firing Example

## 4 Process Modelling and Simulation Service API

The previous chapter described in detail the PSM Tool, its structure and functionality and provided detailed information on how a process model can be developed and simulated. However, the development of the process model of an industry is a one-time job. The created model is saved, and it can be accessed at any time without having to re-create it from scratch. The actual process model is dynamic and needs to be updated in situations like when a new process unit is added in the production line or flows between units are added or removed but overall is something that does not occur frequently. On the other hand, simulations are a helpful tool to test various settings and operational scenarios and find e.g., the optimal operating conditions, or create a reschedule plan when a failure occurs. With the help of the PSM Tool and by changing the model parameters it is very easy to recalculate the results and provide insight on scenarios that there is no other way to simulate.

However, up until now we have described the PSM Tool and its capabilities, but we haven't addressed the need for integration with the **Digital Twins Platform of FACTLOG**. As a standalone tool, PSM is not exposing any of its functionality natively. In order to tackle this issue an **Application Programming Interface (API)** has been created. An API is a piece of software intermediary that allows two applications to talk to each other.

### 4.1 Continuous Process Modelling API

In this section, the continuous process modelling API is described, focusing on the key concepts behind that, its services and functionalities as well as the whole lifecycle of the services provided. The main groups of services that the PSM API offers are related to:

- **Model Building**, which is helpful for building the model of the system following the well-defined formal procedure described in the Chapter 2.
- **Parameter Specifying**, which is built for specifying the concrete parameters as well as their types and forms required to define every entity of the system and their interactions from which they will be obtained.
- **Parameter Filling**, which allows the user to define easily all the necessary quantitative parameters of the entities composing the overall model. Such services have also the ability to receive data from other modules (analytics, optimisation, etc.) formulate them according to the needs and fill them in the model.
- **Scenario Simulation**, which is related to the simulation of the defined scenarios.
- **Model Update**, which is related to update the structure of a model with respect to changes in the physical system.

The use of the PSM API is strongly based on the concept of a model instance. A process model is a representation of an industrial production system according to the well-established process modelling methodologies. Flows and process control settings (parameters) are continuously updated, in connection to a real-time monitoring system (sensor network) or by human input maintaining a digital shadow of the physical systems. In case of flows and settings that are not monitored, those are dynamically estimated from monitored flows and parameters, with sufficient accuracy, using the process modelling

algorithms. According to the above, the process model always and accurately represents the current industrial system state in detail. This continuous update might be ideal for monitoring however it can't be used for scenario building, simulation and assessment or optimisation efforts; domains that are an important part of FACTLOG project. On the other hand, **a model instance** is an exact copy of the process model, representing the physical system's state at the moment of instantiation, disconnected from the physical system and it can be modified (change the value of parameters, flows, etc.) without affecting the physical system (in complete isolation from the digital shadow and the physical system).

The services provided by the PSM API are grouped in six (6) categories:

- a) Model Management Services related with the process model management.
- b) Model Instance Services related with the instance management.
- c) System Parameters Services related with the parameters of the system.
- d) Process Parameters Services related with the parameters of processes.
- e) Calculations Services, related to the calculations of parameter values.
- f) Flows Services, related to the flows between processes.

Table 1 provides the formal definition of each service together with the input and output variables is displayed, along with a short description of their provided service.

API Functions	Description of Function
<i>Model Management</i>	
<b>RegisterModel</b> (name, modelSpec)	Adds a new model in the registry. The <i>modelSpec</i> is the .xml representation of the model, as produced by the standalone PSM tool.
<b>UpdateModel</b> (modelID, name, modelSpec)	Updates the specified model.
<b>RemoveModel</b> (modelID)	Removes the specified model from the registry.
<i>Instance Management</i>	
<b>Create Instance</b> (modelID)	Creates a new instance of the specified model. The instance is an isolated exact copy of the model that can be used for scenario simulation, optimisation, etc., without affecting the model (and consequently the physical system).
<b>RemoveInstance</b> (instanceID)	Removes the specified instance from the registry.
<i>System Parameters</i>	
<b>GetParameter</b> (instanceID, symbol)	Returns the current value of the specified model parameter.

API Functions	Description of Function
<b>GetParameters</b> (instanceID)	Returns a collection of all model (system wide) parameters, including their values.
<b>SetParameter</b> (instanceID, symbol, value)	Updates the value to the specified parameter.
<b>SetParameters</b> (instanceID, parameters)	Bulk updates the values of all process parameters.
<i>Process Parameters</i>	
<b>GetProcessParameter</b> (instanceID, process, symbol)	Returns the current value of the specified process parameter.
<b>GetProcessParameters</b> (instanceID, process)	Returns a collection of all parameters of the specified process, including their values.
<b>SetProcessParameterValue</b> (instanceID, process, symbol, value)	Updates the value of the specified process parameter.
<b>SetProcessParametersValues</b> (instanceID, process, parameters)	Bulk updates the values of all parameters of the specified process.
<i>Calculations</i>	
<b>Calculate</b> (instanceID)	Performs a recalculation of the specified model instance. Returns a structure of model results, including all parameters.
<b>CalculateUnit</b> (instanceID, unit)	Performs a local recalculation of the specified process. Returns a structure of the specified process results only.
<i>Flows</i>	
<b>GetFlow</b> (instanceID, fromNode, toNode, resource)	Returns the current value of the specified flow (the flow or "resource" from "fromNode" to "toNode").
<b>SetFlow</b> (instanceID, fromNode, toNode, resource, value)	Updates the value of the specified flow.

Table 1: PSM Tool API Services

The PSM Tool API described above has been implemented in Swagger, which is an Interface Description Language for describing RESTful APIs using JSON. Swagger is used together with a set of open-source tools to design, build, document and use RESTful web services. Some screenshots of the deployed API interface are presented in Figure 5 - Figure 8.

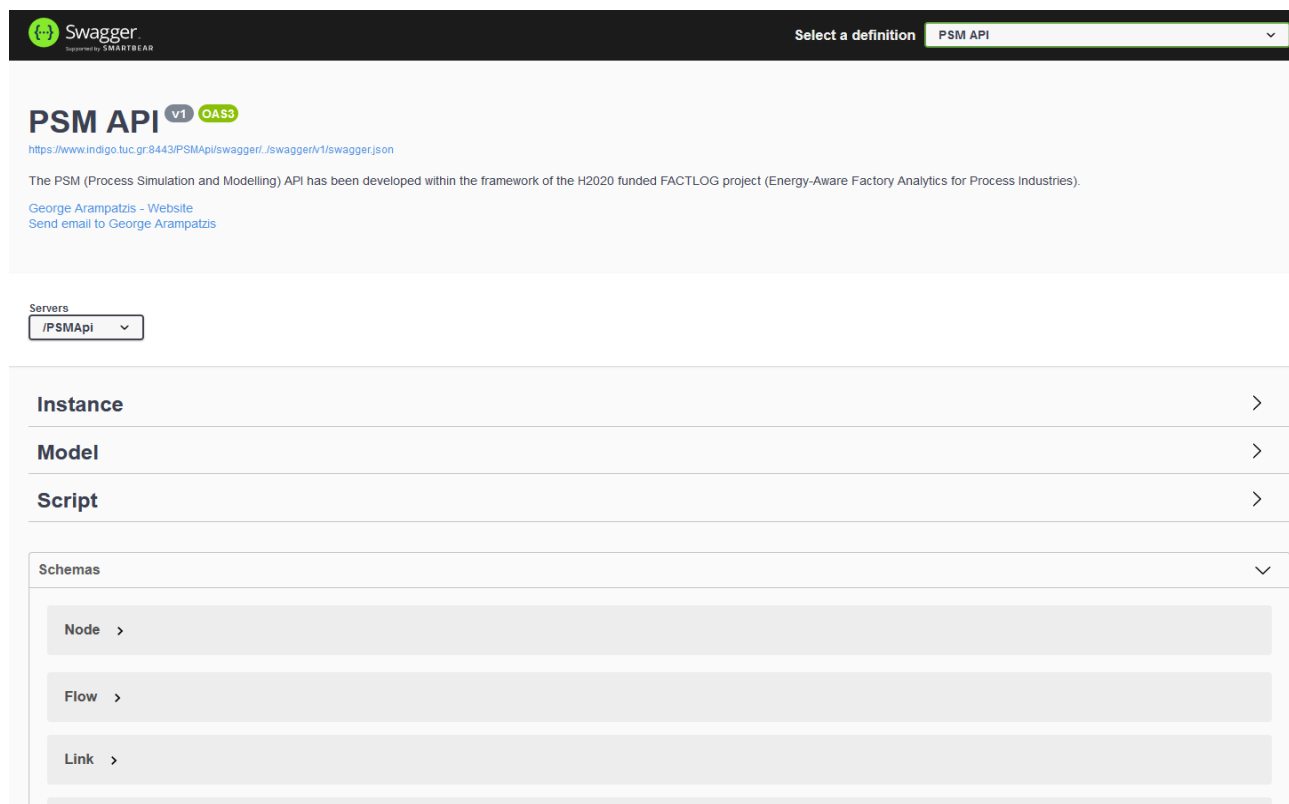


Figure 5: PSM Tool API main interface

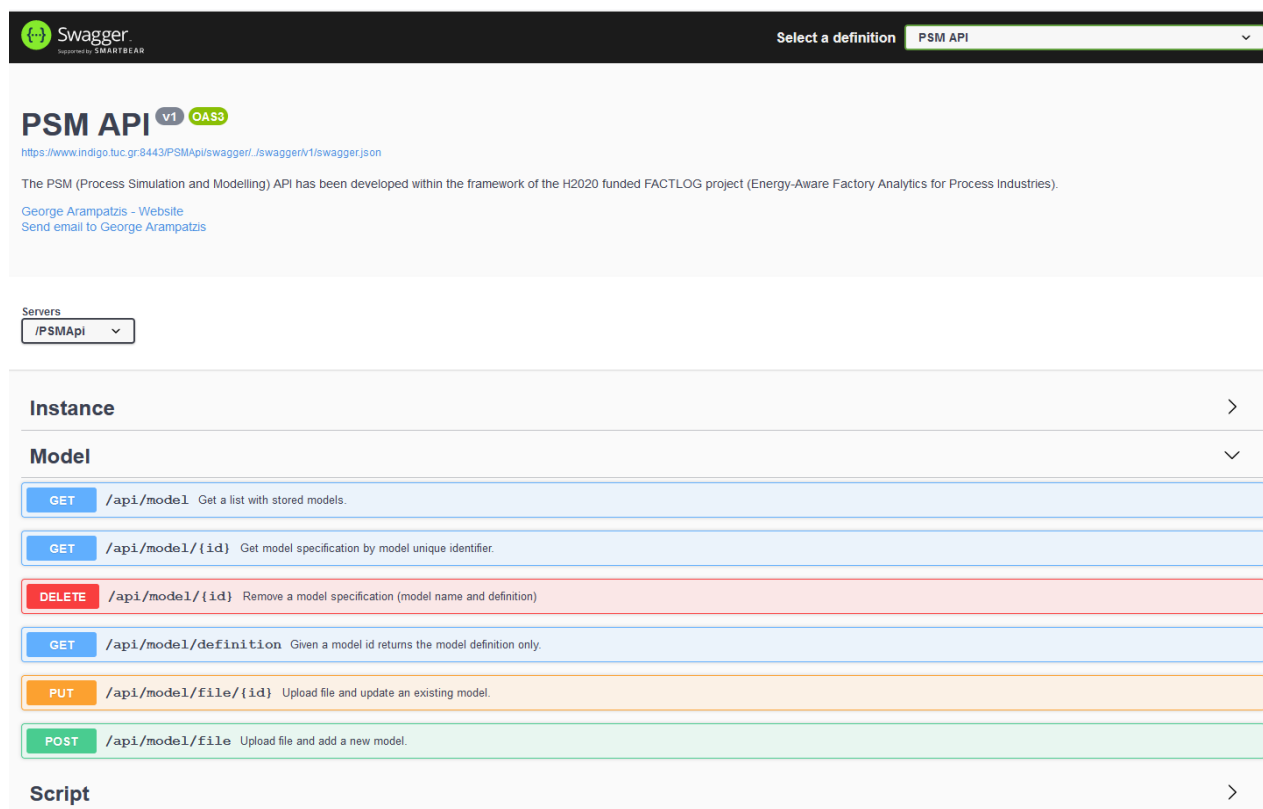



Figure 6: PSM Tool API Services for Models



Swagger  
Powered by SMARTBEAR

Select a definitionPSM API

PSM API

v1

OAS3

<https://www.indigo.tuc.gr:8443/PSMApi/swagger/.jsswagger/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

GET

/api/instance/solve/{id}

Calculate all flows of the specified instance model.

GET

/api/instance/{id}

Get a model instance specification by specifying the model instance unique identifier.

PUT

/api/instance/{id}

Update an existing instance model.

DELETE

/api/instance/{id}

Remove an instance using its unique identifier.

GET

/api/instance/definition

Given an instance ID, returns the model instance definition.

POST

/api/instance/{modelId}

Create a new instance from an existing model.

GET

/api/instance/model-parameters

Get all model parameters by specifying an instance unique identifier.

GET

/api/instance/model-parameter

Get a model parameter by specifying an instance unique identifier and parameter name.

Figure 7: PSM Tool API Services for Instances

Schemas

Node

description

Parent entity from which process, junction, input, output nodes inherit

stageId

string

nullable: true

solved

boolean

id

string

nullable: true

name

string

nullable: true

description

string

nullable: true

Flow

resourceId

string

nullable: true

name

string

nullable: true

quantity

number(\$double)

nullable: true

manual

boolean

calculated

boolean

formula

string

nullable: true

factor

number(\$double)

nullable: true

Link

sourceId

string

nullable: true

targetId

string

nullable: true

flows

> [...]

id

string

nullable: true

name

string

nullable: true

description

string

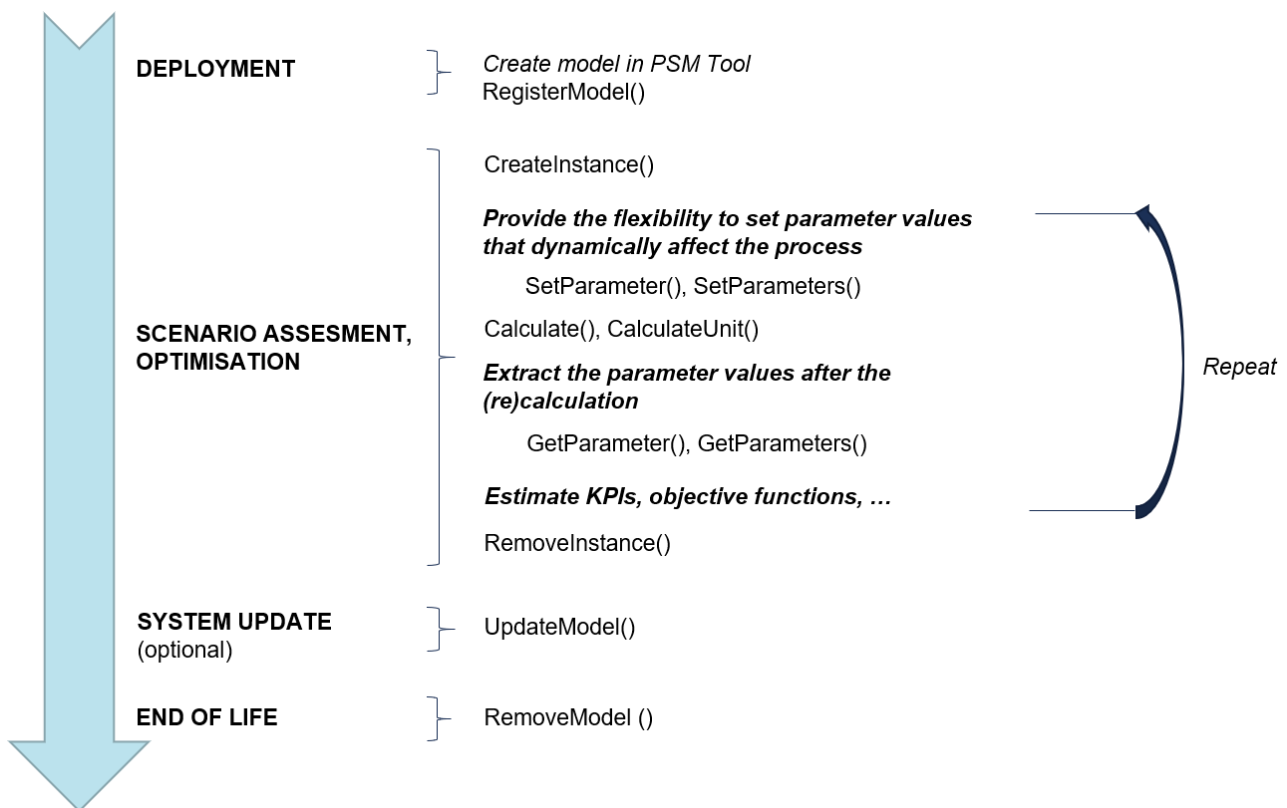
nullable: true

Stage

>

Figure 8: PSM Tool API schemas

The actual value of the PSM API is illustrated by presenting the **life-cycle of a model** in the context of FACTLOG (Figure 9). Initially, the model needs to be created with the PSM stand-alone Tool. Having the model ready, with the help of model management services it is registered in the system providing as inputs the name and the xml representation of the model as produced by the PSM stand-alone Tool. The next step is to grab an instance at the specific time frame important for the specification assesment or optimisation effort using the CreateInstance() function. The API provides the flexibility to set parameter values that dynamically affect the process by using SetParameter() / SetParameters(). When the values are set we are able to call Calculate() / CalculateUnit() (depending if we are looking for a specific unit or the whole model values) to run the simulation and then call GetParameter() / GetParameters() to read 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. Finally, when we are done with our calculations regarding the specific instance we can call RemoveInstance() to discard it. In case the registered model needs to be updated we can call the UpdateModel() function however that is not going to update any instances already created before the update but only the model. Finally, when we have completed every simulation and experimentation and we don't require the model any more we can call RemoveModel() and completely remove the model from the system marking the end of the life-cycle we have been discussing. In the following figure this life-cycle described above is being depicted.



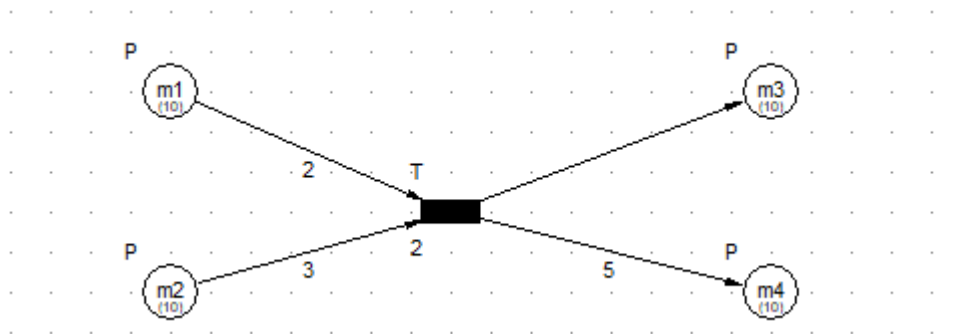
**Figure 9: Life-Cycle of the API use**



## 4.2 Discrete Process Modelling

At the moment, the tool being used for the creation and simulation of Petri Nets is **Visual Object Net**, a standalone tool that makes the design and simulation of such networks quite easy following a similar fashion as the PSM tool with the use of a graphical interface. Using the interface, one can create the required network and then run simulations and visualize the results. Even though this is adequate for the time being and the creation of the process models, in the context of FACTLOG, communication with the rest of the platform is required. Unfortunately, Visual Object Net is not a custom-made software like the PSM Tool and accessing its backend functionality is not possible. However, the created network can be exported as a text file and transferred as a whole.

Based on this notion, we have developed a Python application that takes as input the Petri Net developed in the Visual Object Net and simulates the network similarly to the existing tool. This Python software has been built upon SNAKES library [8], [9] a flexible High-Level Petri Nets library which has been adapted to take as input a text description of the Petri Net and simulate the outputs. The software provides a step-by-step description of each transition and the exact time of the firing, the initial and the final marking (number of tokens) as well as the total simulation time until no more firings are able to happen. In addition, the tool creates graphs of the network both initial and final as well as the description of the network in Petri Net Mark-up Language an XML-based format for standardized description of Petri Nets. Following, a quite simple example of a Petri Net is displayed. Figure 10 provides an example of the implementation of a network in Visual Object Net.



*Figure 10: Basic Example of a Timed Petri Net*

In this example, we have 4 places (2 input and 2 output) and 1 transition that is triggered every 2 time units. The initial marking in these 4 places are 4,5,1,2 tokens respectively. The transition is fired after 2 time units. After the transition, the final marking in the 4 places are 2,2,2,7 tokens respectively. No more transitions can be fired as the tokens remaining in the input places are not enough for a second transfer. The output of the Python software is depicted in Figure 11, while the input and output graphs are displayed in Figure 12. The curly brackets are displaying the number of tokens while the number on the arrows the transfer of tokens taking place when a transition is fired.

```

m0 = { 4, 5, 1, 2 }

Transition t1 fired at time 2.0
Simulation took 2.0 time steps to complete.

mf = { 2, 2, 2, 7 }

```

Figure 11: Output of Python software

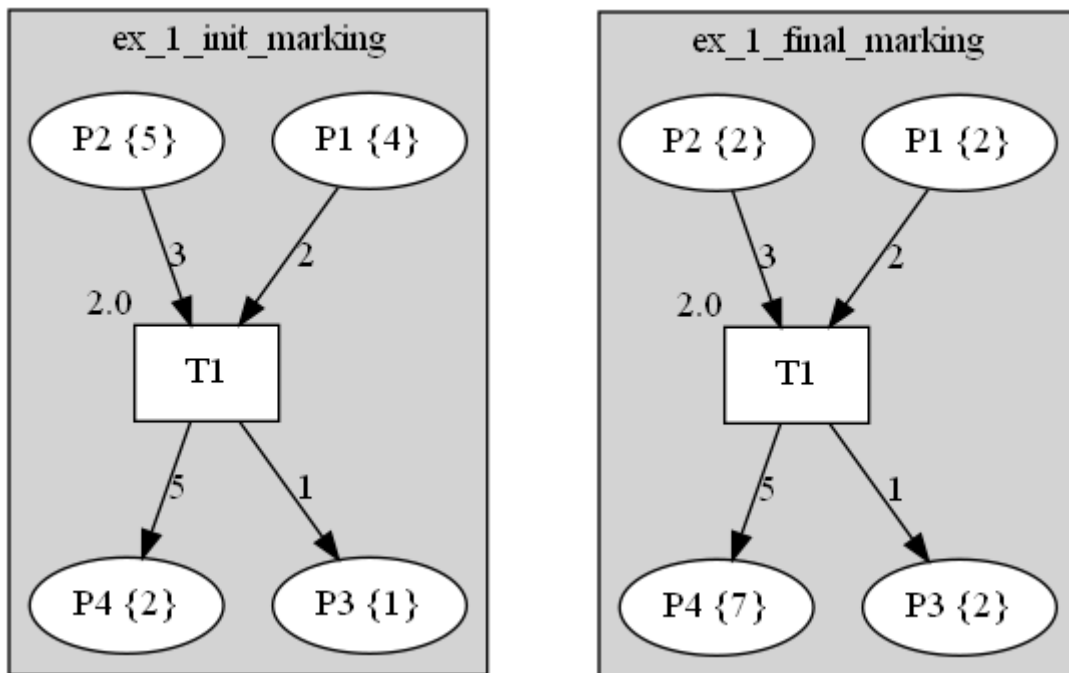


Figure 12: Initial and final graphs of a Petri Net

The ultimate goal is, with the help of the developed Python software, to add the option of building models for discrete processes with the help of the PSM Tool that currently is only capable of building models for continuous processes. In a similar fashion, an API that will expose information, such as the total time steps of the simulation, the initial and final marking and the fired transitions information already calculated, is being currently designed. Those additions are planned to be implemented within the FACTLOG project and are scheduled to be completed within Deliverable D4.3 due on month 32. We consider important to point out here that this library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

## 5 Process Models in FACTLOG Pilot Cases

There are 5 distinct pilot cases in FACTLOG project. These are meticulously selected to reflect the application of the developed innovations in different conditions and prove the diversity and scalability of the developed platform in process industries with different needs. These business cases are distinguished in continuous and discrete according to the nature of the process taking place in each industry.

- JEMS and TUPRAS pilots are the continuous cases, while
- BRC, Continental and Piacenza represent the discrete cases.

The description and the explanation of the developed process models per pilot case, according to the principles described in chapters 2 and 3, is provided in the following paragraphs.

### 5.1 Continuous Processes Pilot Cases

#### 5.1.1 Oil Refineries: Pilot Case by TUPRAS

##### 5.1.1.1 Introduction

TUPRAS refinery, located in Izmit, Turkey, produces various petroleum products such as LPG (Liquefied Petroleum Gas), gasoline, diesel and naphtha. The refinery is composed of multiple units, each one serving a specific role in the production process (e.g., production of LPG, production of gasoline, production of diesel, purification of products). FACTLOG project focuses on the LPG purification unit, i.e., on the various processes that need to be applied to turn the LPG production streams to LPG refined streams in order to meet specific quality criteria.

**Liquefied Petroleum Gas (LPG)** is a mixture of hydrocarbon gases. Varieties of LPG include mixes that are mostly propane ( $C_3H_8$ ) and butane ( $C_4H_{10}$ ). Propylene, butylene, and various other hydrocarbons are usually also present in small concentrations. LPG in TUPRAS refinery is produced with the following 6 types of process units:

1. Atmospheric or Crude Distillation Unit (CDU)
2. Hydrocracker (HYC)
3. Fluid Catalytic Cracking (FCC)
4. Delayed Coker Unit (DCU)
5. Maximum Quality Diesel (MQD)
6. Platformer

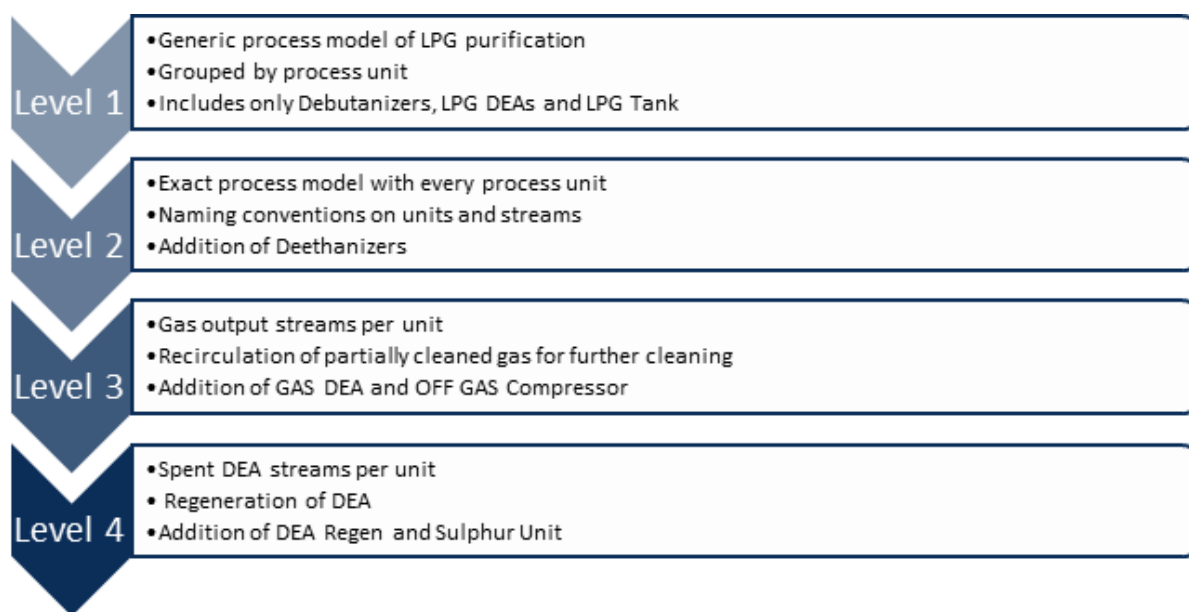
After the production, some impurities remain in the LPG. Such impurities render the product unsuitable for the market and need to be removed (LPG purification process) in order for the LPG to adhere to the standards and regulations of the petroleum market. These specs are summarized in Table 2.

LPG Specifications	Total Impurities	Vapour Pressure	C <sub>5</sub>	C <sub>2</sub>	Sulphur
	5% mol/mol	1430 kPa	2% mol/mol	0.5 mol/mol	50 mg/kg

*Table 2: LPG Specifications that the final product must meet*

#### 5.1.1.2 LPG Purification Process Modelling

The focus of FACTLOG project in TUPRAS pilot is the purification process of LPG production that will let the product be on-specs and ready for the market. Purification process takes the LPG through various steps of refining and uses additional organic compounds such as diethanolamine (DEA). In order to be easily comprehensible, the modelling of those complicated processes has been implemented in **4 Levels of Detail**, each one adding more information to the previous. Those 4 levels have been developed not only for easier comprehension among project partners of the purification process but also due to the fact that different modules require different levels of abstraction, e.g. optimisation is utilizing Level 2 and Level 3 schematics since Level 4 is adding nothing but complexity to their efforts. Figure 13 is depicting the flow from Level 1 to Level 4 and the additional information in every process model.



*Figure 13: Process Model progressive level of detail*

Next, we provide the process models for each level of detail with colour coding for each different stream and accompanied by a short description. To this end, TUPRAS indicated that there are ten (10) LPG raw streams (F1 – F10), eight (8) LPG streams which are the outputs of the debutanizer/deethanizer (LPG 1 – LPG8) and seven (7) streams which are the inputs to the LPG tank (T1 – T7) which can be seen on Level 2 process model.

#### 5.1.1.2.1 *Level 1*

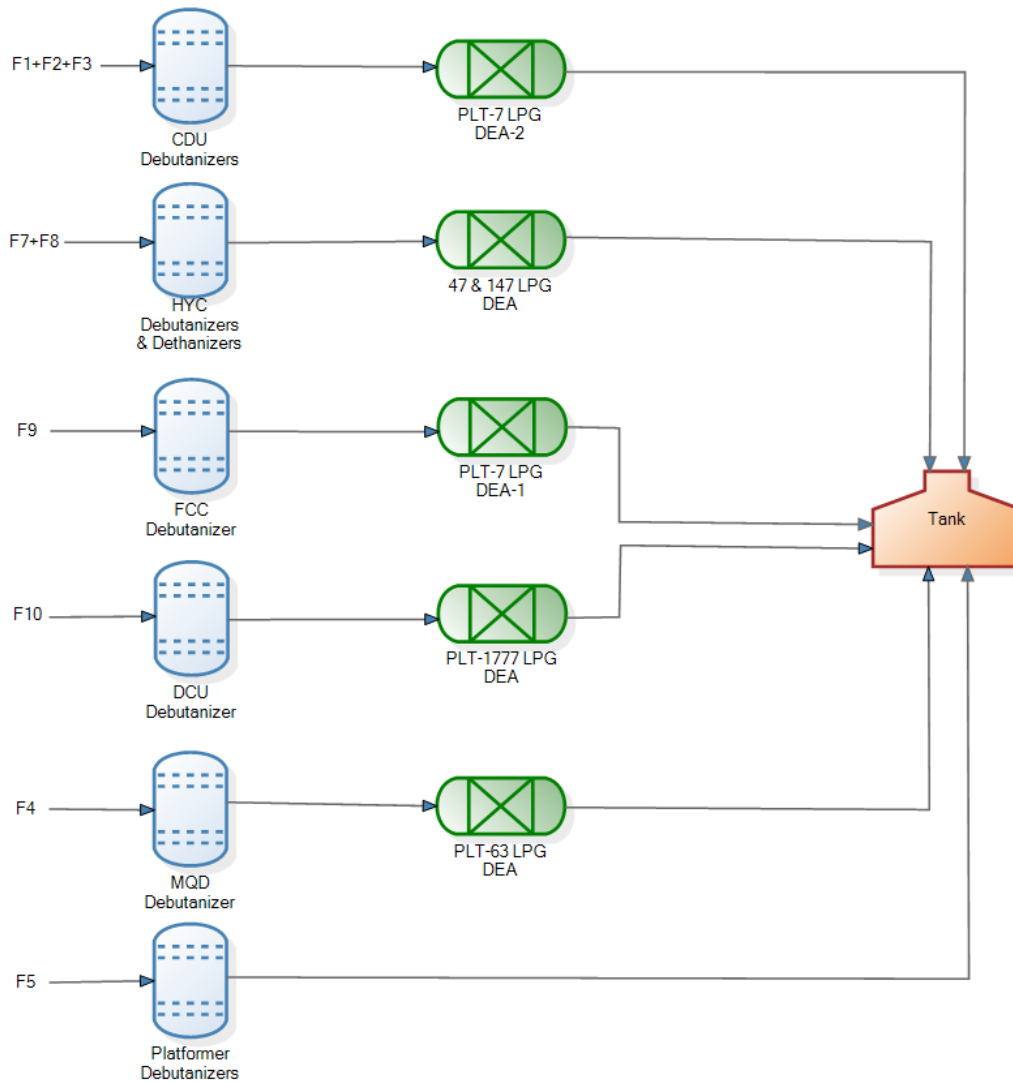
Level 1 serves as the basis of the process modelling. The 6 different units involved in the production of LPG are presented here. Specifically, the Debutanizer / Deethanizer columns that are integrated in those units, since these are responsible for the LPG purification process we are modelling.

A Debutanizer / Deethanizer is a type of fractional continuously operated distillation column used to separate butane / ethane from a mixed feed of hydrocarbons during the refining process. The output of those units, which is “partially” clean LPG (butane / ethane removed) is run through some LPG DEA units and end up to a tank that cumulative collects the LPG. DEA units are a treating system used to remove hydrogen sulphide ( $H_2S$ ) and carbonyl sulphide from gas streams such as the LPG stream in our case.

In Level 1 for abstraction reasons, we have combined the same units into a single module; e.g. there are 3 CDU Units in TUPRAS plant but here they are considered as one. In addition, the HYC’s Debutanizer and Deethanizer units have been grouped in this level of abstraction.

A simplification applied in Level 1, but remains in all process models created up to Level 4 is the number of tanks. Even though it is not just one on a refinery like TUPRAS, we have been informed that they are not feeding LPG to more than one tank at the same time; meaning that they are waiting for one to be filled before switching to the next. Based on this note we decided to add only one Tank in our process models since it is representing the one currently in use. This does not mean that tank capacity remains the same when switching to a new one, but this is a parameter that can be easily modified without affecting the design procedure of the actual model.

Regarding the naming of the streams in Level 1 we distinguish only the input streams ranging from F1 to F10 and the naming of LPG DEA units.

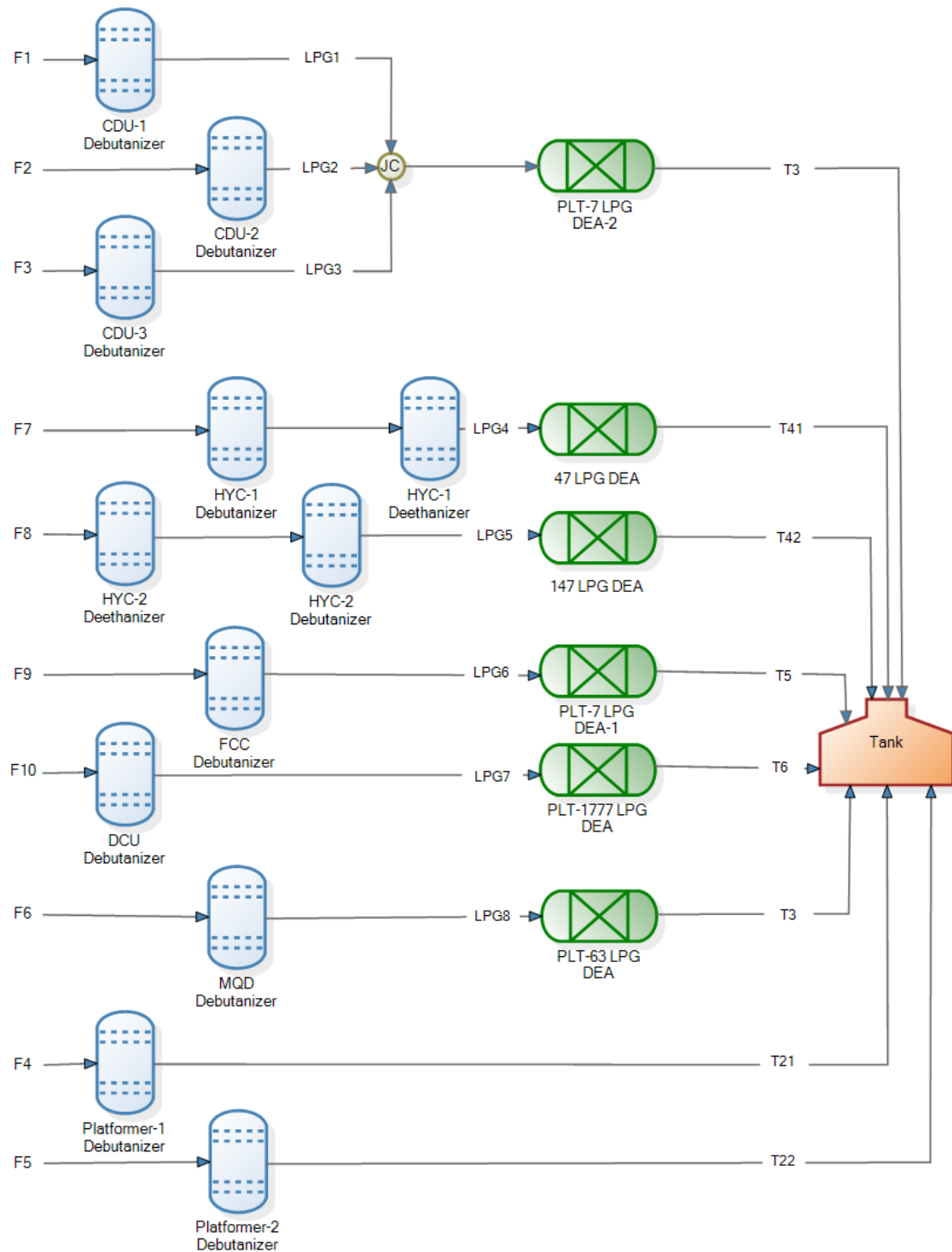


**Figure 14: Level 1 process model of TUPRAS LPG purification plant**

#### 5.1.1.2.2 Level 2

Level 2 process model is giving additional information on the exact units in the LPG purification process in TUPRAS. There are 3 CDU Debutanizer units and 2 Platformer units whose outputs are mixed respectively. In addition, on the HYCs' which include both a debutanizer and a deethanizer, in Level 2 process model they are displayed as separate units.

We can identify the 10 input streams ranging from F1 to F10 in a one-to-one relation; meaning that one stream enters one specific unit and not like Level 1, the partially cleaned LPG. Also in Level 2 we can see a junction of streams (the JC symbol) where 3 streams (LPG1, LPG2 and LPG3) are mixed before entering the corresponding DEA unit.



**Figure 15: Level 2 process model of TUPRAS LPG purification plant**

#### 5.1.1.2.3 Level 3

Level 3 process model is building upon Level 2 and adds information regarding Sour Gas, a gas containing  $H_2S$  that is the top product of the debutanization / deethanization distillation process. This gas even though it is a by-product it contains quantities of hydrocarbons that can be captured, compressed, and re-purified to gather additional quantities of clean LPG.



A specific DEA unit, namely the GAS DEA unit, receives every debutanizer / deethanizer sour gas output. The acid gases are absorbed by the diethanolamine (DEA), and sweet gas leaves at the top of the absorber. Finally, this gas is compressed again by the OFF-GAS compressor into liquid and returns to 3 of the 10 LPG purification units (Platformer Debutanizer 2, MQD Debutanizer and HYC Deethanizer 1) that handle the re-purification of this LPG. From this level on colour coding is used in the streams with yellow depicting the sour gas (Gas + H<sub>2</sub>S + CO<sub>2</sub>), red the sweet gas (after the GAS DEA absorption) and grey the compressed returning gas.

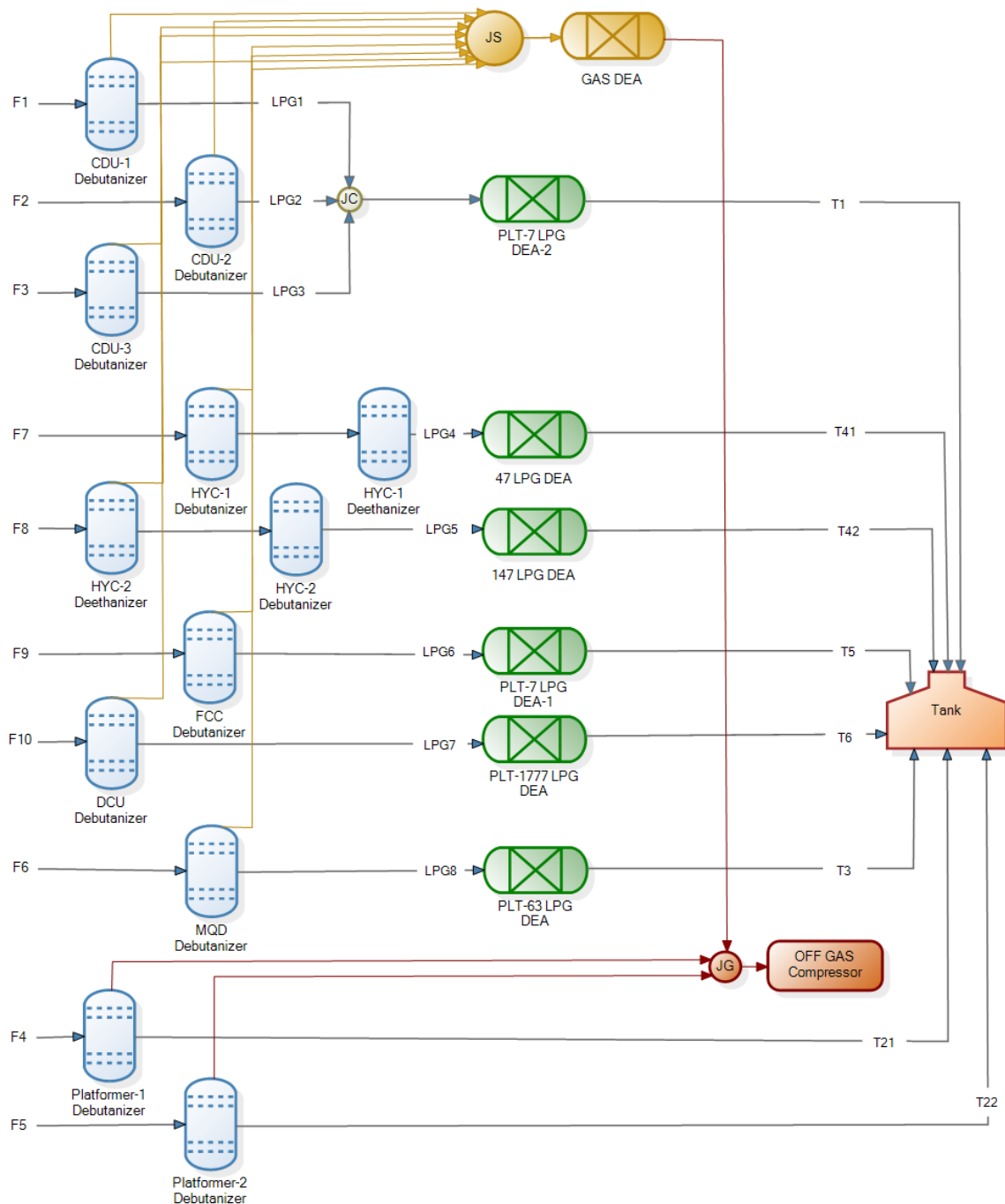


Figure 16: Level 3 process model of TUPRAS LPG purification plant

#### 5.1.1.2.4 Level 4

Finally, on Level 4 details regarding the functionality of the DEA units are added. As mentioned earlier, DEA units are using diethanolamine organic compounds to retain  $H_2S$  and  $CO_2$  from the LPG stream. This amine is deteriorating and loses its absorption capabilities over time. Since DEA has the ability to be regenerated and reused instead of being replaced, a DEA Regeneration unit is present in the installation. This unit with the use of steam (intra-factory reuse) is cleaning the diethanolamine in order to be reused in the DEA units involved in the purification process. The retained  $H_2S$  from the DEA is finally fed in a Sulphur Unit and this by-product of the purification process will be used in another part of the refinery or to produce another product.

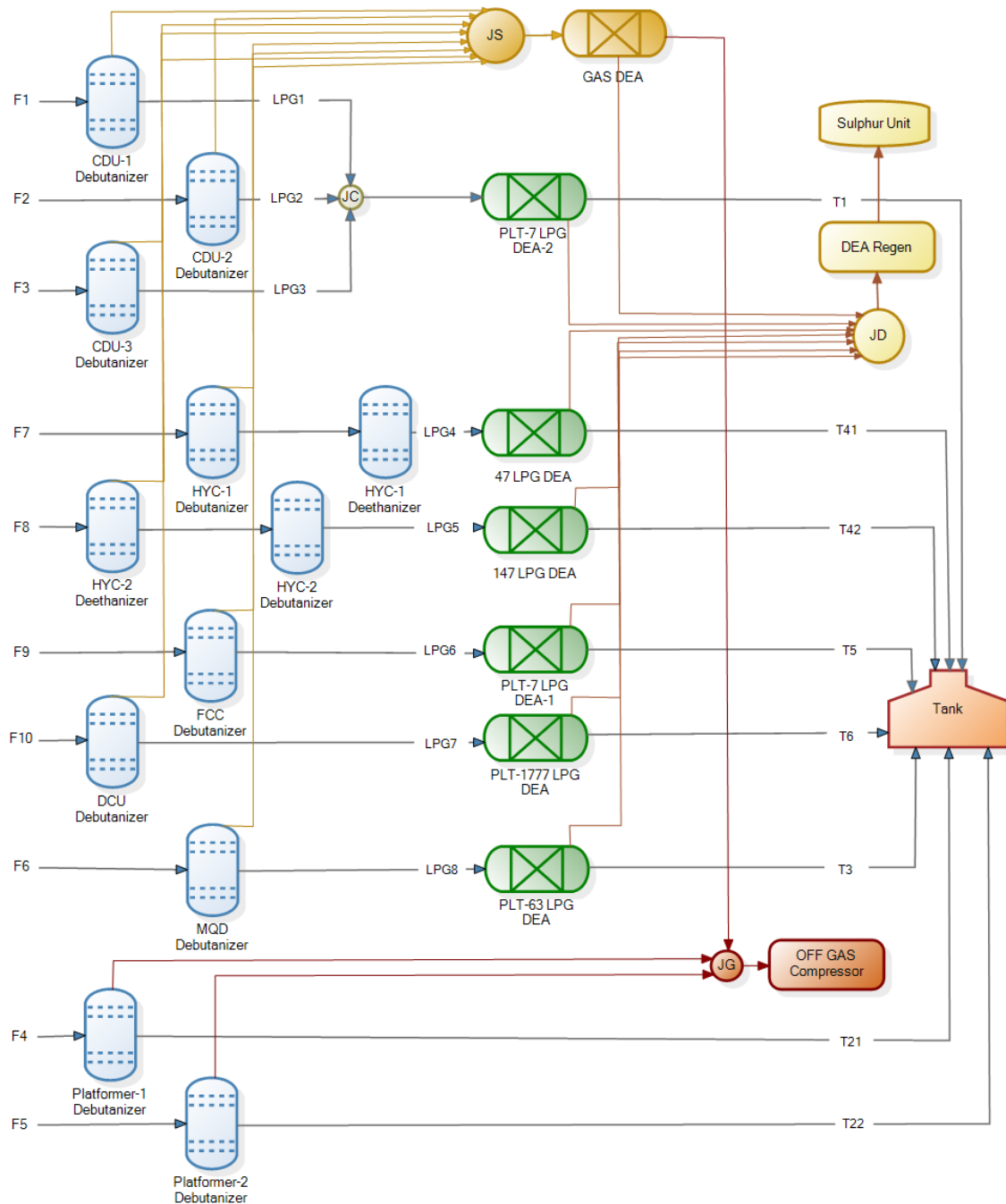


Figure 17: Level 4 process model of TUPRAS LPG purification plant

### 5.1.1.3 *LPG Purification Units*

#### 5.1.1.3.1 *Debutanizer Units / Deethanizer Units*

The operation of the debutanizer/deethanizer columns refer to a simple distillation column that separates components based on their boiling point. This distillation process aims to separate liquid components by heating a liquid to vapour, to later condense the vapour back to liquid in order to purify or separate it. Hence, the heaviest components (containing C<sub>5</sub>) will leave from the bottom of the column, whereas lighter components will leave from the top (S and LPG components).

The overall goal of the debutanizer columns is to remove (or adjust) the C<sub>2</sub> and C<sub>5</sub> content from the input streams. Similarly, the deethanizer is designed to take a liquid as input and remove ethane and lighter components from the liquid.

In total there are 10 debutanizer columns in TUPRAS refinery feeding from streams F1 to F10. Each unit involved in the LPG production has its dedicated debutanizer column. The deethanizers are only 2, both in the HYCs but installed in different configuration. In HYC-1 stream F7 enters the debutanizer and then the deethanizer column where in HYC-2 stream F8 is directed first to the deethanizer and then to the debutanizer. All those units have two products (important for our process modelling), a sour gas output and an LPG output. Neither of these outputs are a final product since they are fed into the DEA Units for further treatment. Platformer-1 and Platformer-2 are the only exception with their LPG output being fed directly to the LPG concentration tanks and their gas output to the OFF-GAS compressor without any further treatment.

#### 5.1.1.3.2 *LPG-DEA Units / GAS-DEA Unit / DEA Regen Unit*

DEA units are performing a process called amine treating, which is a group of processes that use aqueous solutions of various alkylamines (or simply amines) to remove hydrogen sulphide (H<sub>2</sub>S) and carbon dioxide (CO<sub>2</sub>). Processes within oil refineries that remove hydrogen sulphide are often referred to as "sweetening" processes because the odour of the processed products is significantly improved by the absence of H<sub>2</sub>S.

A typical amine treating process includes an absorber unit (LPG-DEAs / GAS-DEA) and a regeneration unit as well as accessory equipment. In the absorber, the amine solution absorbs H<sub>2</sub>S and CO<sub>2</sub> to produce a sweetened stream as a product and an amine solution rich in the absorbed products. The resultant "rich" amine is then routed into the regenerator (a stripper with a reboiler) to produce regenerated or "lean" amine that is recycled for reuse in the absorber.

There are in total 6 LPG-DEA units operating towards reducing (or totally removing) S-components from LPG in TUPRAS plant. Each LPG DEA unit receives LPG streams from the one (or more) of the 10 debutanizer columns. Specifically, we can see that the 3 CDU Debutanizers are feeding the same LPG DEA unit (PLT-7 LPG DEA-2), HYC 1 Debutanizer/Deethanizer feeds 47 LPG DEA, HYC 2 Deethanizer/Debutanizer feeds 147 LPG DEA, FCC Debutanizer feeds PLT-7 LPG DEA-1, DCU Debutanizer feeds PLT-177 LPG DEA and MQD Debutanizer feeds PLT-63 LPG DEA. There are also 2 Debutanizer streams; namely Platformer 1 Debutanizer and Platformer 2 Debutanizer that don't run through an LPG DEA. A single GAS-DEA unit receives and treats all the Debutanizer/Deethanizer sour gas outputs (S1 to S10) is present in the TUPRAS refinery.

The working principle described earlier applies to GAS-DEA also, but instead of liquid the feed is gas. The spent DEA both from the LPG-DEAs and the GAS-DEA is routed to the DEA Regen unit, to separate it from S-components via heating with steam. The chemical reaction in the regenerator reverts the amine sulphide and amine carbonate to amine and acid gas (a gas mixture containing significant quantities of  $\text{H}_2\text{S}$  and  $\text{CO}_2$  or similar acidic gases) when heat and stripping steam are supplied. The percentage of amine sulphide & carbonate that convert to acid gas and amine depends upon the amount of heat and stripping steam applied to the regenerator. The product of the GAS Regen unit is fed into the Sulphur unit which utilizes the feedstock of acid gases. Those gases are sent into a proprietary burner system, resulting in a mixture of hydrogen sulphide and sulphur dioxide that form elemental sulphur which is then removed through condensation.

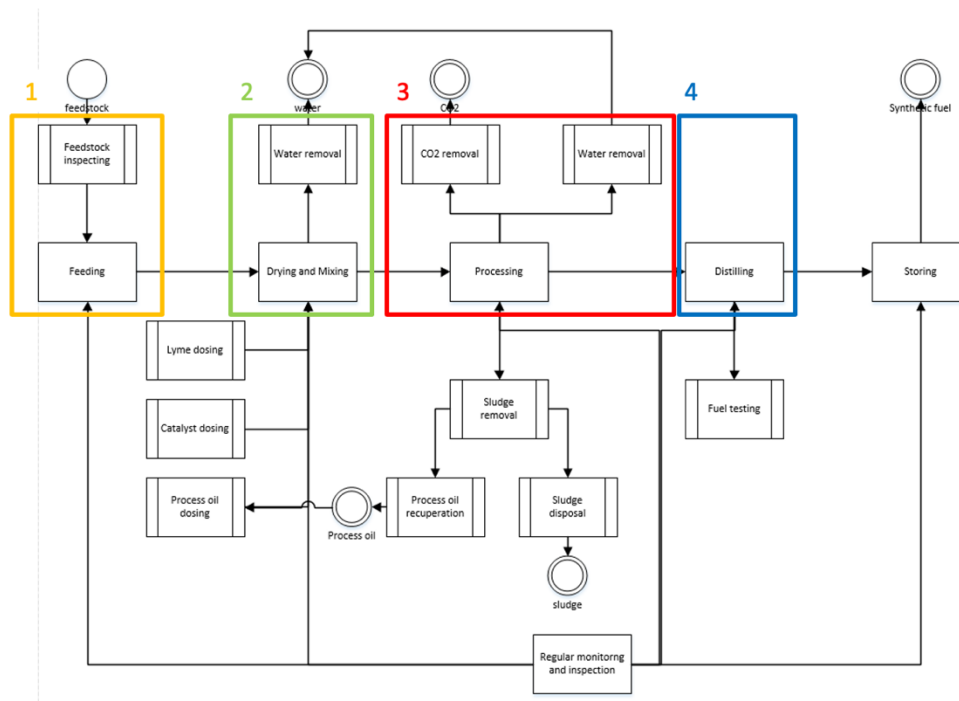
#### 5.1.1.3.3 OFF GAS Compressor

The gas that has been produced by the GAS DEA unit (processed sour gas) and the Platformers' Debutanizer contains some concentration of LPG. This cannot be considered as a by-product and thrown away but is also not on-specs to be redirected into the LPG concentration tank. Therefore, it is compressed into a liquid, in the OFF GAS compressor, and sent again through the purification process to detain as much LPG as possible. In this unit, Platformer 1, Platformer 2 and Gas DEA outputs are entering, compressed and recycled back to the debutanizers in order to retrieve LPG that has "escaped" through the sour gas from the debutanizer / deethanizer distillation columns and the GAS DEA unit. The OFF GAS compressor returns LPG in three units, HYC-1 Deethanizer, MQD Debutanizer and Platformer 2 Debutanizer and not to every unit in the treatment plant.

#### 5.1.2 Waste-to-Fuel Transformer Plants: Pilot Case by JEMS

The JEMS pilot presents a specific challenge for process modelling. The details of the methods and processes used in the JEMS plant to process organic waste into fuel is proprietary and confidential. Since this information is necessary for construction of a process model based on first principles, a different method is needed. A bottom-up data-driven analytics approach was chosen since sensor data from the JEMS plant was readily available.

The StreamStory tool was used to model the processes in JEMS. StreamStory is a tool for analysis and modelling of multivariate time streams. It identifies the typical states of the observed system and builds a Markov chain model of these states. This way the typical transition of the system between the states is captured. The state model is hierarchical, allowing observation and inspection at different levels of detail. The states are described by the distributions of the variables in their corresponding states. The model allows us to both assign new states to the states, identify potential anomalies (i.e. outliers with respect to model states) as well as simulate the system using Monte Carlo simulation. More details on the technical aspects of use of StreamStory in the JEMS pilot can be found in deliverable D7.1 [10] and more details on the tool itself is available in D2.4.



**Figure 18: JEMS plant pipeline split into the four main stages of the process**

To gain insights into the processes inside the JEMS plant the plant pipeline was conceptually split into successive stages and then each stage was modelled using the StreamStory. The stages are shown in Figure 18 and are:

1. Feedstock inspecting and Feeding
2. Drying and Mixing
3. Processing
4. Distilling

The first stage does not have any sensors corresponding to it so only stages 2-4 were modelled. The other boxes presented in Figure 18 represent supporting subsystems, the influence of which is represented in the stages indirectly through the sensor readings. For each stage a model was built such as the one in Figure 19. The model graphs were inspected by JEMS plant experts, which have confirmed that the model reasonably represents the behaviour of the plant. There are some surprising nodes that do not correspond to any expected behaviour and these are being inspected as possible indicators of process anomalies as part of ongoing work. Better understanding of these nodes could lead to raising anomaly detection alerts when they are detected or predicted.



Figure 19: Stream Story model for stage 1 - Drying and Mixing

## 5.2 Discrete Pilot Cases

### 5.2.1 Steel Manufacturing: Pilot Case by BRC

#### 5.2.1.1 Introduction

BRC Company was founded in 1908; it is the UK's largest supplier of steel reinforcement and associated products for concrete. BRC fabricates cut & bent rebar to the specs of BS8666:2005 and governed by the independent steel reinforcement governing body C.A.R.E.S. In 2009 BRC was acquired by the Celsa Steel Services UK group and currently has 4 depots in the UK with the largest being in Newport South Wales which can produce up to 2000 tonnes of fabricated reinforcement for the construction industry per week. The rest are located in Romsey near Southampton, Mansfield in the midlands and Newhouse up in Scotland.

#### 5.2.1.2 Bay 3 Description

The BRC factory consists of 3 Bays with respect to the part types processed in each and certain features of the machines (automation, types etc.). In the case of BRC, only Bay 3 is considered in the FACTLOG project. The processes performed in this bay involve cutting and shaping various diameters of steel reinforcing bar using various manual or automatic operations. 3 types (or families) of final products can be produced: Coils, Bars and Bent Bars of different diameters and with different numbers of bents. Bending especially makes possible the production of certain shapes – everything from simple straight bar to complex 3D shapes (BRC has certain catalogues of standard shapes but the customer can even ask

for any other geometry as well). 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, 4 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., speeds).

Each machine performs strictly processes of one stage and setup is necessary before changing the operation of a machine. Moving equipment (cranes) is used for transportation loading and unloading of parts and products between machines and buffers (but no time data connected to this operation are available at the moment). The described production process is presented in Figure 20 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 the specifications of the customers products with given characteristics concerning, size, geometry, weight, type of raw material used etc. can be produced using the resources of Bay 3.

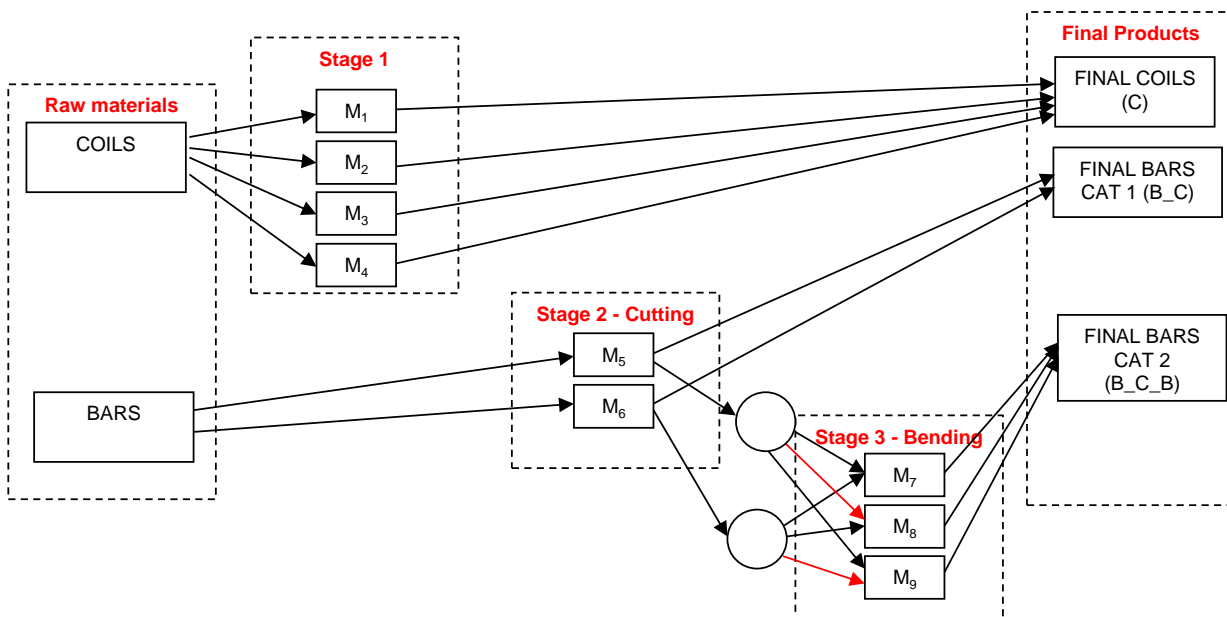


Figure 20: Production process in Bay 3

### 5.2.1.3 Bay 3 Process Modelling

As already discussed the implemented models are modular in all discrete pilot cases including BRC. In this the main entities that have to be considered initially are the machines of Bay 3 that process parts according to the specifications provided. In addition, the jobs under process must be considered and modelled. The orders consist of jobs that refer to batches of products with similar characteristics.

In this part the main assumptions regarding process modelling and simulation in the case of BRC are described. According to this, before running Optimisation and PSM, the time horizon of the procedure, 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 have to be known. When a job is scheduled, no pre-emption / interruption is allowed

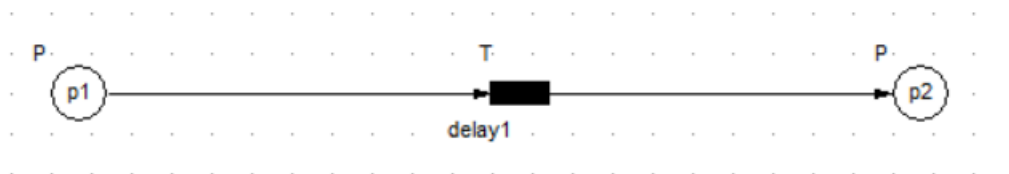


and no job can be processed by more than one machine at the same time (i.e., job splitting is not allowed). In addition, machines cannot process multiple jobs at the same time and if a product is flagged as finished, then it cannot be processed again. Finally, raw material availability in all cases has been considered infinite, as a basic assumption.

The Process Modelling and Simulation module produces a Timed Petri net model of the structure of the production system, as well as the data concerning the schedule produced from optimisation module for the given time horizon of the jobs performed in Bay 3. For this reason, the structure of the model is not static, as the number of jobs considered (with respect to the given time horizon) and their routing through the system is defined in previous steps. However, the modelling approach followed is modular and specific Petri net structures (subnets) are repeated and appropriately connected, in order to form the model of the overall system with respect to the scenario considered for study. These initial models are initially presented and explained in this part as well as the way they are connected in order to define interactions between the system components.

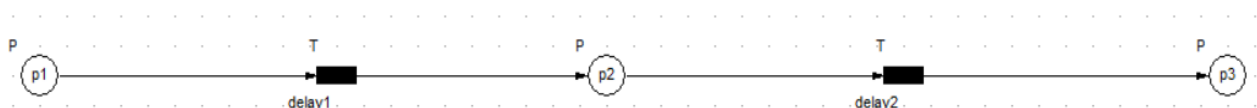
#### 5.2.1.3.1 *Model of a process*

The simplest case of the Timed Petri net model of a process is shown in Figure 21. According to this the model consists of two places and a transition between them. The initial place (p1) refers to the available unprocessed parts (resources) while the final one (p2) refers to products after the performance of the process. Transition T with delay equal to a value represented as delay1 describes the performance of the process in a part. If multiplicities of parts or products are not unitary, then weights can be added on the input or output arc of the transition.



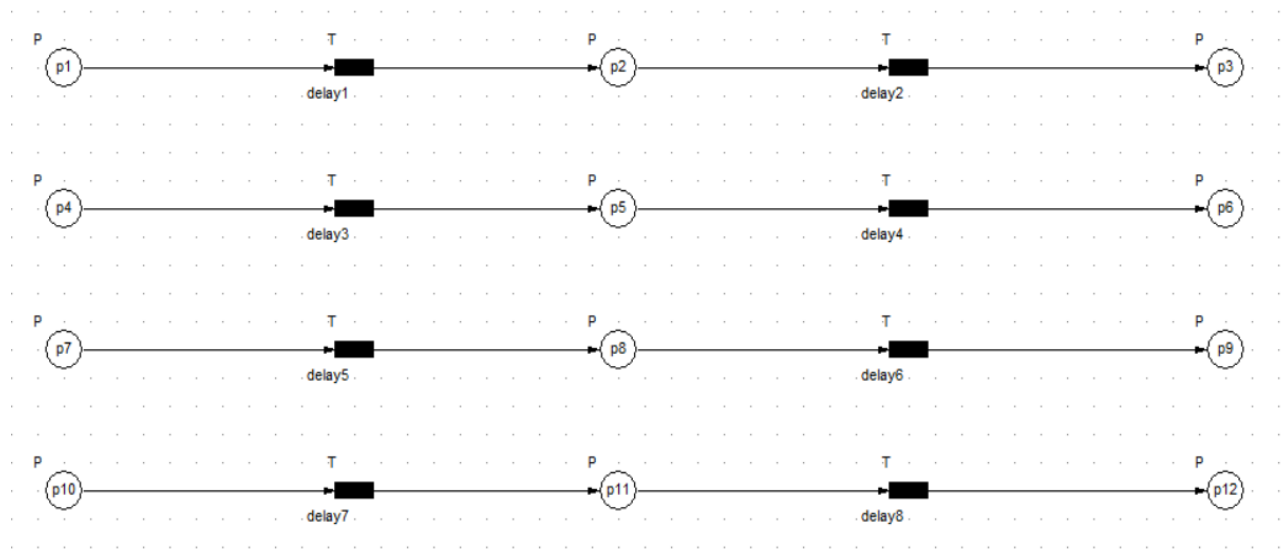
**Figure 21: Basic Timed PN model of a process.**

It must be noted that in the general case a setup is needed in order for the machine to start the performance of a new process. Machine setup is performed once before a new batch process begins. According to this the fundamental model of a machine setup and process of a batch of parts in a machine is Figure 22. First transition refers to the machine setup and the second one to the batch process.



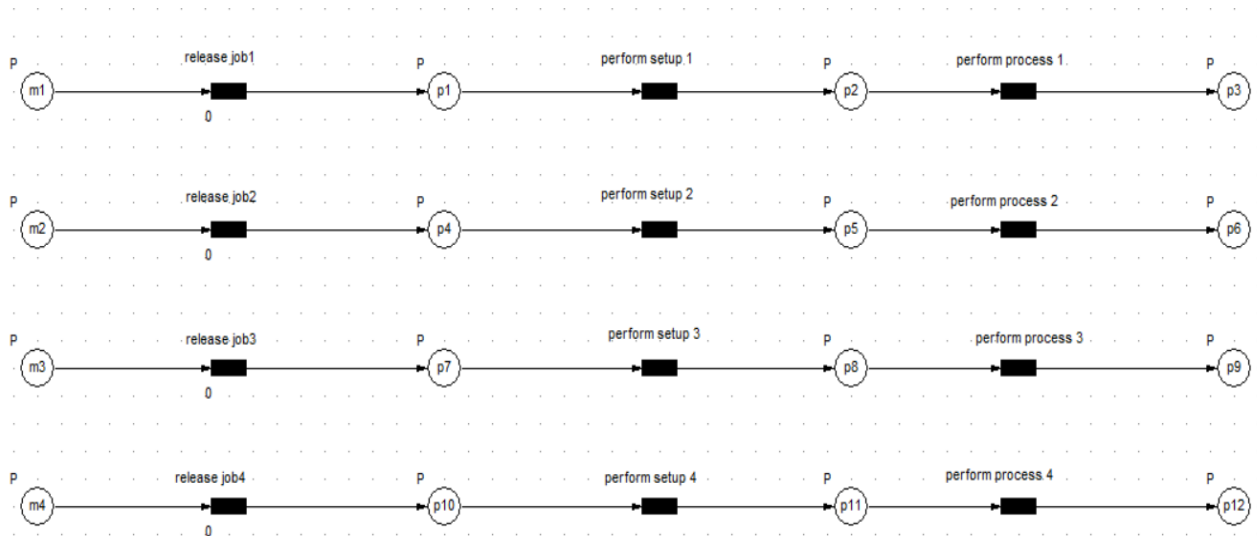
**Figure 22: Timed PN model of a process with machine setup.**

According to the modelling procedure described a Timed Petri net model for each machine will be implemented. In order to do this the number of jobs performed in this machine according to the schedule must be fed from optimisation module. For each job such a structure will be iterated. For example, the model of a machine that in different time periods performs 4 types of processes consists of 12 places and 8 transitions according to Figure 23. It must be pointed out that the 4 states of the machine are mutually exclusive and cannot be performed at the same time and this would be enabled in the following steps of model building procedure. In particular, the connections to places  $p_1$ ,  $p_4$ ,  $p_7$ ,  $p_{10}$  and the ones from  $p_3$ ,  $p_6$ ,  $p_9$  and  $p_{12}$  are the points of possible interaction of the machine model with the other entities of the system. The general case of a machine refers to the performance of  $n$  processes and the respective model is extended correspondingly with respect to the output of optimisation.



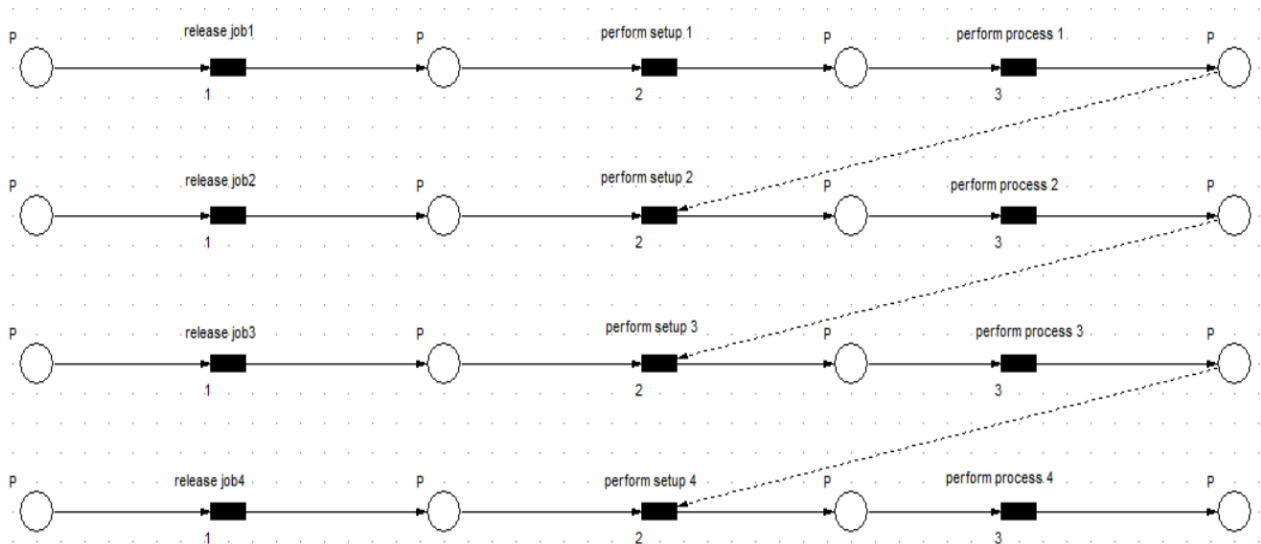
**Figure 23: Timed PN model of a machine that performs 4 types of process.**

A similar structure (a pair of places and a transition between them) will be used to represent the release dates of certain jobs in the system in the case that not all these jobs are available from the beginning. The delay of the transition will refer to the actual release date while the first place represents the job that has to be processed and the second one the job available for process in the system. Even in the special case where the jobs are available from the beginning, the delay associated to the respective transition can be equal to zero. Figure 24 shows a general implementation of the model of a machine that in different time periods performs 4 types of processes that are released at different time instants and require a machine setup before the process. This is the most general case of the Petri net model of a machine that performs  $n$ -types of processes.



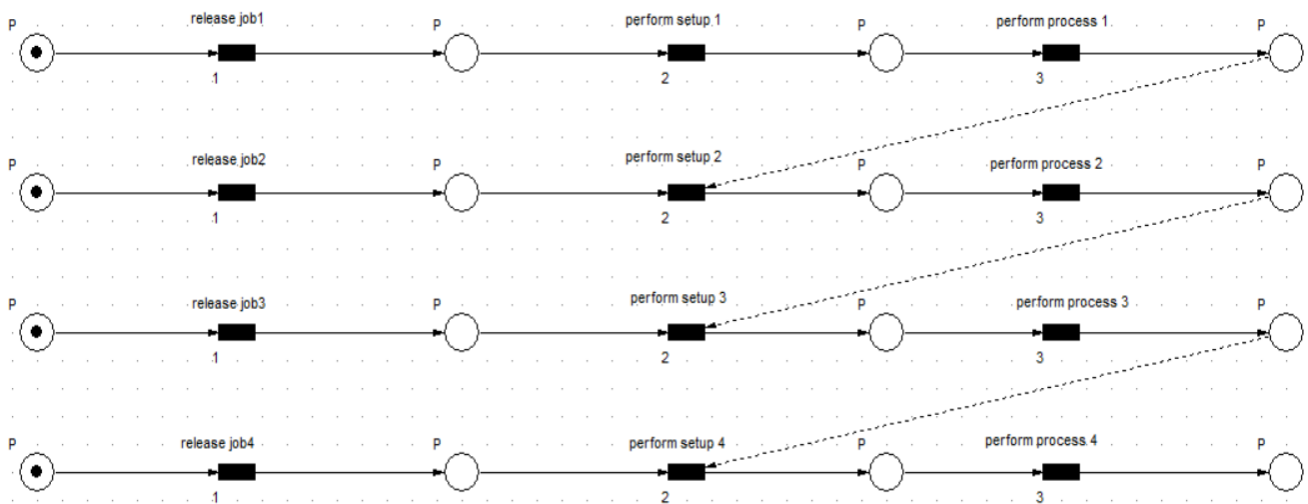
**Figure 24: Timed PN model of a machine that performs 4 types of process with different release dates.**

Till this step, the main structure of the machine module model has been defined. However, in this structure, the sequence in which the different types of processes will be performed must be defined. Let's consider in the previous example that the sequence of the jobs performed in the machine is 1-2-3-4. The structure of the Petri net model must enable the implementation of this policy. This concerns two main constraints. First of all, that the sequence specified from optimisation will be followed and second that machine will have finished the performance of a job before the setup of the next in the schedule. This is achieved by using a test arc to connect place representing products after the performance of a process (final place of a process model) to the transition representing the performance of the next process according to the schedule. For the case considered, the structure of the Petri net model of the machine module is shown in Figure 25.



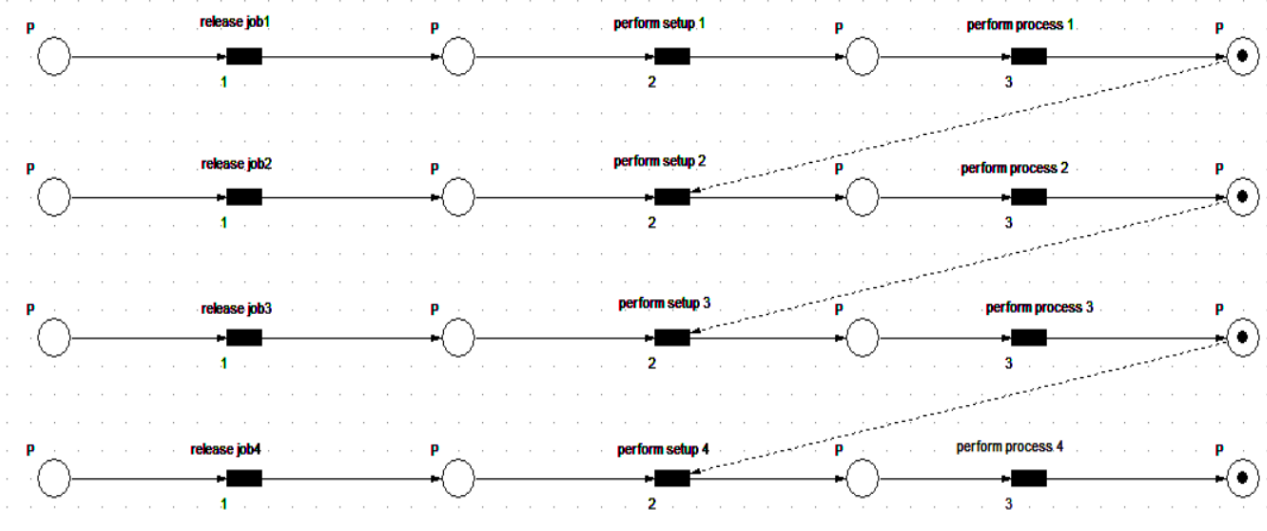
**Figure 25: Timed PN model of a machine that performs 4 types of process with different release dates following a defined sequence.**

The next step considers the definition of quantitative parameters in the PN model structure. Such parameters refer to time delays of the transitions as well as to the initial marking (initial model state) as described from token distribution. Initial marking defines the initial state of the system and with respect to the specific parameters will enable the simulation of defined scenarios. In this model, one token must be in any case present in each one of the places representing the jobs that have to be processed (initial places of the process branches). For this case, Figure 26 shows the PN model after the addition of quantitative parameters.



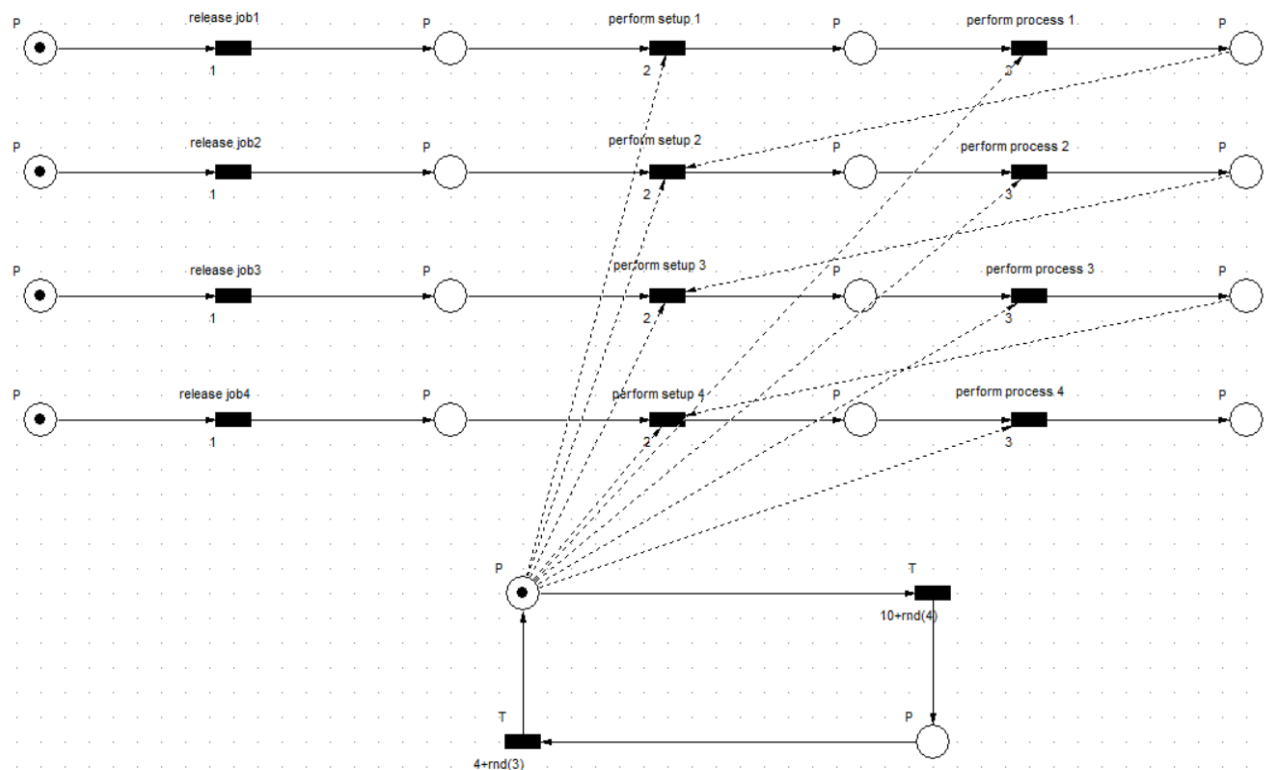
**Figure 26: Initial state (scenario) of the Timed PN model of a machine that performs 4 types of process with different release dates following a defined sequence.**

This model represents a specific scenario and can be simulated. The final state of the simulation, when no more events can be implemented, is shown in the following Figure 27. The final token distribution represents the final state of the system and in this case refers to the finish of the performance of the 4 different processes from the machine that becomes idle. From this simulation, the overall duration can be calculated as well as measures such as idleness of machines, when the performance of each part finishes and so on.



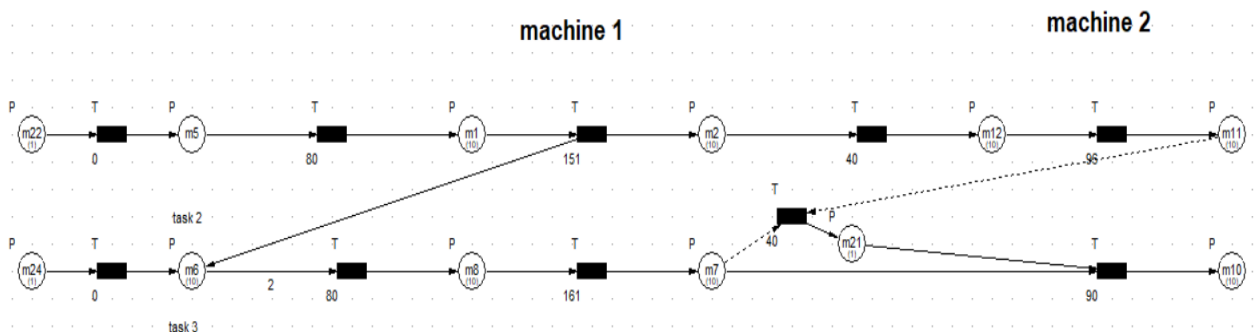
**Figure 27: Final state of the Timed PN model of a machine that performs 4 types of process with different release dates following a defined sequence**

In cases where related data exists, the distribution of the breakdowns appearances of as well as their maintenance durations exist can be added to the structure of the Petri net model (Figure 28). In this case, the two transitions refer to the appearance and maintenance of breakdowns in the machine, while the first place refers to machine being productive and the second one to machine out of order and under maintenance.



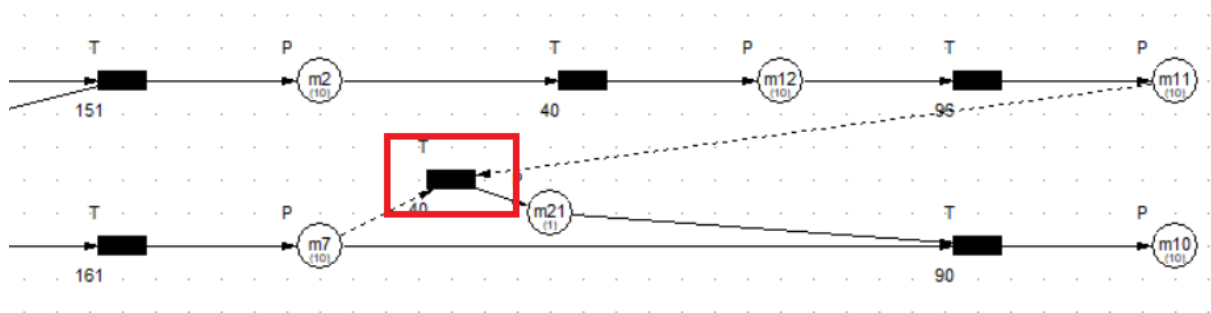
**Figure 28: Timed PN model of a machine with breakdowns (unreliable machine) that performs 4 types of process with different release dates following a defined sequence**

Up to this moment the fundamental model of machine that is the main entity of the system has been considered independently. However in the case of an industrial system there are interactions between machines, as certain processes consist of 2 stages (cut and then bent of bars) that occur in different types of machines with a given sequence. Let us consider a simple case representing two machines, each one performing two processes after machine setup, where process type 1 must be performed before process type 2, there are release dates and machine 2 processes parts after their processing in machine 1. In this case, the Petri net model is shown in Figure 29.



**Figure 29: Timed PN model of two interacting machines that perform two types of process**

In this model, there is only one release transition per job before the performance of the first process in machine 1. In addition, an interesting point has to do with the implementation of the second process in machine 2. As can be seen in detail in Figure 30 the machine setup of machine 2 begins only when machine 2 finishes the process of the previous job and at the same time machine 1 finishes the process of this job. This is achieved through the use of 2 input test arcs to the transition representing machine setup for process 2 in machine 2 (showing with red). These two input test arcs to the transition showing machine setup come from the place representing products after the performance of the process in machine 1 and from the place representing products after the performance of the previous process in machine 2.



**Figure 30: Implementation of the machine sequence constraints in Timed PN structure.**

A specific production example in Bay 3 is considered. According to this, 8 jobs are assigned in the machines m1.m3, m5, m6, m7 and m8 in the sequence shown in Table 3.

Jobs Assigned		
Machine	1st job	2nd job
machine 1	1	2
machine 3	3	4
machine 5	7	8
machine 6	5	6
machine 7	7	8
machine 8	5	6

**Table 3: Job sequence in each machine as defined from optimisation module.**

Optimisation provides one job sequence for each machine used. Each machine performs exactly 2 jobs, while 4 jobs receive a two-stage process and 4 jobs one stage processes. 6 machines are used in total according to the schedule calculated while 3 more are not used for many reasons (production characteristics, possible maintenance etc.) The machine numbers correspond to their types as defined above for the machines of Bay 3. The operation of the machines has been considered without breakdowns. jobs are not split and correspond to the production of batches with given quantitative characteristics.

The respective Petri net model of the scenario is presented in Figure 31.



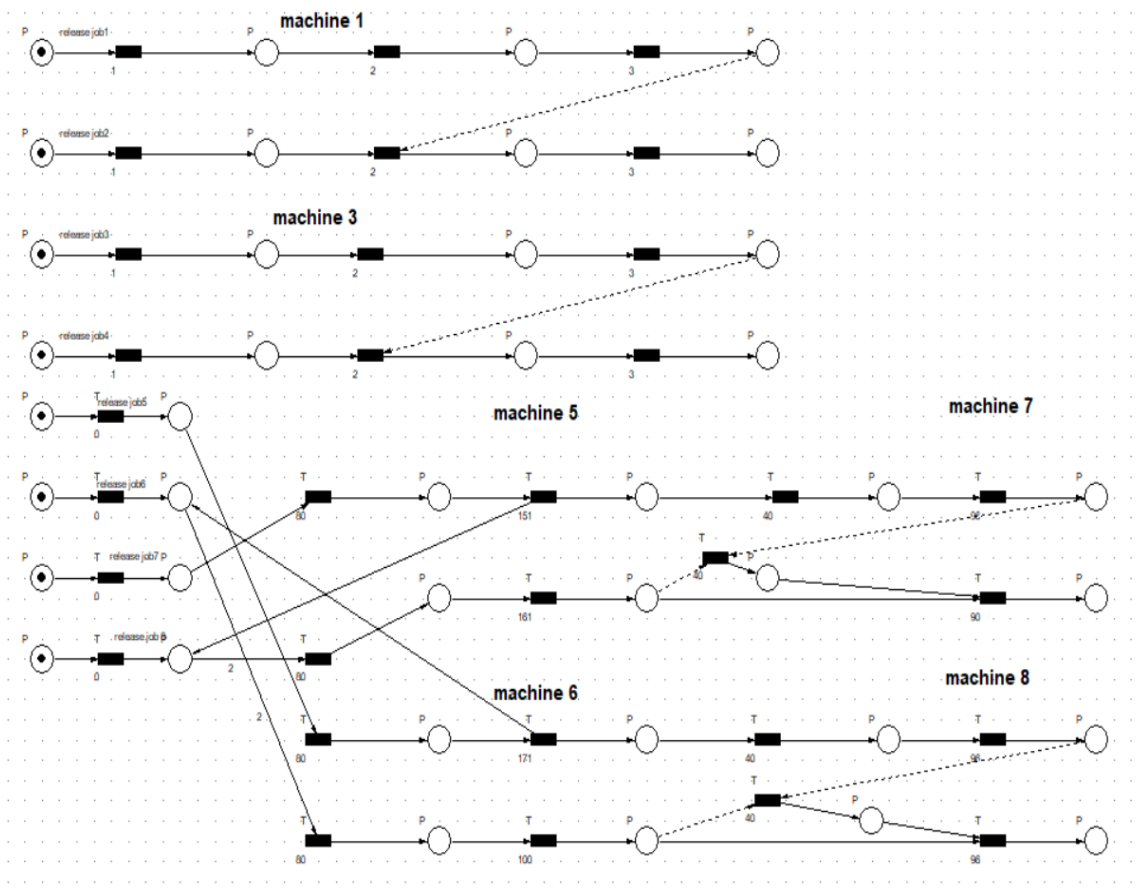


Figure 31: Timed Petri net model of a complicated production scenario in Bay 3

## 5.2.2 Automotive Manufacturing: Pilot Case by CONTINENTAL

### 5.2.2.1 Introduction

Continental is among the top worldwide electronic manufacturers. Its products are manufactured in Electronic plants such as the plant in Timisoara a part of which is considered as test case. The specific plant is producing high electronic products and covers the stages from design till the production. The design stage is performed by the group of development located in different plants worldwide. Continental's customers may define specific parameters of the products to their individual needs from design phase.

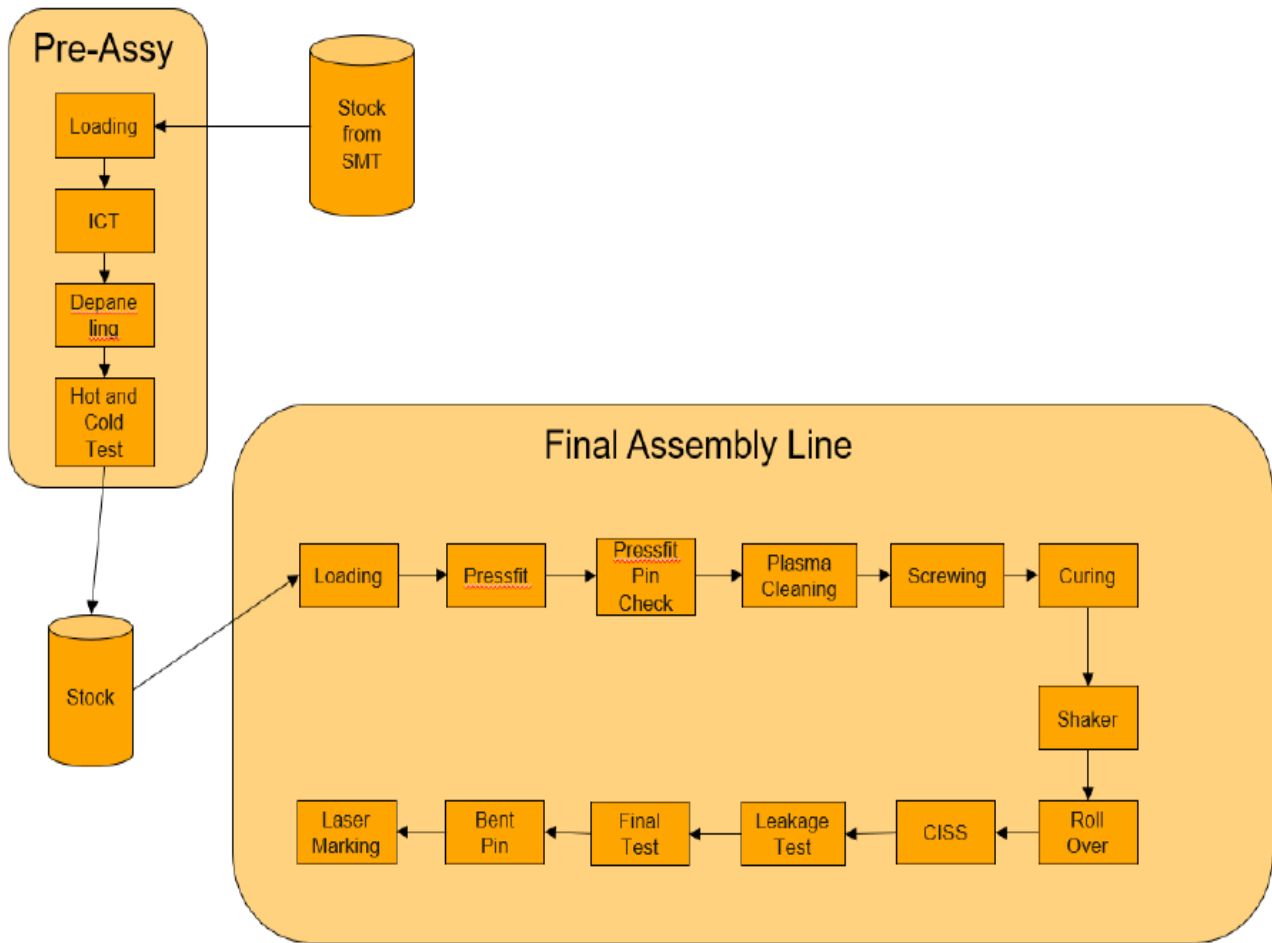
#### 5.2.2.2 Case Description

Although the products produced (e.g. airbag control units, chaises controllers, hand brake controllers etc.) have a high complexity degree, their routings can be described (in brief) as follows.

- SMT (Surface Mount Technology) lines. High automated lines where electronic components are placed on the PCB boards.
- PCBA (Printed Circuit Board Area). PCB area, where the electronic built in SMT will be separated in smaller parts ( PCB's ) and tested electrically ( In Circuit Test ). Additional processes can also take place in this area like Press Fit, Handling, Flashing of Microcontrollers and Temperature functional tests.

- FA (Final Assembly) and Test Area. This is the step of production where the electronic is connected to the mechanical part and finally tested and labelled. The processes in this area are in the area of connecting the mechanical parts: Screwing, Press Fit, Gluing, Riveting, Snap In. The testing area consists of tests line Functional test of the product, Automatic Optical Inspection, Force monitoring for the snap in, air leakage test.
- Packaging and delivery operation. Within this step of manufacturing, we are packing the products in customer specific boxes and link all the information needed by customer to the unique number of each box.
- To maximise life-time of equipment's involved in production process (Process equipment but also Test equipment), Continental maintenance & repair departments perform different maintenance techniques. Also, workers from the shop floor play an important role through the information provided about equipment's' behaviour in operations. In this regard, valuable information is gathered and interpreted in order to detect early possible failures or defects.

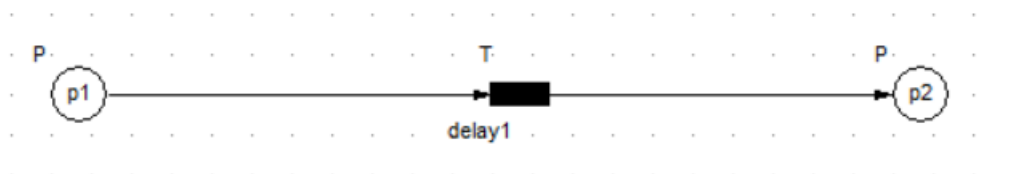
In the case of Continental, the Production procedure considered in FACLOG Project is depicted in Figure 32. A production line consisting of two parts (or two subsequent production lines) is considered. The first subline is called preassembly line, and is composed of 4 stages, followed by one buffer and then by a 13-stage final assembly process. The names and types of the processes performed in each stage are shown in Figure 32. The main assumptions regarding this case are the following: all jobs follow the same routing through each line (there is no flexibility), which means that schedules provided from optimisation module should be calculated per line and not per workplace, as they would be common for all the workplaces of a line. No internal buffers are considered between workplaces except the one mentioned, no parallel workplaces exist in each production stage (single server stages) and the processes performed are highly automated. Change of type of parts processed in the line is followed by setup with given duration of the workplaces of each line. The duration of the movement of parts between workplaces is considered negligible. Orders refer to batches of products with similar characteristics and an order received from Continental can be divided in suborders following specific rules and constraints.



**Figure 32: Continental production line stages**

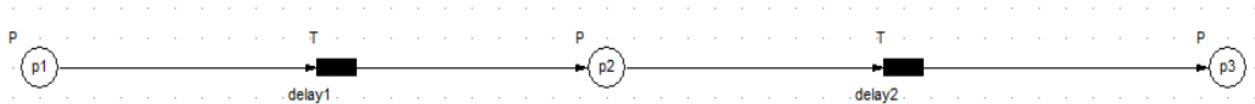
#### 5.2.2.2.1 Model of a process

The simplest case of the Timed Petri net model of a process is shown in Figure 33 and is the same one with the one presented for BRC model. According to this, the model consists of two places and a transition between them. The initial place ( $p_1$ ) refers to the available unprocessed parts (resources) while the final one ( $p_2$ ) refers to products after the performance of the process. Transition T with delay equal to a value represented as delay1 describes the performance of the process in a part of the workplace. If multiplicities of parts or products are not unitary, then weights can be added on the input or output arc of the transition.



**Figure 33: Basic Timed PN model of a process**

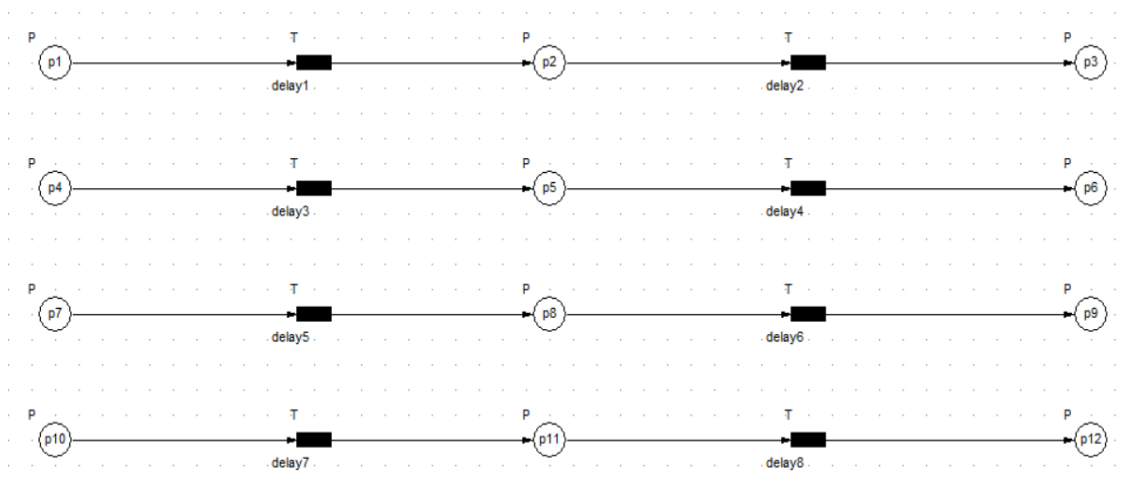
In the general case a setup is needed in order for the workplace to start the performance of a new process. Workplace setup is performed once before a new batch process begins. According to this the fundamental model of a workplace setup and process of a batch of parts in a workplace is the following. The first transition refers to the workplace setup and the second one to the process of a batch of resources in the workplace. Transfer delays associated to moving and loading/unloading of parts are considered negligible.



**Figure 34: Timed PN model of a process with machine setup**

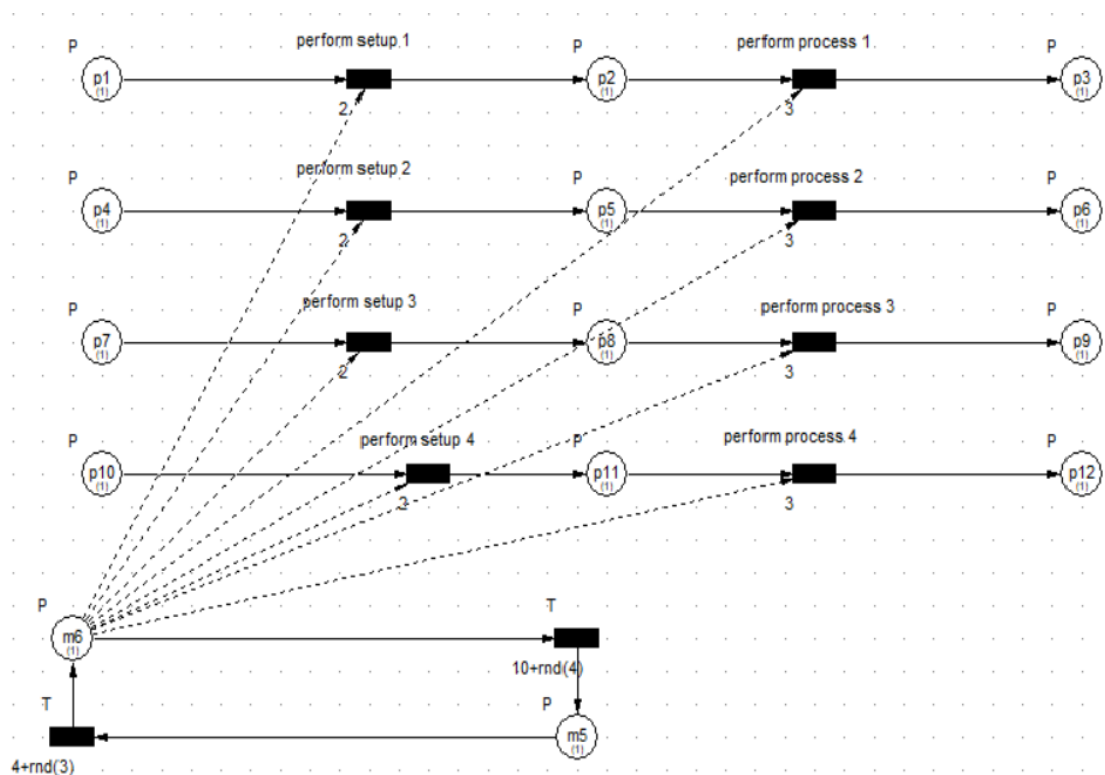
According to the described modelling procedure, a Timed Petri net model of each workplace will be implemented. In order to do this the number of jobs performed in each line according to the schedule must be fed from optimisation module. The number of Jobs is equal to the total number of suborders scheduled from optimisation module. An order is divided in suborders concerning minimum and maximum number of parts according to assumptions and rules specified by production operation of Continental. If a certain batch of resources is not being processed in all workplaces, the associated delays referring to the setup times and process times can be considered equal to zero (these batches receive dummy processes in the respective workplaces).

For each job, such a structure will be iterated. For example, the model of a machine in different time periods performs 4 types of processes consists of 12 places and 8 transitions according to the following Figure. The states of the workplace are mutually exclusive and processes cannot be performed at the same time. In particular, the connections to places p1, p4, p7, p10 and the ones from p3, p6, p9 and p12 are the points of possible interaction of the workplace model with the other entities of the system.



**Figure 35: Timed PN model of a machine that performs 4 types of process**

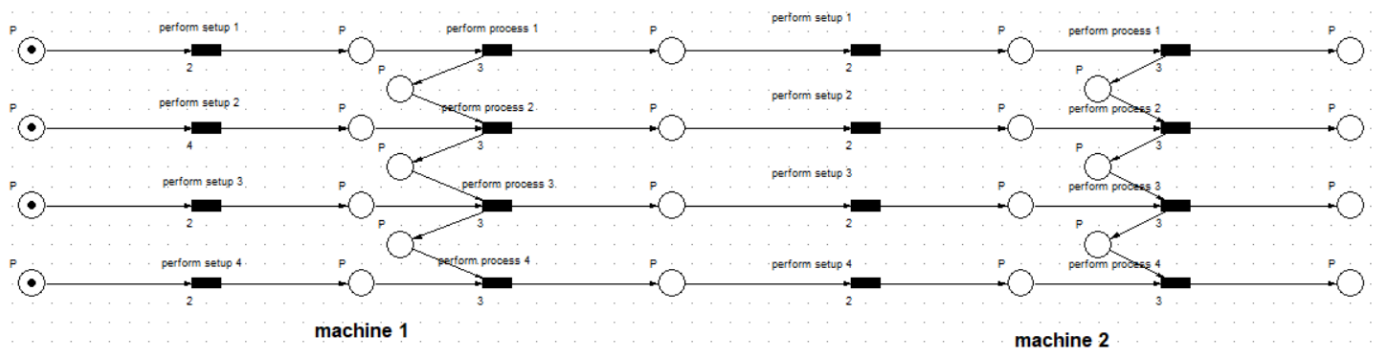
If for a workplace data concerning the distribution of the breakdowns appearances as well as their maintenance durations exist, then this can be added to the structure of the Petri net model, as shown in Figure 36. In this, the two transitions refer to the appearance and maintenance of breakdowns in the workplace, while the first place refers to workplace being productive and the second one to workplace out of order and under maintenance. The place representing workplace being productive is connected through test arcs to all the transitions representing workplace setup performance and performance of processes. This happens as no setup or parts process can be take place when workplace is under maintenance. The delays associated to the two transitions can be non-constant and have a random part, if the models are used offline, or may receive inputs from sensors if they are used online with the physical system.



**Figure 36: Timed PN model of a machine with breakdowns (unreliable machine) that performs 4 types of processes**

In the general module model structure, the constraints regarding the use of the workplaces must be implemented. Each workplace has to finish the process of a job in order to start a new one, while a machine can start the setup in order to process a job only when the process of this job in the previous workplace has finished. In order to achieve the first condition workplace models are connected linearly (model of the second machine after the respective model of the first and so on). For the second one, a dummy place is added for each branch, with input the transition representing the performance of the previous process in the workplace and output the transition representing the performance of the process in the current workplace. This structure enables the performance of the process in the current workplace only when it has finished in the previous one. These concepts are presented in the implemented in the Petri net structure in Figure 37. The model of this figure represents two consecutive workplaces that perform 4 types of processes. As these workplaces are

parts of the same line the sequence is common and is implemented in the PN model structure in the first of the two machines. In addition, the output places of the first machine are the input places of the second and this is the part of the system where the interactions between the entities take place, as there are no multiple or parallel machines that perform the same type of process.



**Figure 37: Timed Petri net model of two consecutive workplaces (from the same production line) that perform 4 types of processes**

The next step considers the definition of quantitative parameters in the PN model structure. Such parameters refer to time delays of the transitions as well as to the initial marking (initial model state) as described from token distribution. Initial marking defines the initial state of the system and with respect to the specific parameters will enable the simulation of defined scenarios. For all scenarios, one token must reside in each one of the places representing the jobs that have to be processed (initial places of the process branches).

A specific production example for Continental preassembly and final assembly line is presented next. According to this, 4 jobs are scheduled to be performed in both lines, and the sequences that have to be followed are shown in Table 4, as derived from optimisation module. The operation of the machines has been considered without breakdowns while all jobs are not further spitted and correspond to the production of batches of produces with given quantitative characteristics.

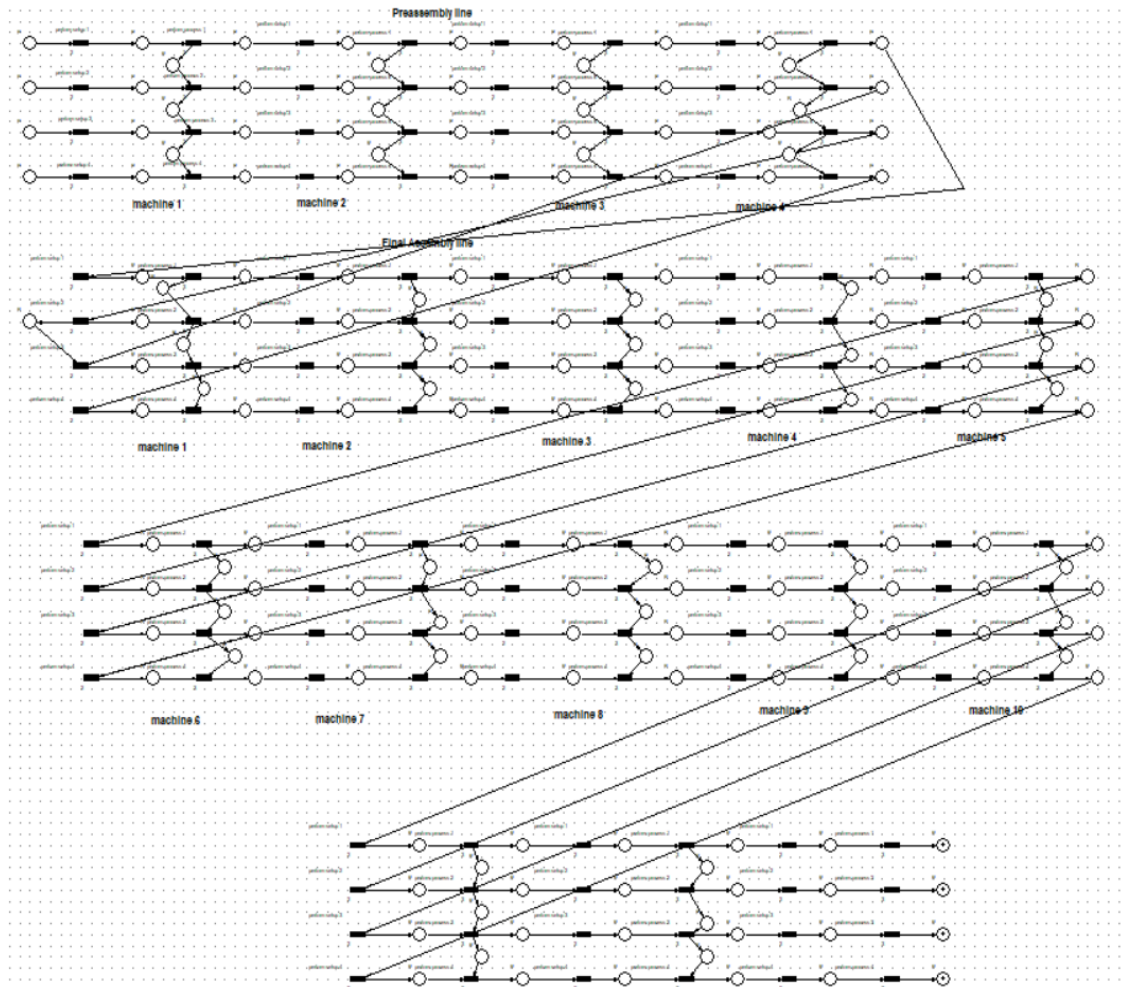
Jobs Assigned				
line	1 <sup>st</sup> job	2 <sup>nd</sup> job	3 <sup>rd</sup> job	4 <sup>th</sup> job
preassembly	1	2	3	4
Final assembly	1	3	2	4

**Table 4: Description of a scenario composed from a 4 Job sequence for each production line**

The respective Petri net model, of the scenario described is presented in Figure 38. That follows. It consists of 188 places and 134 transitions, and it is live as long as tokens that correspond to parts under process exist, bounded as the number of tokens never exceeds a given number and can be used for simulation if the delays are defined. For its implementation the fundamental PN module has been used as structural component 17



times (as there are 4 machines in preassembly and 13 in final assembly line) appropriately connected with respect to system topology. A dummy place has been added in the structure at the end of the preassembly line and before the final assembly in order to enable the sequence change described in Table 4 above.



**Figure 38: Overall System (Preassembly and final Assembly lines) Timed Petri net model for the scenario defined in Table 4**

### 5.2.3 Textile Industry: Pilot Case by PIACENZA

For the PIACENZA pilot, modelling focuses on the weaving part of fabric production, which is the part of production that will be optimized within FACTLOG. Weaving is the process of building a fabric, obtained by intertwining the warp and weft threads. In the weaving process a set of articles/orders arrive over time and have to be woven by a set of parallel looms, with respect to their strict delivery dates. Each order is linked to the production of a specific fabric type and is accompanied by a positive quantity/demand (in meters), while the looms are uniform, meaning that each loom operates on different speeds.

The simplicity of the weaving production process schema, i.e., a single level of parallel looms, and its close connection to the optimization problem at hand has led to integrating



the model of the Piacenza weaving process within the Optimization module. The corresponding representation enables modelling all parallel looms and their processing speeds, all orders' types and their fabric characteristics, as well as the sequence dependent set-up times between different orders. The latter refers to the time required to set up a loom that was processing a fabric of one type so as to process a fabric of a different type; notably, the setup time between any two orders  $j$  and  $k$  is different than setup time between orders  $k$  and  $j$  on the same loom. Moreover, the corresponding model incorporates loom setup resource constraints, in the sense that there is a limited number of setup workers which places a constraint on the number of setups that can be performed simultaneously on different looms.

The main entities included and their corresponding attributes are provided below:

- **Orders.** Each order has a specific type (C: sample, P: regular, F: pre-sample) and a processing status (ATEL: under process, ORD: waiting/to be scheduled, LANC: new arrival). Each order is associated with a strict deadline, a specific quantity of the fabric type and the corresponding energy consumption for the time period that is processed.
- **Fabric.** Each fabric is identified by a fabric type, a design code and a colour change code as well as a list of attributes such as number of yarns, strokes per meter, "annotability" and "chainability" codes, comb height and code and complexity index. These attributes are used to calculate the processing time of each order in each loom and its setup time in that loom.
- **Looms.** The set of available parallel looms, each loom associated with a fixed production speed.
- **Workers.** The set of available groups of workers. These groups are considered to be identical. They are responsible, within their corresponding shifts, to set up the looms to handle the corresponding orders that have been assigned to them. Typically, 3 groups of workers are available in each shift.

Moreover, the weaving process model enables incorporating disturbances as well as maintenance activities. Disturbances typically refer to events such as sudden loom malfunction/breakdown that negatively affect the production. Another type of disturbances is yarn breakage, which seems to occur quite often during the processing of each order on a loom. Maintenance activities refer to possible pre-determined machine unavailability.

The provided model enables the simulation and evaluation of different production schedules (i.e., different orders placed on different looms for a given time horizon, also enabling order-splitting, where each order is allowed to be split and processed on multiple looms simultaneously). The evaluation is performed under different KPIs, i.e., makespan, average completion time of each job, reduction on the total tardiness and the number of tardy jobs, change on energy consumption.

## References

- [1] 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.
- [2] 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
- [3] K. M. Hangos and I. T. Cameron, “Process Modelling and Model Analysis - Chapter 2 : A Systematic Approach to Model Building”, Process Systems Engineering, Academic Press, Volume 4, 2001, Pages 19-40
- [4] A. Desrochers, A. and R. Al-Jaar, “Applications of Petri Nets in Manufacturing Systems - Modeling, Control and Performance Analysis”, IEEE Press, 1995.
- [5] 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.
- [6] 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, Singapore.
- [7] 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, November 12–14, 2011
- [8] Franck Pommereau, Quickly prototyping Petri nets tools with SNAKES, International Workshop on Petri Nets Tools and Applications PNTAP 2008, Mar 2008, Marseille, France.
- [9] Pommereau, Franck, 2015, SNAKES: A flexible high-level Petri nets library, 9115, 10.1007/978-3-319-19488-2\_13.
- [10] FACTLOG Deliverable D7.1 (2021). FACTLOG Installation and Initial Testing (Interim Version)