POLITECNICO DI MILANO

Scuola di Ingegneria Industriale e dell'Informazione
Master's Degree in Management Engineering

POLITECNICO
MILANO 1863

Master's Degree Thesis

A digital approach for the automatic definition of performance evaluation models for flow-/job-shops

Supervisor

Prof. Marcello URGO

Doc. Massimo MANZINI

Candidate

Filippo BORTOLOTTI

Student number
905363

Academic year 2019/2020

Abstract

The increasing request of connectivity and interoperability in the manufacturing sector, in particular for small and middle size companies, creates the need of flexible and integrated solutions for the management and performance evaluation of a system.
The objective of this thesis is to define an approach for the representation of a physical production system and from it developing a procedure for the automatic generation of performance evaluation models. The approach aims to represent an ontology based production system in a queuing network trough the definition of modelling hypothesis and rules. Next step is to apply the model to integrate the ontology production system to the simulation tool for the performance evaluation. This integration is possible by analysing both the connected elements to understand how the data can be extracted from the ontology and how the approximate model can be imported in the simulation tool. This approach is then verified through the definition of multiple study case where for each of them, some approximate model KPIs are compared with the one obtained from a commercial software. Moreover, a possible representation of assembly system for the performance evaluation tool has been presented and validated.

Estratto

L'aumento della richiesta di connecttività e interoperabilità all'interno del settore industriale crea la necessità di ottenere soluzioni integrate e flessibili per la gestione e la misurazione delle performance per un sistema produttivo.
L'obiettivo della tesi è quello di definire un approccio per la rappresentazione di un sistema produttivo fisico e da esso sviluppare una procedura per la generazione automatica di modelli per la valutazione delle performance. L'approccio mira a rappresentare un sistema produttivo basato sull'ontologia con un sistema di code attraverso la definizione di regole e ipotesi di modellizzazione. Il prossimo passo è quello di applicare il modello per intrgrare il sistema produttivo di ontologia al tool di simulazione per la valutazione delle performance.
Questa integrazione è possible analizzando entrambi gli elementi connessi e capendo come i dati possano essere estratti dall'ontologia e come il modello approssimato possa essere importato nel tool di simulazione. Questo approccio è poi verificato attraverso la definizione di multipli case study dove per ognuno di essi alcuni KPI del modello approssimato sono confrontati con quelli ottenuti da un software commerciale. In più, una possibile rappresentazione di un sistema di assemblaggio per il tool di valutazione delle performance viene presentata e validata

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Nowadays, the digital world is acquiring more and more importance. With the new revolution of Industry 4.0, also manufacturing industry is showing signs of this shift to the connected environment. The manufacturing industry shows trend towards automation and data exchange in manufacturing technologies and processes such as cyber-physical systems, the internet of things devices and cloud computing. These solutions can be achieved with an interconnected system, where data are immediately available and updated; grounding on this, a new need of connection solutions arises.
In particular, manufacturing sector experienced a shift to a continuous improvement focus exploiting new technologies available on the market. Under this light, also the production plan design requires solution for the improvement of the efficiency. During the life cycle of a production system, there could be multiple events that trigger the change or reconfiguration of the system. Every time the system changes is it needed to define a new formal representation, a new model of this system and the performance analysis representation. This performance analysis can be conducted with an analytical approach or through a simulation by exploiting different techniques and obtaining results with a different level of detail. The focus of this thesis is on the simulation side. The process of updating a system can require multiple step in order to find the best solution. During each step, the physical model is defined with the new parameters, then the simulation model is created based on the physical one and the simulation is launched. The performance are collected and evaluated by comparing the data with the previous one and the process can start again. A search of the most suitable solution requires multiple analysis, and for each analysis, multiple update of the system according to the different representation, such as physical and simulation model.

In the market, there are available simulation tools which let the user define a single system keeping the formal and performance evaluation representation connected. The problem is that these software are expensive, which limit their use,

and have a rigid structure that allows only a little integration with external tools. In the design and configuration phase, multiple systems, have to be evaluated in order to obtain the best solution. In this evaluation phase, there are lot of other elements to be considered together with production KPIs, such as physical location of the production system, CAPEX and OPEX; these parameters to include in the choice are scattered along different tools. The lack of integration of the commercial software requires a manual update of every single representation. A definition of an automatic tools for the generation and performance estimation of a system could improve the efficiency of this process. The objective of the thesis is to define a model for the representation of a physical production system from a ontology representation to a queuing network representation and, based on that, to develop a tool for the conduct of automatic performance evaluation without the need of a manual update.
This solution is possible through the use of open-source tools in order to obtain the maximum possible flexibility. For this reason, we focus on the use of ontology representation. The use of an ontology-based representation in the manufacturing environment is increasing in the lasts years thanks to open standards which allow the creation of connections between system [1]. The use of an ontology allows us to formally represent the production system and use this representation as a way to integrate different software coming from different ICT companies, including performance evaluation tools.

The first barrier to the diffused use of ontology is an efficient instantiation and management tool [2]. For this reason, OntoGui software can be adopted to overcome this obstacle and define the production system with simplicity through the use of its GUI.
Regarding the choice of the simulation tool, we consider the use of Java Modelling Tool (JMT from now on), a free open source suite. The suite uses an XML data layer that enables full re-usability of the computational engines. It consists of six tools, such as SimWiz (Java Simulation Textual) , JSIM Graph (Java Simulation Graphic), JMVA (Java Mean Value Analysis), JMCH (Java Markov Chain), JABA (Asymptotic Bound Analysis) and JVAT (Java Workload Analyzer Tool). In particular, we will consider the use of JMT JSIM that allows exact and approximate analysis of single-class or multi-class product-form queuing networks, processing open, closed or mixed workloads.
With this availability of tools whose flexibility allows a high level of iteroperability, there is the need to understand how to create this connection. After the definition of the model from the ontology structure to the queuing network representation, the focus of the research is moved to develop a bridge between the system represented in the ontology for the performance evaluation process. We will work on the definition on this connection using the OntoGui for the ontology representation and JMT JSIM as the performance evaluation tool.

The results of this thesis will be part of the results of EU funded project called VLFT: Virtual Leaning Factory Toolkit [3]. The aim of VLFT convey the results of research activities in the field of digital manufacturing, starting from modelling, performance evaluation and virtual and augmented reality into the curriculum of engineering students. The final objective of this project is to use these virtual tools to support the students in learning complex engineering concepts and enhance their learning ability. The project consortium is composed of three different universities in Europe: Politecnico di Milano, Tallinna Tehnikaulikool and Chalmers Tekniska Högskola, and two research centers: STIIMA-CNR and MTA-SZTAKI. VLFT project is supported and funded from ERASMUS +, for this reason they are defined also activities in collaboration with foreign students and workshop in project partner premises.
The project also includes the definition of a case study grounding on a real production system. The industrial partner is Cosberg, an italian manufacturing company located in Bergamo area. Cosberg design and create machines and modules for the automation of assembling processes. One of their new assembly line is used as the case study to be developed. This case study will be used for the application and testing for the generation tool developed.
In Chapter 2, one it is defined is is defined the state of the art si discussed, where it is also explained the use of the tools used. In Chapter 3, it is described the creation of the model requiring the analysis of both the element of the connection in order to define hypotheses and rules for the a valid approximation. Chapter 4 discusses how the data defined necessary for the model definition can be obtained from the ontology and how its model representation has to be encoded to define the system in JMT JSIM for the performance evaluation. This step requires the understanding of the OntoGui structure and the XML standard input required by the simulation software.
Chapter 5 involves the real performance evaluation and its verification to prove the model ability to approximate the original system.
Final consideration of the possibilities and the limits of the model, and possible future improvement are defined in Chapter 6 .

Chapter 2

State of the art

In this chapter is being analysed the context in which this thesis is being developed and what is the current research in the topic defined. In the first part it is introduced the new connection need that is increasing in the industry. In Section (2.1 it is defined what are the possible solution for the digital representation of a system. Section 2.2 discusses about the performance evaluation issue and what is the solution chosen.

The thesis lies in the new industrial revolution called Industry 4.0 . This new revolution brings with itself a new definition of system: there are not more small systems with a limited way to communicate with each others, but everything is connected. Each system should be able to exchange data with others without any limitation. In particular, the manufacturing sector shows a great interest in this change, due to the presence of multiple sub system within a single system: the physical production line, its digital representation on various tools, the sensors able to collect data from the line and the KPIs obtained. The different sub-system representation are all independent, for this reason any update on the real system requires the change to be manually defined on each of its representation. This updating process can be addressed by the creation of connections between the different representations that a change in one of it can also be applied simultaneously to the linked systems. Due to the increase of digital tools in the industry, the need of interoperability is more and more critical.

The market already provided some solutions with professional digital tools which includes "all in one" solutions to include different type of representations of the system. These software can answer to the need of the manufacturing sector but present some limitations. For example, the interoperability of the various system representations is defined only within the tool boundaries, connections from it to an external one is limited or even not possible. Moreover, these solutions can be too expensive for small and middle players in the industry forcing them to rely on

different and scattered solutions. In particular, they have to rely on various tools for the representation of physical production system and their performance evaluation. For this reason a change of even a single parameter in the physical system requires the need to update of the simulation model. During a system creation or update the number of times that a system has to be changed are countless, each parameters has its effect on the system, thus, the possibilities to analyse are wide; a lack of connection is only decreasing the efficiency of the whole process. This issue will be analysed in detail in this chapter.

Research on the topic has been conducted over the years looking for a way to create an open system representation which could grant connections with other digital tools where data are defined in different structures. In particular for the performance evaluation field this has been deeply studied.
In this problem there are two element to analyse before the definition of a connection model: the system representation and the performance evaluation.

2.1 Digital system representation

The digital system representation is the key structure of the whole process since all tools refer to this definition for their execution. Due to its central position, it is crucial to find a structure that allows the integration of fragmented data models into a unique model without losing the notation and style of the individual ones. An effective data model has to include information in relation to both the physical production system and the intangible representation given by the products and their production plans. Different solutions can be found in the existing documentation. [1].
For example, consider the ANSI/ISA-95 [4], which is is an international standard for the development of interfaces. This standard proves to be effective to represent processing and production system but has some limits regarding integration with physical representation tools.
Another solution can be found with the Process Specification Language (PSL) standard [5]. PSL standard is an ontology-based standard which allows a robust representation of a process and its characteristic together with a good level of interoperability.
A valuables standard to consider is The Industry Foundation Classes (IFC) standard by buildingSMART [6] which provides most of the definitions needed to represent tangible elements of a manufacturing systems and it is possible to include also the intangible ones through definitions provided. The connection capability of the PSL standard is given from the ontology-based structure. An ontology encloses the representation and the definition of categories, properties and connections between

entities defined in a single or multiple system. Their open definition allows the integration without the risk of losing level of details of the data such as the notation [7].
One of the most common languages for an ontology definition is OWL, Web Ontology Language, it is a semantic web language designed for ontology representation. The benefit given from an ontology-based system are countermeasured by a difficulty instantiation of the OWL ontology which limit their use to the general public. A solution to this issue is proposed through the use of OntoGui [2]. OntoGui is a tool presenting a Graphical User Interface for Rapid Instantiation of OWL Ontologies. The tool allows the definition of both physical production system and product process plan through a user interface which deeply simplify the process. It can be possible to see the user interface in Figure 2.1 .

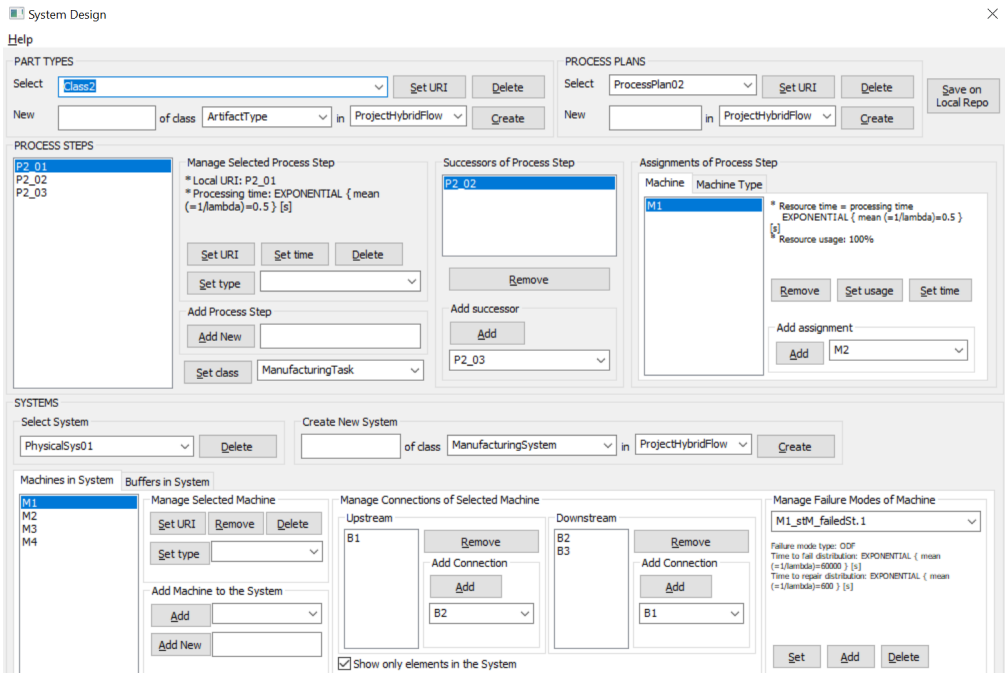

Figure 2.1: OntoGui user interface example

OntoGui is defined by different modules, some of them are pre-defined, while others provided as input or to be developed. The hierarchy of the ontology modules is depicted in Figure 4.3.

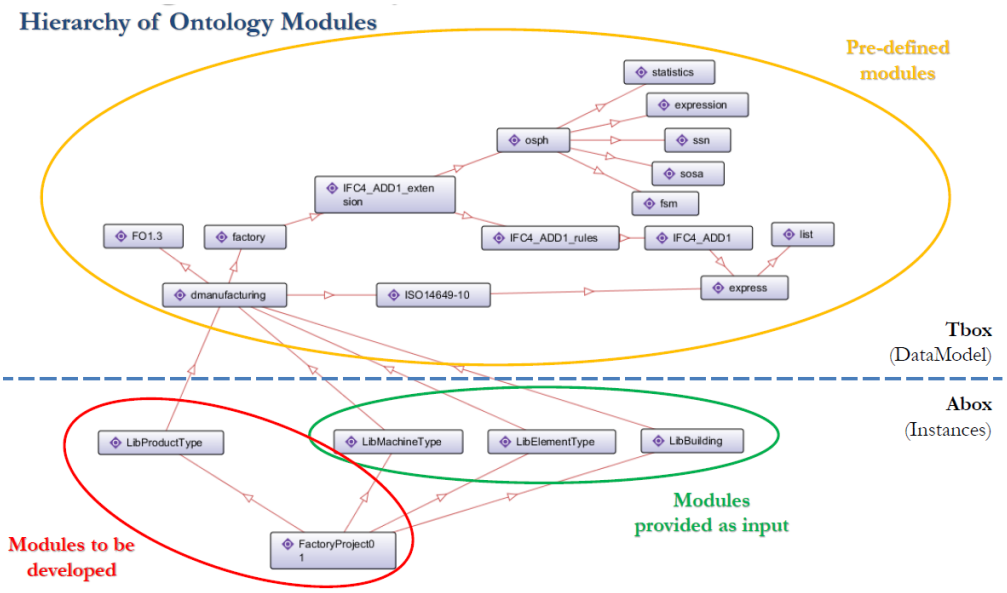

Figure 2.2: Hierarchy of Ontology Modules

2.2 Performance evaluation

Once the system representation has been identified, it is needed to analyse the second element of the connection model: the performance evaluation.
The choice of design for a manufacturing system comes from the evaluation of their performance in the production process [1]. For this evaluation two main approaches can be used:

- Analytical model: using mathematical or symbolic relationships to provide a formal description of the system
- Simulation models: designing a dynamic model of an actual dynamic system for the purpose either of understanding the behavior of the system or of evaluating various strategies (within the limits imposed by a criterion or set of criteria) for the operation of the system.

Analytical model proves faster results but with more approximation while simulation one can provide a higher level of precision but it is a slower process, moreover, this high level can only be reached from a really formal and wide definition of the original production system.
A lot of simulation commercial tools are available on the market but most of them lack of the connectivity capabilities needed. Moreover, the objective is to rely on open source software that can be accessible from a wider audience. We consider the use of JMT [8] tool. This tool is an open source software developed from

Politecnico di Milano and Imperial College London based on Java language. It provides six different tools performance evaluation, capacity planning, workload characterization, and modelling of computer and communication systems. The one used for the simulation is JMT JSIM which allows the discrete-event simulation for the analysis of queuing network and Petri net models through a intuitive user-interface; it can be possible to see an example in Figure 2.3. The simulation is based in queuing network which is a really important feature to the future definition of the model.

JMT JSIM is particularly interesting since it works through a XML layer to

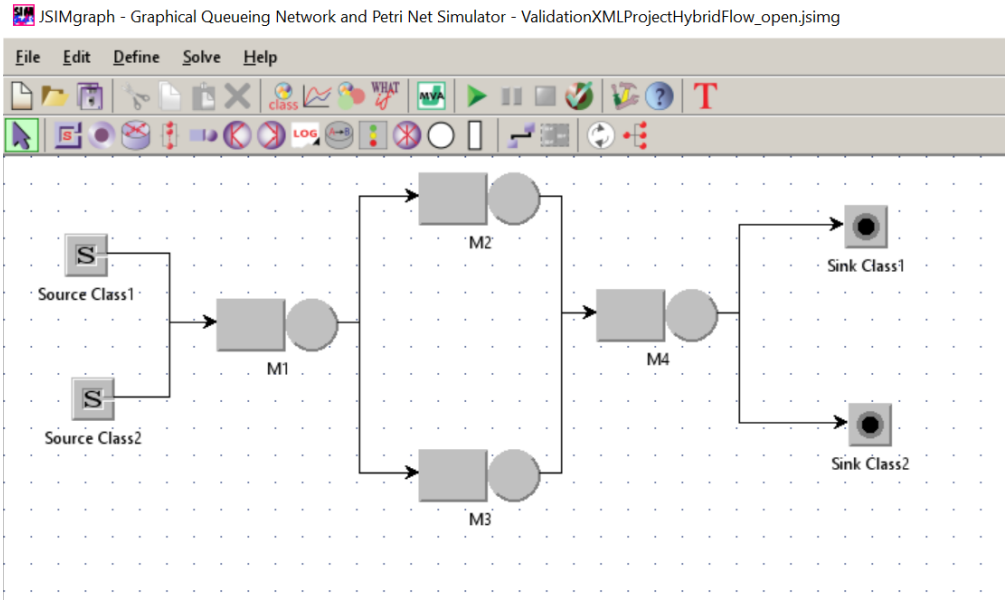

Figure 2.3: Example of a system representation in JMT JSIM user interface

connect the model definition to the analytical and simulation engine, a scheme can be found in Figure 2.4. In fact the system to be analysed can be defined through the GUI or through an XML format input. The second method allows the possibility of integration with the ontology by defining an appropriate XML based on the ontology model created.The great diffusion of this markup language makes the integration easier.

2.3 Automatic generation of a performance evaluation model

Once the tools used have been defined, it is needed how to understand how automatically generate the simulation model which can be represent the ontology

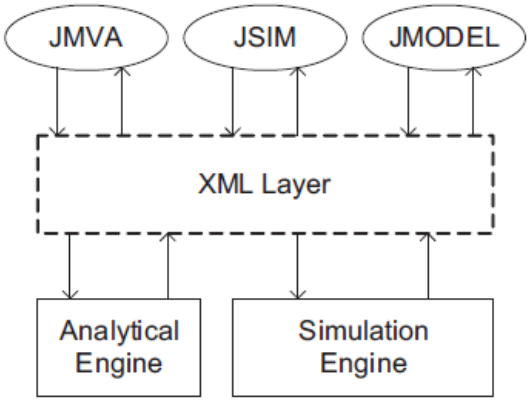

Figure 2.4: Scheme representing JMT JSIM XML structure

system.
The challenge of modelling is to define a general workflow to be applied for different systems but able to represent them correctly. All the steps for the definition of models and its application have been studied and defined at the end of the previous century [9]. The model has to be created with clear hypothesis and rules defining its field of application.

The new challenge is to define the connection between the system representation and the performance evaluation. Connection between an ontology based system and an JMT JSIM XML codification is not been defined yet. On the other side, there are tools for the extraction of data from the ontology exploiting SPARQL queries. For this reason a coding phase is required to create the connection once the data are obtained.
With this analysis concluded it can be possible to move to the problem statement phase of chapter 3, where all the hypothesis and rules of the modelling are defined according to the knowledge collected in this chapter.

Chapter 3

Problem Statement

In this chapter it defined the model for the system representation. This model grounds on a set of rules and hypothesis that are defined by an analysis on the problem presented in Chapter 1.

The chapter is organized as follow: analysis of the problem and identification of the issues that requires modelling hypotheses, production system issue analysis (Section 3.1), buffer issue analysis (Section 3.2) and buffer and machine relationship analysis (Section 3.3).

The starting point of the modelling is the structure that defines the data representation in the tools used for the physical system representation and for the performance evaluation.
A production system in order to be suitable for this model has to respect the following assumptions:

1. It must represent a flow shop, hybrid flow shop or a job shop;
2. All machines assigned to each process step can be visited with the right sequence from the part types, thus, the production plan has to be feasible with respect to the physical system;
3. each machine has a dedicated buffer;
4. The product of a process plan strictly follow the flow defined, there are no split, for example reworking.

OntoGui product and physical representadio present an ontology-based formalization which can be find represented in Figure 3.1 [10]. In particular, the modules represent:

- IfcTypeProduct: Definition of part types;

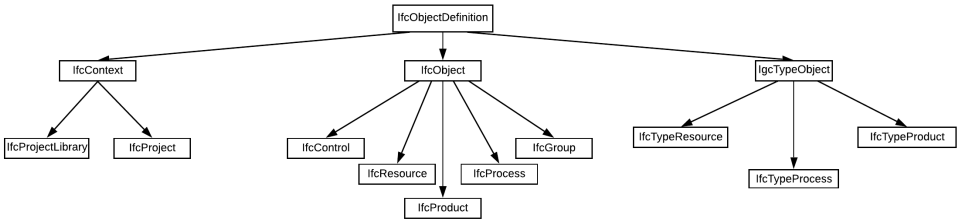

Figure 3.1: OntoGui ontology structure

- IfcTypeProcess: Definition of specific process plans;
- IfcProcess: Definition of specific production process steps;
- IfcProduct: Definition of factory elements like Buffer elements, Machine tools, Pallets;
- IfcGroup: Definition of transformation systems;
- IfcControl: Definition of a work plan (e.g. production plan) to control the behaviour of a system.

From this structure, which can also be defined as process-based, the physical system and the products are defined separately. The reason of this formalization can be found in the high interoperability that it grants. This tree structure allows the data to be integrated but on the same way to maintain their original representation.

On the other side JMT JSIM adopt a flow based formalization which results in a merge between the process plan with the physical system. This structure is easier for an user to be understood and applied but limits the possible exchange of data with other tools. This difference between the two formalizations makes necessary the process of create a model to move from one to another. The differences requires definition of hypotheses and rules to limit and approximate the system that can be represented from the model.
It is needed to start from the idea that is at the base of the different representations. From one side there is the ontology, a way to create flexible data model by integrating various knowledge domains without loss for any individual one. On the other side there is a queuing network structure with a very limited flexibility, also due to the presence of a user interface with increase the restrictions. This difference can be seen by looking at the data representation of ontology from Figure 3.1 and the structure of queuing network in Figure 3.2

Before continuing it is important to understand how the the creation of a production system is defined in OntoGui. This step will help the reader to have a clear idea on the following steps, please refers to Figure 2.1 for a view on the user

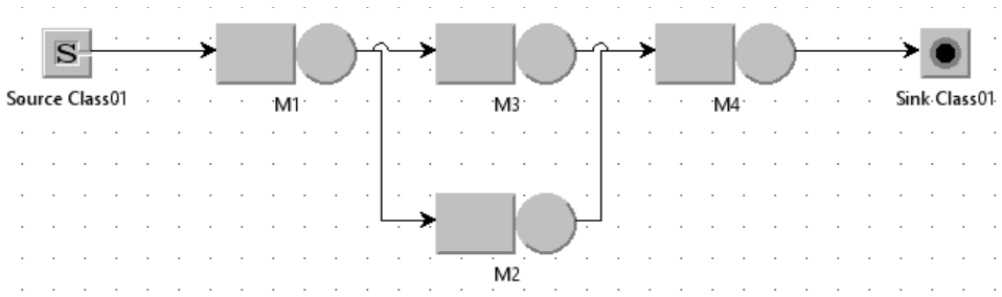

Figure 3.2: JMT JSIM representation open hybrid flow shop single class

interface. The definition part can be defined into two parts, the first one defines the creation of a library for the production plan, while in the second defines the creation of a subfolder, where the previous library is imported, where the physical system is designed and the process steps can be assigned to the physical machine.

A complete production plan definition is composed into three main steps: the part type is created, a process plan assigned to the part is created, process steps of the plan are defined and for each of them it is defined the order and the processing time. In order to define the physical system the subfolder has to be created and the previous library imported; at this point it is possible to define each buffer and machine element in the system with their parameter such as buffer size, and define the physical connection. Last step is to assign each process step to the machine needed in order to define the connection between the physical system and the process plans. For this reason, what really connects the product and the physical machines is the assignment of a precise machine, or multiple ones in case of parallel processing, to a process step of a determined process plan to produce/assemble a part type. This representation of a product defines the ontology as process plan based.

Thus, the real link between the two entities, physical system and production plan, is the machine assigned to the process step. As it is discussed later this definition is critical and must always be considered during the modelling process. The concept expressed in the previous lines is depicted in Figure 3.3, where the ramification clearly shows the two part of the ontology connecting trough the machine assignment. Excluding that point, there is no other connection, thus, due to this logic, the buffer has no product assignment.

This representation has a great impact on the modelling that will be developed in the following steps, needing the creation of precise rules and hypothesis on the representation of the buffers. Moving to the queuing network, before going into detail it is important to focus its main characteristic. A queuing system is defined by one or more “customers” waiting for a service by one or more ”servers”. Queue forms due to a not perfect balance between the arrival rate of the customers

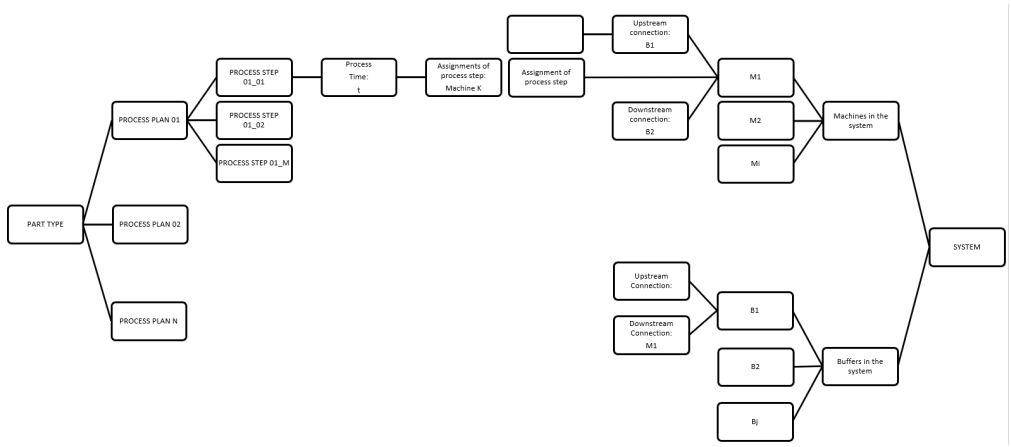

Figure 3.3: Ontology structure for system definition

and the service rate of the service. An example can be found in Figure 3.4 A

Figure 3.4: Example for a queuing network with the server

queuing network has different characteristic in the analysis: the number of the population, the arrival process, the service process, the queue configuration and the queue discipline. For some of these parameters there is the need to create some assumption which are defined along the chapter.
Due to this different structure, process plan is assigned differently to the physical system. The user must define the "class" which represent the products, and the physical production system. At this point the assignment of the production/assembly processes to the machines is direct, there is no process plan to define.

As explained in Chapter 2, in this kind of network each station is composed in 3 main parts: the queue section, the service section and the routing section. The first one is responsible for the buffer of the station, the second one to the processing time and the last one to the direction that each product is taking after leaving the station. For this reason, the definition of the physical system is merged with the one of each part type and its process plan.

The addition of the process step characteristic within a station parameter represents a critical change compared to the Ontology definition. There is a merging of information that in the original representation are kept separated and finds the only point of contact through the assignment of the machine to a single process step. In this way, a queuing network eliminates the need of defining a

process plan separately. This define that the new representation necessary for the performance evaluation is Flow Based which simply means that for each product it is necessary to define all the machine visited with their processing time and set the routing of the part types from machine to machine.

The new radical change is the basis of the modelling of the new system. The first objective is to understand what needs to be limited passing from an open and flexible model to a strict one. Following this which information are necessary to operate from production plan based to flow based system.

3.1 Analysis of the production system

In this section, the first problem related to the difference between a flow-based logic and a production plan logic is analysed. The two different data structures have a different system definition and it is needed to find an approach to pass from first to the second.It has been seen how the two different codification change the system definition. In a practical way, moving from the first to the second means starting from a product with its production plan and obtain a list of all the machines visited from the product. Doing this for all the part type of the system completes the process.

These two different approaches also change the way to define the type of production system. Let us start from the hypothesis that all the part type defined in the same ontology library are assigned to the same physical system and there it is not possible to define the system without one of them. With this approximation, a production system is composed of: part types and their process plan and a physical system where the process steps are assigned to. There can't be any process step that is not assigned to an existing machine. System with more flexibility such as open shop can be easily distinguished from more rigid one such as flow shop, due to the presence of multiple following process steps. For other cases, without the definition of the production system it can't be possible to operate this distinction. For example, a flow shop and a job shop from only the point of view of the process plan cannot be defined, it is its assignment on physical machines that creates the type definition. On the other side in the queuing network the physical system defined together with the process steps, so there is no space for flexibility. These is no possibility to assign the process steps of a "class" to another system.
Moreover, system with less constraints, such as open shop, grants higher flexibility but this flexibility requires a wider analysis of the ontology structure. For this reason, in order to start the analysis with more focus, it has been defined another hypothesis which limits the field of application of the model: only systems that can be included under the categories of flow shop, hybrid flow shop (flow shop with

parallel machines) and job shop without parallel machines are considered. Opening the definition to more complex system would increase greatly the definition of the model. Once the model is defined it can be possible to go back and remove some constraints and start again a new analysis.
Another issue that need to be faced in before continuing with the in-depth analysis of the XML input file, regards the buffers station of the physical system.

3.2 Analysis of the buffer

This problem comes from the different concept of buffer between the Ontology defined structure and the queuing network. In OntoGui the buffer is an independent element of the production system, completely separated from the machines, on the other way in queuing network there is no real representation of the buffer. The buffer is represented as the queue section inside a station. Within the same station, as wrote before, there is both the queue and the service section, where the last one represents the machine in the Ontology. This duality of a station creates a great issue to be solved.

Going back to the concept of production plan and flow-based representation, it's critical to understand how the buffer issue can be faced in these two different models. Starting from the ontology, the part types are connected to the physical system through the process steps which have the machines assigned; on the other side there is no real direct connection with the buffers. It is no possible to act on the linking between a buffer and a product. This relationship can be easily deduced in case of simpler production system such as flow shop; in fact in relation to the this system, if a buffer is connected downstream a machine and upstream another, and both of them are visited in succession from a certain part type, this means that also the buffer is being crossed from the same product. In cases where there is more than one buffer between two machines, it is not feasible to claim that the path of the part types crosses the buffer.
This lack of information is a cause of great uncertainty which need to eliminate for the fact that the model to be developed is going to be applied from a program, which is not able to reason like a human mind in case of ambiguous situation like this one. First of all there is to consider that the model is already limited to production system such as: flow shop, hybrid flow shop and job shop which are already limiting the variety of buffer configuration.

Starting from this point it is important also to point out again that the ontology is not containing any information on how buffers are being assigned to products. For this reason, even if it was possible to convert with any change the production system it would still be a certain level of lack of data for a correct and reliable

performance evaluation. Due to this issue, it is easier to have a new hypothesis that there is only one buffer before each machine, this solution is also going to become handy in following problems. This new rule can look like a limitation but considering the constraints we already have and the lack of parameters in the ontology, it has a limited impact, with great improvement in term of performance evaluation and on the reliability of the model within its hypothesis.
The definition of a single, machine-dedicated buffer puts a limit in the use of this model to already defined ontology production system. This approach let the user define its own approximation so that the results can be as closest as possible to its own point of view in the buffer management.

Within a buffer station there are various possible strategies that can be implemented. The first one is the queue policy, so how products in queue should behave while waiting in the buffer. It can be defined into two levels, a general station level and a part type level. For the station specific level, it represents the general queue policy without distinction between classes of product. For this policy the best general approach is the non-pre-emptive scheduling for all the queue element. It grants a better modelling for the production system considered in the hypothesis already made.
Going to focus on the single part type it can be possible to define a queue policy also for the single product. Due to the lack of this definition in the ontologies taken into analysis and the need of simplify the initial model, it is decided that the best option is to use for all buffer a FCFS logic. The policies chosen represent the most common type for the production system analysis, so these hypotheses still grant the most general approach as possible in terms of queue policies.
Another parameter to analyse is the "drop rule", so how each station should behave with the products within it when there is a blocking situation in the system.
It is important to define this parameter since it has a great effect on the system performance. Discarding a product other than keep it waiting in the queue until the next one has space again can have multiple impact such as on throughput and time spent in the system. Considering the kind of system it is being modelled, it can be easily excluded a rule based on dropping the product out of the system such as it could be modelled a real person queue for a service provided. The logic to follow is to be able to represent at the best the majority as possible of the production system that respect the boundaries expressed by the hypothesis. For this reason, the best rule to apply would be the block after service one. In this way, the part types that have just finished their operation in a station must wait in the service station until the end of blocking if they find the buffer of the following station full.

After a definition of the main hypothesis for the buffer part of a station there is the need also to define some rules for the processing. In the ontology, it is possible to define a wider set of processing rules thank to the fact of the great versatility of the structure, on the other side from the queuing network the regulation can be

more imiting. The machines are considered to process only one product per time, so each process step has 100% usage on each station, considering a processing time which is always load independent and that can be defined only trough deterministic or exponential distribution. The last hypothesis is simply defined in order to develop a model that is able to consider more than one time distribution, so a being able to recognise the kind of input, but to keep easier the creation of it.

3.3 Analysis of the relationship between

The last step of this first part of analysis is focusing on the difference between the definition of the machine and the buffer for the two models. To recap, in the ontology defined structure the machine is considered independent from the buffer, on the other side for the queuing network the two elements are merged into one represented from a queue. Through some specific modelling that is explained in detail for JMT JSIM tool it can be possible to represent also in a queuing network the machine separated from the buffer. Since there is the possibility to merge the two element or keep them separated it is useful to think about the possible implication of this choice, later this analysis can also be supported with some testing.
The merging strategy would prove to be difficult considering all the possible existing production systems, and the infinite combination of configuration would make the assignment of the right buffer size to the dedicated machine hard. Instead, thanks to the hypothesis defined in the previous steps, by creating a unique buffer before each machine, the fusion process is direct with no space for mistakes.
On the other side, by keeping the original ontology structure, a modification of the system definition to split the element is required. This change would go against the basic hypothesis of a queuing network, which obviously define the queue as basic structure, possibly creating issue in the performance calculation. In conclusion, leaving the station divided the system is reflecting better the ontology structure, on the other side the merged structure is more faithful to the queuing network structure. We choose to adopt the queuing network merged structure. The creation of the previous hypothesis already grants a great applicability of this merging solution, hoping in the most accurate performance analysis.

In the next chapter with the definition of the single section of a queue it is also analysed this last choice by testing to make sure that it is the best one for the model definition.
Once these initial hypotheses have been defined, the system application field where the model is applicable has been limited. This reduction comes at the cost of excluding those systems that presents very few constraints in terms of product

routing, such as open shop, or those that presents a not simple buffer configuration. The rules have been set to guarantee a better reliability of the model to the production system the user want to analyse.
The next step of the process is the development of the real resolution model. From a deep analysis of the XML file, all the different JMT JSIM structure are going to be coded and the whole logic to move from the ontology to the simulation will be defined.

Chapter 4

Solution

This chapter deals with the effective application of the model defined in Chapter 3 in order to develop the connection between the ontology-based system and the queuing network.

The creation of a connection between the two environment and the automatic generation of the model is developed through different steps depicted in Figure 4.1

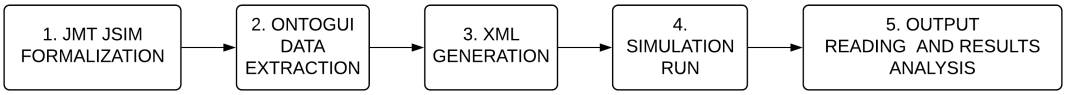

Figure 4.1: Solution summary

Section 4.1 addresses the analysis of the XML input file for the queuing network. All the needed JMT JSIM elements are analysed and their XML codification is defined. In this way, it is possible to understand the data needed for the definition of the simulation model.
In Section 4.2, the process of the data extraction from the ontology is analysed, defining what are the possibilities and describing the whole process to obtain all the information needed.
Section 4.3 deals with the XML generation. All data obtained from the ontology are used to generate the different elements needed to define the queuing network model.
Section 4.4 explains how the simulation works and how to set the necessary parameters.
Section 4.5 addresses the definition of an approach for the output reading process and their analysis, by explaining where are the results saved and how to automate the activity.

4.1 JMT JSIM Formalization

After the definition of the problem to be faced and the basic hypothesis, it is needed to understand how the different elements are defined according to the ontology and the XML logic.
It is needed to operate a process of "reverse engineering" to understand how the various information collected in the ontology are represented in the queuing network. After this part, the actual "conversion" is operated. Due to the different codification of knowledge, once both are defined, the aim is to understand how to move from the ontology to the XML. This process has been thought to be the most efficient share the content of the modelling.
The first part of the chapter is dedicated to describing the process of getting an empirical definition of the multiple structure needed in XML input format.
On the JMT documentation [8] it is not possible to find any XML codification of the different element on JMT JSIM; this lack of information leads to the choice of personally develop a codification for each structure is needed.
This long process can be conducted by defining single stations, saving the file and read the output which contains the XML structure. Step by step, increasing continuously the complexity of the structure, it can be possible to understand the rules that define the elements.
The analysis has been limited only to the feature of the tool needed for our application.
The process to obtain the needed information from the ontology is explained in Section 4.2, from now on it will be considered as if everything needed is immediately available.

In the following subsection it is described how each element needed in JMT JSIM can be represented with the XML input. For each element described it is also presented their representation through the JMT JSIM graphical user interface to give the reader a better understanding.

4.1.1 Station definition

4.1.1.1 Queue

The main element in JMT JSIM is the "Queue". A Queue is a station which includes a queue section, service section and routing section.
The queue section is the part that represent the buffer of the station. In this part it is possible to set the capacity (the buffer size) and the queue policy. The first one can be set to infinite or to a finite number; for this modelling it is important to set this number equal to the buffer size plus 1. This setting is due to the fact that for a queuing network the buffer size is not equal to the maximum number of

customer but it is smaller of one unit, in fact the number of customers represents the one waiting in the queue and the one that is being serviced. In the queue policy tab, it is possible to select a non-preemptive scheduling or a preemptive one; according to the hypothesis it is considered to use a non-preemptive policy. Once this assignment is concluded, the class specific queue policies must be set.

- FCFS: first come first served;
- LCFS: last come first served;
- RAND: random;
- SJB: shortest job first;

According to the assumptions presented in Chapter 3 the standard queuing policy is set as FCFS.
Moving to the next policy there is the drop rule; for this parameter it has been already be assumed that all stations follow a block after service logic, where here it is defined BAS Blocking.
Next section of the station is the service station. In this section it is possible to set the policies in relation to the processing activities for the products in the machine. It can be set the number of servers, in case of parallel machining for two identical machines, and service time distribution. For each class of products, it is possible to set different strategies:

- Load independent;
- Load dependent;
- Zero-time services;
- Disabled;

According to the assumptions of Chapter 3 a load independent strategy has been decided as standard.
The next and the last section of a Queue is the routing section. This section is the responsible of directing the product that has finished its process activity to the right following station. Between the set available on JMT, here are defined the two routing policy used in the modelling:

- Random: “with this strategy, jobs are routed randomly to one of the stations connected to the routing device”;

- Probabilities: “with this algorithm you can define the routing probability for each outgoing link”;

The random routing is used for all the stations with only one possible destination, in fact the implementation of random routing is the easiest one and where the path is already constrained by the connections, such as flow shop, there is no need to develop a more detailed routing strategy.
On the other side "Probabilities" strategy requires a deeper analysis to understand how it can be used for the product path where there is more than one choice, such as hybrid flow shop or job shop.

4.1.1.2 Source and Sink

Once the buffers and the machines are defined, it is needed to understand how the products flow through the system. In queuing network there are two types of class to define: open and closed. Each one requires a different configuration of the production system. In particular, the open one needs the definition of two additional stations: Source and Sink.
The sink is the element generating the part type arriving in the system according to a determinate arrival time. The sink on the contrary is the station where parts exit the production system. In term of parameters these two stations are very limited, indeed, for the source it is only possible to set the routing policy. The arrival time is set in another tab, where is possible to set all the classes present in the system. The Sink has no possibility to have any customization since it is the last station visited and the part types after that are disappearing from the system.

4.1.2 Class Definition

Next step is to understand how the products can be defined within the production system. In “Define customer classes” it is possible to set the part types. For each class it is possible to act on some different setting according to the type: open or closed. For both it is possible to set the priority and the reference station, which represent the station where the product is starting the routing in the system. In general, for all open classes it is necessary to set the arrival time according to the preferred time distribution; as the service station for the model developed the choice of time distribution is limited only to deterministic and exponential. Regarding the reference station, as wrote in the paragraph before, for all open classes it must be set a source. On the other side for closed class, it is needed to set the population, which represents the number of part type in the system, and the reference station as the first machine of its production plan. This kind of class

can be used to represent these systems where the products that are produced or assembled are moving on a pallet. Thanks to the pallet, it is possible to consider the pallet itself at the part type that is undergoing the processes. In this way the class is considered closed since the pallet in the system are limited. In case that their number is high it can be useful to analyse if it has any effect on the performance and, if not, consider the class as open.

4.1.3 Performance Evaluation

Last point to be analysed is the reason for the creation of this model: the performance evaluation.
In the ontology codification there is no direct correspondence between the possible performance to evaluate and the one that are available in JMT JSIM. So, for the sake of a general application it has been decided to include the automatic evaluation of certain performance if the user requires. The performance to be evaluated are:

- Response time per station;
- Queue time per station;
- Utilization per station;
- Throughput per station;
- System throughput;
- System response time;

The choices should be able to give the user a clear view on the system general performance. In case of additional needs, it is the duty of the user to manually add the needed one through the user interface.
A performance is chosen from a set of predefined set including the most elementary ones such as station throughput, queue time and system response time. The definition is possible from the tab “define performance indices”. The performance that includes “system” in the name are system specific and can be set for the evaluation of a specific class or all. All the other performance, except those which have a name of a determined element in the name such as sink or fork, can be applied at a single station by specifying it under “Station/Region” column. Moreover, it can be possible also to set the confidence interval and the maximum relative error of the simulation.
Once the structure of the JMT JSIM element is defined and the use and parameters definitions of each of them are clear, it can be possible to start the XML reading for the codification of the elements’ representation. Only with a perfect idea of

the features of the different elements and their functioning it can be possible to understand the XML structure.

4.2 Data extraction from ontology

This section explains the process of data extraction from the ontology, these data are used in next section (Section 4.3) to create the XML model. The use of the different tools is presented for the process and the final elaboration is realized with Python.
The process is composed by several steps, the first is the "Ontology preparation" where it is required to find a feasible way to access to the ontology, the second step is the "definition of the queries" in order to extract the information needed. The third step is the "extraction process", where a tool using the query to extract the information has been developed using Python language. The last step is the "refinement of the data" obtained in the previous step. Trough this last step, the data are converted in another representation suitable for the XML definition.
In figure 4.2 the scheme for the extraction of data from the ontology is depicted.

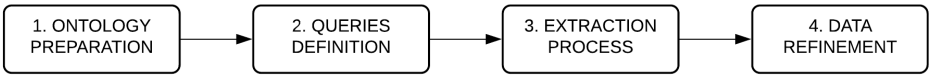

Figure 4.2: scheme representing the data extraction process.

4.2.1 Ontology preparation

This step deals with the preparation of the ontology to the data extraction. As explained in the chapter introduction it is important to define a way to access to the ontology. Different ways are possible which can be resumed into two main alternative: direct access on a local ontology or remote access. Each one of the two possibilities has its own point of advantage and its weaknesses.
Regarding the local access, its benefits are correlated to the simplicity of use. A local ontology allows the user access to its information without the need of external support such as internet connection and remote servers. The data are always accessible when working in local and protected from external unwanted access.
On the other side a remote access gives the user the possibility to work on the ontology from everywhere with the only need of an internet connection. Moreover, the use of a server can reduce the risk of a data loss since they can be stored with

multiple backup. The most important aspect of this possibility it that it allows also the use of a remote machine for all the processes required for the performance evaluation of the required system.
Following the analysis of the pros and cons defined before it has been decided to follow a remote access approach due to the vision of a future remote computing of all the processes. Moreover, uploading the ontology on a server allows user from all over the world to access them improving opportunities of team work. The use of a remote server requires the search for the right tool for this operation. Ontogui has been developed with an add-on to include an integration with Stardog [11] which is a commercial RDF database, which allows SPARQL query, transactions, and OWL reasoning support. For our use Stardog gives the possibility to create databases on a remote server where to save the ontologies and to make SPARQL queries for the data extraction process this add-on gives the user the possibility to upload the ontology directly on Stardog. The possibility to access Stardog directly from Ontogui enables the users to access directly to the ontology trough the user interface they already knows.
For the first phase it is needed to create a database on Stardog server and upload the ontology needed.

4.2.2 Queries definition

The ontologies, as the RDF database, are particularly suitable to a reading through SPARQL query. SPARQL [12] is a semantic query language for database which allows the user to write queries against data that follow RDF specification.
This parts regards the real definition of the SPARQL queries that are developed for the extraction of the needed data for the XML development.
First it is needed to understand what are the needed data, after that different type of queries have been developed for the different cases. Some information require more than one SPARQL query while others can be directly obtained with a single one. Before continuing please refer to the Figure 4.3 to see the ontology structures and the modules it is composed.

In the following, we report an explanation of all the pre-defined modules: [13]

- statistics: basic concepts about probability distributions and descriptive statistics. (http://www.ontoeng.com/statistics). [14]
- fsm: basic concepts to model a Finite State Machine (http://www.learninglab.de/ dolog/fsm/fsm.owl). [15]
- sosa: Sensor, Observation, Sample, and Actuator (SOSA) Core Ontology (http://www.w3.org/ns/sosa/). [16]
- ssn: Semantic Sensor Network Ontology (http://www.w3.org/ns/ssn/). [17]

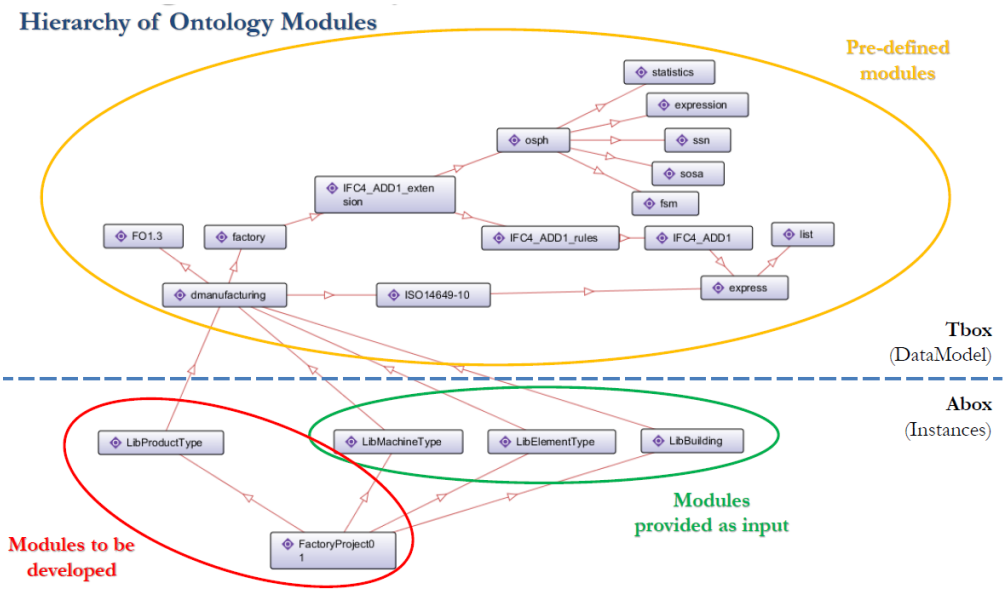

Figure 4.3: Hierarchy of Ontology Modules

- expression: formalization of Algebraic and Logical expressions (http://www.ontoeng.com/expression). [18]
- osph: ontology modeling Object States and Performance History, while integrating fsm, statistics, ssn, sosa, expression (http://www.ontoeng.com/osph). [19]
- list: ontology defining the set of entities used to describe the OWL list pattern. (https://w3id.org/list). [20]
- express: ontology that maps the key concepts of EXPRESS language to OWL (https://w3id.org/express). [21]
- IFC_ADD1: ifcOWL automatically converted from IFC_ADD1.exp .
- IFC_ADD1_rules: add class expressions to ifcOWL derived from WHERE rules in IFC_ADD1.exp .
- IFC_ADD1_extension: integration of modules and general purpose extensions of ifcOWL.
- factory: specialization of ifcOWL with definitions related to products, processes, and systems.
- dmanufacturing module further specializes the industrial domain of discrete manufacturing.

- powertrain: ontology that defines specific concepts for the powertrain domain.

In every SPARQL query, it is needed to define the prefix defined in Listing 4.2

```
PREFIX list: <https://w3id.org/list\string#> \\
PREFIX express: <https://w3id.org/express\string#> \\
PREFIX ifc: <http://ifcowl.openbimstandards.org/IFC4\string_ADD1\
   string#> \\
PREFIX factory: <http://www.ontoeng.com/factory\string#>\\
PREFIX dm: <http://www.ontoeng.com/dmanufacturing\string#>\\
PREFIX osph: <http://www.ontoeng.com/osph\string#>\\
PREFIX fsm: <http://www.learninglab.de/~dolog/fsm/fsm.owl\string#>\\
PREFIX ifcext: <http://www.ontoeng.com/IFC4\string_ADD1\
   string_extension\string#>
PREFIX ssn: <http://www.w3.org/ns/ssn/>\\
PREFIX sosa: <http://www.w3.org/ns/sosa/>\\
PREFIX stat: <http://www.ontoeng.com/statistics\string#>
```

Listing 4.1: SPARQL query prefix

In the following sub-sub-section it is analysed each needed SPARQL query, presenting also what parameters must be changed according to the modules of the ontology to be modelled

4.2.2.1 Physical system stations

The aim of this SPARQL query is to obtain data about the physical station of the system, like the machines and the buffer of the production system. The data needed are for each element to obtain:

- name of element;
- ame of physical system;
- type of element: MachineTool or BufferElement;
- downstream element.

In order to obtain these information it is developed the SPARQL query in Listing 4.2:

```
PREFIX LINES
select distinct ?sys ?elem ?class ?downstreamElem
FROM <http://ifcowl.openbimstandards.org/IFC4_ADD1>
FROM <http://www.ontoeng.com/IFC4_ADD1_extension>
FROM <http://www.ontoeng.com/factory>
FROM <http://www.ontoeng.com/dmanufacturing>
FROM <http://www.ontoeng.com/SUB_LIBRARY>
where {
# get systems
?sys rdf:type/rdfs:subClassOf* factory:TransformationSystem .
# get elements in system
?sys ifcext:hasAssignedObject|^ifcext:hasAssignmentTo ?elem .
# downstream connection
OPTIONAL{
?elem ifcext:isConnectedToElement|^ifcext:isConnectedFromElement
?downstreamElem .
}
# class
?elem rdf:type ?class .
FILTER ( ?class != owl:NamedIndividual ) .
}
```

Listing 4.2: SPARQL query Physical system stations

The only parameter that needs to be changed is the one defined as: SUB_LIBRARY which should be substituted with the ontology module name where the physical system is defined.

4.2.2.2 Buffer Size

The objective of this SPARQL query is to obtain the data related to the elements in the physical system and their buffer size. The information obtained are:

- name of element;
- name of physical system;
- type of elements: MachineTool or BufferElement;
- buffer size.

```
PREFIX LINES
select distinct ?sys ?elem ?class ?buffCap
FROM <http://ifcowl.openbimstandards.org/IFC4_ADD1>
FROM <http://www.ontoeng.com/IFC4_ADD1_extension>
FROM <http://www.ontoeng.com/factory>
FROM <http://www.ontoeng.com/dmanufacturing>
FROM <http://www.ontoeng.com/SUB_LIBRARY>
where {
# get systems
?sys rdf:type/rdfs:subClassOf* factory:TransformationSystem .
# get elements in system
?sys ifcext:hasAssignedObject|^ifcext:hasAssignmentTo ?elem .

# class
?elem rdf:type ?class .
FILTER ( ?class != owl:NamedIndividual ) .

# buffer capacity
OPTIONAL{
?elem ssn:hasProperty ?prop .
?prop rdf:type factory:BufferCapacity .
?prop osph:hasPropertySimpleValue ?buffCap .
}
}
```

Listing 4.3: SPARQL query Buffer Size

Also in for this SPARQL query, the only parameter to be changed is SUB_LIBRARY¦ that has to be substituted with the module of the ontology where the physical system has been defined.
With this SPARQL query and the previous one, the physical system is defined.

4.2.2.3 Part Types

The first step for the collection of the information regarding the production plan is to extract the part types present in the system together with their process plans. The data obtained are:

- name of part type;
- name of process plan of part type.

```
1    PREFIX LINES
2    select distinct ?parttype ?pplan
3    FROM <http://ifcowl.openbimstandards.org/IFC4_ADD1>
4    FROM <http://www.ontoeng.com/IFC4_ADD1_extension>
5    FROM <http://www.ontoeng.com/factory>
6    FROM <http://www.ontoeng.com/dmanufacturing>
7    FROM <http://www.ontoeng.com/LIBRARY>
8    FROM <http://www.ontoeng.com/SUB_LIB>
9    WHERE {
10   ?parttype rdf:type owl:NamedIndividual .
11   ?parttype rdf:type/rdfs:subClassOf* factory:ArtifactType .
12   OPTIONAL {?parttype ifcext:hasAssignedObject|^ifcext:
     hasAssignmentTo ?pplan . }
13   }
```

Listing 4.4: SPARQL query Part types

In this query, it is needed to define two parameters: LIBRARY and SUB_LIB. This duality is defined due to he fact that for the ontology definition has been chosen to define the part type and their process plan on an library ontology module. While the physical system is defined in an ontology module which is a InSubFolder that has imported the previous modules. So when both information are required, it is needed to select both the modules.
In the SPARQL query, LIBRARY word needs to be replaced with the name of the library module while SUB_LIB word must be substituted with the InSubFolder ontology module name.

4.2.2.4 Process Steps

Once the part types with their associated production plans have been extracted, it is possible to extract the data related to the single process steps of each plan.
The choice to collect these information in two different queries is due to the fact that by keeping them separated it is possible to make a specific query for a specific production plan. The data obtained are:

- part type name;
- name of process plan of part type;
- process step name of process plan;
- process step successor name.

```
1     PREFIX LINES
2     select distinct ?pplan ?task ?successor
3     FROM <http://www.ontoeng.com/LIBRARY>
4     FROM <http://www.ontoeng.com/SUB_LIB>
5     WHERE{
6     VALUES ?pplanstr {"http://www.ontoeng.com/LIBRARY#PROCESS_PLAN"}
7     BIND(URI(?pplanstr) AS ?pplan) .
8     # ?pplan rdf:type owl:NamedIndividual .
9     # ?pplan rdf:type/rdfs:subClassOf* ifc:IfcTaskType .
10    ?pplan ifcext:isNestedByObject|^ifcext:nestsObject ?task.
11    OPTIONAL { ?task ifcext:isPredecessorToProcess|^ifcext:
     isSuccessorFromProcess ?successor . }
12    }
```

Listing 4.5: SPARQL query Process Steps

In this SPARQL query, it is important to notice the PROCESS_PLAN parameter. As wrote, this is due to the fact that this SPARQL query is process plan, specific in order to collect the data in an ordered way. Since a process plan is part type specific in order to collect the data of all process steps this SPARQL query must be used for as many interrogation as the number of process plan replacing each time in place of the word PROCESS_PLAN the name of the process plan extracted in subsubsection 4.2.2.3.
The parameter LIBRARY and SUB_LIB are defined as the previous SPARQL queries.

4.2.2.5 Assigment of processes steps to machines

This SPARQL query is the used to collect the data representing the assignment of a machine of the physical system to the process step related to a certain process plan. The processing time is also defined with the help of this query. The production flow of a part type within the physical system is defined using this query, in the case the processing time is not defined as a deterministic value, a second SPARQL query is needed.
So from the data obtained in this step are:

- part type name;
- name of process plan of part type;
- process step name of process plan;
- process step machine assignment;

- time distribution of processing time;
- processing time or, for stochastic distribution, its URI;

```
PREFIX LINES
PREFIX osph: <http://www.ontoeng.com/osph#>
PREFIX fsm: <http://www.learninglab.de/~dolog/fsm/fsm.owl#>
PREFIX ssn: <http://www.w3.org/ns/ssn/>
PREFIX sosa: <http://www.w3.org/ns/sosa/>

select ?parttype ?pplan ?task ?timeDet ?timeStoch ?stochDistr ?machine ?durationDet ?durationStoch ?usage  # ?restype ?res ?machinetype

FROM <http://ifcowl.openbimstandards.org/IFC4_ADD1>
FROM <http://www.ontoeng.com/IFC4_ADD1_extension>
FROM <http://www.ontoeng.com/factory>
FROM <http://www.ontoeng.com/dmanufacturing>
FROM <http://www.ontoeng.com/FactoryProject01>
FROM <http://www.ontoeng.com/LibMachineType>
FROM <http://www.ontoeng.com/LibElementType>
FROM <http://www.ontoeng.com/LibProductType>
FROM <http://www.ontoeng.com/LibBuilding>
FROM <http://www.ontoeng.com/SUB_LIB>
FROM <http://www.ontoeng.com/LIBRARY>

WHERE{

  ## The selected process plan is a user-defined parameter
  VALUES ?pplanstr {"http://www.ontoeng.com/LIBRARY#PROCESS_PLAN"}
  BIND(URI(?pplanstr) AS ?pplan) .

  # get process plans
  ?pplan rdf:type owl:NamedIndividual .
  ?pplan rdf:type/rdfs:subClassOf* ifc:IfcTaskType .
  ?parttype ifcext:hasAssignedObject|^ifcext:hasAssignmentTo ?pplan .
  ?parttype rdf:type/rdfs:subClassOf* factory:ArtifactType .

  # get tasks in a process plan
```

```
34      ?pplan ifcext:isNestedByObject|^ifcext:nestsObject ?task.
35
36      # get default processing time (deterministic)
37      OPTIONAL{?task ifc:taskTime_IfcTask/ifc:
    scheduleDuration_IfcTaskTime/express:hasString ?timeDet.}
38      # get default processing time (stochastic)
39      OPTIONAL{
40        ?task ifc:taskTime_IfcTask/ifcext:hasStochasticDuration/osph
    :isQuantitySampledFrom ?timeStoch.
41        ?timeStoch rdf:type ?stochDistr.
42        FILTER ( ?stochDistr != owl:NamedIndividual ) .
43      }
44
45      # get resources where the task can be executed (it can be a
    resource or a resource type)
46        ?task ifcext:hasAssignedObject|^ifcext:hasAssignmentTo ?
    restype .
47      ?restype ifcext:typesObject|^ifcext:isDefinedByType ?res .
48      OPTIONAL{
49        ?res ifcext:hasAssignedObject|^ifcext:hasAssignmentTo ?
    machine .
50        ?machine rdf:type/rdfs:subClassOf* factory:MachineTool .
51      }
52      OPTIONAL{
53        ?res ifcext:hasAssignedObject|^ifcext:hasAssignmentTo ?
    machinetype .
54        ?machinetype rdf:type/rdfs:subClassOf* factory:
    MachineToolType .
55        ?machinetype ifcext:typesObject|^ifcext:isDefinedByType ?
    machine .
56      }
57
58      # get resource time and usage
59      OPTIONAL{
60        ?res factory:usage_ProductionResource ?restime.
61        # get resource consumption overriding the default processing
     time (deterministic)
62        OPTIONAL{ ?restime ifc:scheduleWork_IfcResourceTime/express:
    hasString ?durationDet . }
63        # get resource consumption overriding the default processing
     time (stochastic)
```

```
64        OPTIONAL{ ?restime ifcext:hasStochasticWork/osph:
    isQuantitySampledFrom ?durationStoch . }
65        # get resource usage
66        OPTIONAL{ ?restime ifc:scheduleUsage_IfcResourceTime/express
    :hasDouble ?usage . }
67      }
68    }
```

Listing 4.6: SPARQL query Process Steps Machine assignment

Also this SPARQL query is process plan specific.
The parameters that need to be replaced are LIBRARY, SUB_LIB and PROCESS_PLAN as defined in Process Step query.
For this query four more prefixes have been defined.
Otherwise, if stochastic distribution are are used to characterize the processing times, a second SPARQL query is needed to extract those values. This SPARQL query is described in the following section.

4.2.2.6 Process steps extraction with Stochastic time

The last SPARQL query is used to extract the exponential time distribution value. According to the assumptions defined in subsection 4.1.2, only exponential distribution are possible in the stochastic model. This query is only limited to the collection of the lambda for exponential distribution. So from the data obtained in this step are:

- URI name;
- stochastic distribution;
- exponential lambda value.

```
1   PREFIX LINES
2   PREFIX osph: <http://www.ontoeng.com/osph#>
3   PREFIX fsm: <http://www.learninglab.de/~dolog/fsm/fsm.owl#>
4   PREFIX ssn: <http://www.w3.org/ns/ssn/>
5   PREFIX sosa: <http://www.w3.org/ns/sosa/>
6
7   select ?distrib ?probDistrClass ?ExpLambda
8
9   FROM <http://www.ontoeng.com/statistics>
10  FROM <http://ifcowl.openbimstandards.org/IFC4_ADD1>
```

```
11      FROM <http://www.ontoeng.com/IFC4_ADD1_extension>
12      FROM <http://www.ontoeng.com/factory>
13      FROM <http://www.ontoeng.com/dmanufacturing>
14      FROM <http://www.ontoeng.com/LibMachineType>
15      FROM <http://www.ontoeng.com/LibElementType>
16      FROM <http://www.ontoeng.com/LibProductType>
17      FROM <http://www.ontoeng.com/LibBuilding>
18      FROM <http://www.ontoeng.com/SUB_LIB>
19      FROM <http://www.ontoeng.com/LIBRARY>
20
21     WHERE{
22
23       ## The selected process plan is a user-defined parameter
24       VALUES ?distribstr {"http://www.ontoeng.com/LIBRARY#
    STOCHASTIC_VALUE"}
25       BIND(URI(?distribstr) AS ?distrib) .
26
27       # get probability distributions
28       ?distrib rdf:type owl:NamedIndividual .
29       ?distrib rdf:type/rdfs:subClassOf* stat:
    ProbabilityDistribution .
30
31       ?distrib rdf:type ?probDistrClass.
32       FILTER ( ?probDistrClass != owl:NamedIndividual ) .
33
34       # Exponential distrib
35       OPTIONAL{ ?distrib stat:hasLambda_ExponentialDistr ?ExpLambda
    . }
36       }
```

Listing 4.7: SPARQL query Process Steps Stochastic

This SPARQL query is defined as URI name specific. From the SPARQL query in subsubsection 4.2.2.5 in case of stochastic value the output of the interrogation is the URI name of the numeric value. With this SPARQL query it is possible to interrogate the ontology on that specific stochastic processing time. Due to this specific request the ontology has to be interrogated through the SPARQL for all the URI name obtained in the previous step.
With this last step the queries definition process is finished and it can be possible to move at the extraction process.

4.2.3 Extraction process

This step involve the detection of the most appropriate way to extract the data needed through the SPARQL query defined in the previous step (see section 4.2.2). Since Python has been chosen as the main environement for developing the resolution, it is needed to find a way to implement the SPARQL query through it. The library SPARQLwrapper [22] has been chosen in order to integrate in the code the SPARQL query interrogation.
This library helps in creating the query URI and operating a results conversion allowing the user to easy manage the data. Regarding this last point SPARQLwrapper is able to change the output format into the most diffused one such as JSON, XML and N3 (Notation3). Between these solutions it has been chosen the JSON format.
The code developed to process the SPARQL interrogation on the ontology database is reported in Listing 4.8.

```
1
2 from SPARQLWrapper import SPARQLWrapper, JSON
3
4 sparql = SPARQLWrapper("QUERY_ENDPOINT")
5 sparql.setQuery("""
6     SPARQL QUERY CONTENT
7 """)
8 sparql.setReturnFormat(JSON)
9 results = sparql.query().convert()
```

Listing 4.8: SPARQL query code

The adoption of this library supports the extraction process and allows the user to focus on the design of the SPARQL queries. The only parameters which requires a modification are the QUERY_ENDPOINT, that exactly represent the endpoint where is required to make the interrogation, the query database URL and the SPARQL QUERY CONTENT where it is needed to insert the query defined before.
It is important to report a critical aspect of this library, it has not been possible to implement a remote SPARQL query to a database requiring authentication for the connection.
Using the Listing 4.8, it is possible to extract the needed data from the ontology, but these data are defined in JSON format. In the next subsection it is discussed how to refine these information in format suitable for the XML generation.

4.2.4 Data Refinement

The last step of the data extraction addresses the transformation of data from the JSON format to the data structures needed for the of XML generation.
In the following sections, for each SPARQL query described in 4.2.2, the format of the data needed for generating an XML file is presented and discussed.
It is defined a small case study so that it can be possible to see the output structure applied to a real case. The physical system is depicted in Figure 4.4, with a buffer size of 10 units for both buffers, the process plan associated is reported in Table 4.1

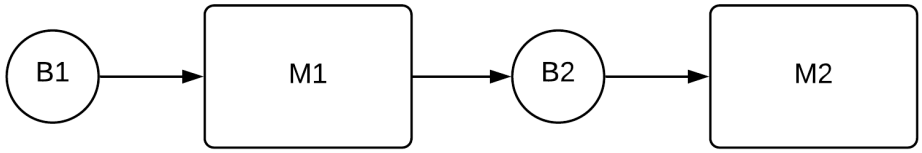

Figure 4.4: SPARQL query case study system representation

PART TYPES	PRODUCTION PLAN	PROCESS STEPS	PROCESSING TIME	TIME DISTRIBUTION	MACHINE ASSIGNED
Product01	ProcessPlan01	P01_01	1 s	Exponential	M1
		P01_02	1.1 s	Deterministic	M2
Product02	ProcessPlan02	P02_01	0.9 s	Exponential	M1
		P02_02	1.2 s	Exponential	M3

Table 4.1: process plan for SPARQL extraction case study

4.2.4.1 Physical system stations

The output needed is represented by two lists of lists:

- physical_station_list: a list of list, where each sublists contains the data about all the element in the system, the physical system name and the element type. The number of list is equal to the number of element in the system. It is strucutred as follow.
  [['M2', 'PhysicalSys01', 'MachineTool'], ['M1', 'PhysicalSys01', 'MachineTool'], ['B2', 'PhysicalSys01', 'BufferElement'], ['B1', 'PhysicalSys01', 'BufferElement']]
- physical_conn: a list of list where each sublist is composed of 2 element, the first one is the station considered, the second one is the downstream element.

The lenght of the list represent the number of connections in the physical system. The strucure is as follow:
[['M1', 'B2'], ['B2', 'M2'], ['B1', 'M1']]

4.2.4.2 Buffer size

The output needed is buffSize_list which is a list of list where each sub-list is composed of two items, the buffer element name and its buffer size. The length of the list represents the number of buffer in the system. It is structured as follow:
[['B2', '10'], ['B1', '10']]

4.2.4.3 Part types

The output needed represented by two lists of lists:

- product_classes:a list of list where each sub-list is composed of two items, the part type name and the process plan assigned. The length of the list represent the number of part type in the system. It is structured as follow:
  [['ProcessPlan02', 'Product02'], ['ProcessPlan01', 'Product01']]

- prod_list: : This list of list is generated following the definition of the previous one and its sub-list is composed of five element which has been defined only as parameters that need to be substituted. In fact the five element are: product name, ARRIVAL_TIME, 0, FIRST MACHINE, DISTRIBUTION. Only the product name can be defined, the others are replaced with the actual values in the following SPARQL queries. The third element represent the priority of the product but the feature has not been implemented so it is not changed. It is structured as follow:
  [['Product02', 'ARRIVAL_TIME', 0, 'M1', 'DISTRIBUTION'], ['Product01', 'ARRIVAL_TIME', 0, 'M1', 'DISTRIBUTION']]

4.2.4.4 Process steps

The output needed is process_plan_classes which is a list of list of list where each sub-sub-list contains the following elements: process step name, process step successor (if existing), process plan name and part type. Each sub-list represent a determined part type.
It is structured as follow:
[[['P02_02', 'ProcessPlan02', 'Product02'], ['P02_01', 'P02_02', 'ProcessPlan02', 'Product02']], [['P01_02', 'ProcessPlan01', 'Product01'], ['P01_01', 'P01_02', 'ProcessPlan01', 'Product01']]]

4.2.4.5 Assignment of processes steps to machines

The output needed is process_plan_machines_time which is a list of lists which follows the same structure list of list of list as process_plan_classes. Each sub-sub-list contains the following element: Part type, process plan name, process step name, machine assigned to process step name, time distribution and time value. Regarding the last value, as explained in Process Steps machines assignment (in section 4.2.4.4), in case of stochastic time distribution, it represents the URI of the time value which is used as input in the next query.
It is structured as follow:
[[['Product02', 'ProcessPlan02', 'P02_02', 'M2', 'ExponentialDistribution', 'd4b7fb22-29f7-4564-a115-04fc5245c574'], ['Product02', 'ProcessPlan02', 'P02_01', 'M1', 'ExponentialDistribution', '6d6dd362-7067-43e3-bc89-9daa2c905812']], [['Product01', 'ProcessPlan01', 'P01_02', 'M2', 'ExponentialDistribution', '4ad4966b-6781-499c-b86b-70cc16380b1d'], ['Product01', 'ProcessPlan01', 'P01_01', 'M1', 'ExponentialDistribution', '44934528-f9be-40c7-8fbe-ba4e75da5735']]]

4.2.4.6 Assignment of stocastic processes steps to machines

For this query the output needed is the same list defined in Process Steps (section 4.2.2.5 machines assignment process_plan_machines_time. Indeed, the aim of this data refinement is to find the time value for the stochastic distribution and substitute it in place of the URI.
It is structured as follow:
[[['Product02', 'ProcessPlan02', 'P02_02', 'M2', 'ExponentialDistribution', '1.0'], ['Product02', 'ProcessPlan02', 'P02_01', 'M1', 'ExponentialDistribution', '1.1']], [['Product01', 'ProcessPlan01', 'P01_02', 'M2', 'ExponentialDistribution', '0.9'], ['Product01', 'ProcessPlan01', 'P01_01', 'M1', 'ExponentialDistribution', '1.0']]]

The direct refinement of data obtained from the ontology . is concluded with this step It is needed to define other lists crossing the information in the output defined in this section.

4.2.4.7 Part type flow

Due to the change from a process plan-based production system to a flow-based production system. It is needed to define for each part type its flow in system. This means to define the machine visited and their processing time. For this reason, a script that takes as input process_plan_machines_time and filters the data in order to obtain a structure feasible for the XML definition has been developed. The output is a list of list defined as product_process, structured as follow:

[['Product02', 'M2', '1.0', 'ExponentialDistribution', 'M1', '1.1', 'ExponentialDistribution'], ['Product01', 'M2', '0.9', 'ExponentialDistribution', 'M1', '1.0', 'ExponentialDistribution']]

4.2.4.8 Buffer and machine merging

The data obtained from the ontology represent a system defined of both buffer and machine, while the model definition is based on a queuing network, where the two elements are represented from the queue. It is needed to define an approach to define this redefinition from the two data structure.
The merging of a buffer and a machine can be defined easier thanks to the use of hypothesis which define that for each machine there must be maximum one buffer and that each buffer is specific for a single machine (section 3.3. With this rule, it is easy to merge the two different element, since the queue element can be defined with the information coming from a machine an its buffer upstream, if present.
In order to apply this hypothesis two list have been defined:

- machineSize_list: represents the maximum number of customer of the new defined queue (buffer size + 1). It is structured as follow:
  [['M2', 6], ['M1', 6]];
- physical_Mergedconn: represent the new connection of the physical system once the buffer are excluded. It has the same structure as physical_conn (subsubsection 4.2.2.1):
  [['M1', 'M2']].

4.2.4.9 Arrival time and number of population

Last missing data for the definition of the XML is related to the part types. According to the type of system analysed, it is needed to defined the arrival time of the products in the system, in case of an open one, or the number of products present in the system, in case of a closed one.
These two parameters are defined in prod_list. Since in the ontology this value has not been defined, it is required a manual input from the user. This step must be completed before the XML creation. For this reason it has been decided to include it in the data refinement phase.

4.3 XML Modelling

JMT JSIM queuing network representation can be realized through the GUI by defining placing the element needed and setting their parameter in a user friendly environment. The system defined, once it is saved, is stored in a file with .jsimg

format where it is formalized with a XML structure. This file acts both as an input and as an output, for this reason it is possible to create the XML input file direct with an algorithm. The possibility of use the XML as both input and output creates the opportunity to learn the element codification by defining it through the GUI and see how it is represented in the output.
The XML file is composed of the following parts:

- heading;
- source definition;
- queue definition;
- sink definition;
- connections definition;
- performance to be evaluated;
- preload;
- ending.

All the part are defined in details in the following sub-sections.
Before continuing with the definition of the XML section it can be useful to understand the method used for the definition of the XML codification of each section. The learning process is constructed as follow: each station analysed in the previous paragraph is inserted alone in a new model with a single class defined, and the file is saved in its .jsimg format. At this point the file is opened as a text file so the XML codification can be easily seen. With the first observation, it is possible to see what is the general definition of the station is, for example how the main parameters are encoded. Once these steps are concluded it is possible to increase the complexity of the system by adding more classes. This part is critical since for each station there are parameters and policies specific for each class, then it's of the utmost importance to understand how the codification is defined when there is one than one class processed in a queue.
To conclude this analysis, it is needed to consider the production system as a series of machines, then the next step is to definite a series of station connected to each other. The connection between queue is also a part that is needed to be defined.
This process has been completed for all the elements needed for the model defined. There are other stations present in JMT JSIM that have not been analysed because the XML input coding was not needed for the model considering the hypothesis defined.
Moreover there is to define the class processed in the system and in case of open

system, the elements for their generation and ending: the source and the sink. All the steps are described in the following section with greater detail where each XML string has given a specific name. For a better visualization of the XML generation process refers to Figure 4.5. The image explains the flow to follow in the definition of the XML input referring to name of the part of code needed

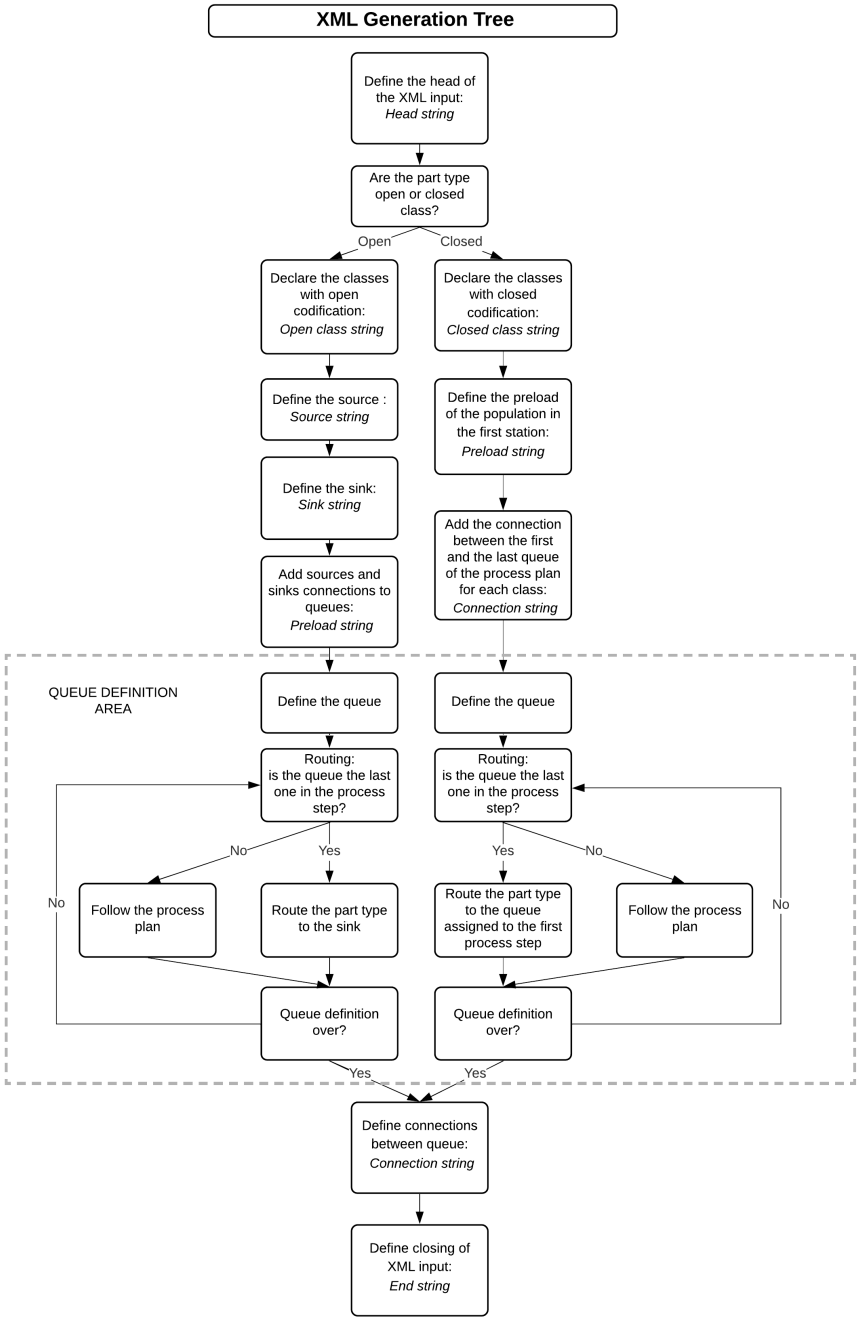

Figure 4.5: XML generation process

4.3.1 Heading

The first part of the XML file is composed by the heading. This part defines the XML codification, explicating the version, the ISO standard followed and other series of information. This part needs attention during for its generation since it contains the date and time of the last save of the file and the information regarding the file path. In the first two lines where it is needed to define the name of the file, as the FILE_NAME and the time and date according to a 24-character string of the following form: Thu Nov 7 23:32:38 2019.
In line 3 it is needed to set the following settings:

- directory path of JMT: replace this information in DIR_PATH_JMT;
- number of maximum samples for simulation: replace this number in MAX_SAMPLE;
- file name: replace the file name in FILE_NAME.

```
1 <?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
2 <archive xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="
    FILE_NAME.jsimg" timestamp="DATE_TIME"
    xsi:noNamespaceSchemaLocation="Archive.xsd">
3 <sim disableStatisticStop="false" logDecimalSeparator="."
    logDelimiter="," logPath="DIR_PATH_JMT" logReplaceMode="0"
    maxSamples="MAX_SAMPLE" name="FILE_NAME.jsimg" polling="1.0"
    xsi:noNamespaceSchemaLocation="SIMmodeldefinition.xsd">
```

Listing 4.9: head string

The second part of the heading includes the definition of the classes of products. For each class all the information that were present in “Define customer classes” in JMT JSIM , except the arrival time, must be defined following the “class string” structure:

```
1 <userClass customers="NUMBER_CUST" name="CLASS" priority="PRIORITY"
    referenceSource="SOURCE" type="closed"/>
```

Listing 4.10: Closed class string

```
1 <userClass name="CLASS" priority="PRIORITY" referenceSource="SOURCE"
    type="open"/>
```

Listing 4.11: Open class string

Where:

- NUMBER_CUST: number of products in the system. Only for closed class;

- CLASS: name of the part type;
- PRIORITY: priority;
- SOURCE: station or source where the class is starting the process.

To define the heading it is necessary to have a list where there are all the class of the production system and their characteristic defined in the bullet points.

4.3.2 Source

We consider to use a single source element for each open class generated. In this way, it is be possible to add other stations between the source and the first station without influencing all the other classes.
The station is composed in two parts: RandomSource which represents the arrival of the product, and the Router, which is responsible for the product path.
The codification of the source starts with the definition of the name of the station. Due to the hypothesis that for each source there can generate only one part, the service strategy parameter, and later also the router one, does not need reiteration for more than one class. In "RandomSource" section, it is needed to define the part type that is generated in the source and its arrival time, in Listing 4.12 it is possible to see the exponential encoding. The next section of the station is the router. In this case, even if there is only one class in the source, it has already been implemented the probability routing coding. Considering the presence of only one part type it can also be applied a random routing logic since the only connection possible it to the first machine of the process plan, but in this way it can be possible to implement easily in the future a single source for multiple class if needed. Here is the source definition string.

```
<node name="SOURCE_NAME">
<section className="RandomSource">
<parameter array="true" classPath="jmt.engine.NetStrategies.
    ServiceStrategy" name="ServiceStrategy">
<refClass>CLASS</refClass>
<subParameter classPath="jmt.engine.NetStrategies.ServiceStrategies.
    ServiceTimeStrategy" name="ServiceTimeStrategy">
<subParameter classPath="jmt.engine.random.Exponential" name="
    Exponential"/>
<subParameter classPath="jmt.engine.random.ExponentialPar" name="
    distrPar">
<subParameter classPath="java.lang.Double" name="lambda">
<value>PROC_TIME</value>
</subParameter>
```

```
</subParameter>
</subParameter>
</parameter>
</section>
<section className="ServiceTunnel"/>
<section className="Router">
<parameter array="true" classPath="jmt.engine.NetStrategies.
    RoutingStrategy" name="RoutingStrategy">
<refClass>CLASS</refClass>
<subParameter classPath="jmt.engine.NetStrategies.RoutingStrategies.
    EmpiricalStrategy" name="Probabilities">
<subParameter array="true" classPath="jmt.engine.random.
    EmpiricalEntry" name="EmpiricalEntryArray">
<subParameter classPath="jmt.engine.random.EmpiricalEntry" name="
    EmpiricalEntry">
<subParameter classPath="java.lang.String" name="stationName">
<value>NEXT_STATION</value>
</subParameter>
<subParameter classPath="java.lang.Double" name="probability">
<value>CONN_PROB</value>
</subParameter>
</subParameter>
</subParameter>
</subParameter>
</parameter>
</section>
</node>
```

Listing 4.12: source string

Where:

- SOURCE_NAME is the name of the source;
- CLASS is the name of the class/part type;
- PROC_TIME is the arrival time of the part type in the system;
- NEXT_STATION represents the machine that is next to the source; in this case the first machine visited from the part type
- CONN_PROB represents the probability for the class to visit the machines defined in NEXT_STATION; which in this case is 1.

In order to define all the source station, a list with the product type name, source name and the first machine visited has been created. Using also a list with the information necessary about the product such as product name, arrival time and its distribution, it is easy to define all the source. It is just required for each source to substitute the value in caps lock with the class parameters. It is important

to identify the time distribution of the arrival time and so choose the right XML coding for that.

4.3.3 Queue station

This station is the most critical of the entire definition since there can be multiple product processed in a single station and each of them has different requirements. As already introduced, the station is composed by three parts: queue, service and routing section.
For each of these parts, every class must be defined separately from the others, creating a series of parameters and sub parameters.
For any doubt in the source definition process, please refers to Figure A.1 where it is possible to see all the necessary steps together with the XML string needed the step.

The definition of the station starts with the name declaration and after that it start the modelling of the queue section with the buffer size and the drop strategies. This is defined in Listing 4.13.

```
<node name="MAC_NAME">
<section className="Queue">
<parameter classPath="java.lang.Integer" name="size">
<value>BUFFER_SIZE</value>
</parameter>
<parameter array="true" classPath="java.lang.String" name="
    dropStrategies">
[insert DROP RULE CLASS SPECIFIC STRING for all the classes defined
    in the station]
</parameter>
```

Listing 4.13: head buffer string

Since the drop strategy is class specific it must defined for each class, using Listing 4.14

```
    <refClass>CLASS</refClass>
    <subParameter classPath="java.lang.String" name="dropStrategy">
    <value>BAS blocking</value>
    </subParameter>
```

Listing 4.14: drop rule class specific string

The closing parameter at the end must be inserted after the definition of all class specific strings to close the drop rule definition.
After the drop strategy, it is needed to define the queue policy. The definition is composed from two parts: a section specific and a class specific. We implemented a FCFS strategy according to the hypothesis. It is possible to find the two strings as

```
<parameter classPath="jmt.engine.NetStrategies.QueueGetStrategies.
    FCFSstrategy" name="FCFSstrategy"/>
<parameter array="true" classPath="jmt.engine.NetStrategies.
    QueuePutStrategy" name="QueuePutStrategy">
     [insert QUEUE STRATEGY CLASS SPECIFIC STRING for all the classes
    defined in the station]
</parameter>
</section>
```

Listing 4.15: Queue strategy string

```
<refClass>CLASS</refClass>
<subParameter classPath="jmt.engine.NetStrategies.QueuePutStrategies.
    TailStrategy" name="TailStrategy"/>
```

Listing 4.16: Queue strategy class specific string

At this point, it is needed to define the service section, thus, specify the characteristic of the part processing.
The first parameter to be defined is "MaxJob" which is the number of servers for the station, which in our model is set to 1. This parameter is considered machine specific (in Listing 4.17).
The next one is the number of visits, which according to the part type definition is also equal to 1 since the products are not visiting the station more than one time. This parameter is composed of sub parameters which are class specific, so there is the need to repeat the strings for each class in the machine (Listing 4.18 for the station string and Listing 4.19 for the class specific).
Last parameter of the section is the processing time, also this part is defined of one station specific string (Listing 4.20) and class specific sub-string (Listing 4.21). For this specification there is the need to verify the type of time distribution so that it can be used the correct modelling. This operation must be repeated for each part type that is processed in the machine.
The construction of this section is realized by having the following data:

- List of all the machines name and for each of them all the product that have a process within the machine together with processing time and time distribution.
- Buffer dimension for each machine

```
<section className="Server">
<parameter classPath="java.lang.Integer" name="maxJobs">
<value>1</value>
</parameter>
```

Listing 4.17: service section string

```
<parameter array="true" classPath="java.lang.Integer" name="
    numberOfVisits">
     [insert SERVICE VISIT CLASS SPECIFIC STRING for all the classes
    defined in the station]
</parameter>
```

Listing 4.18: Service number of visit string

```
<refClass>CLASS</refClass>
<subParameter classPath="java.lang.Integer" name="numberOfVisits">
<value>1</value>
</subParameter>
```

Listing 4.19: service visit class specific string

```
<parameter array="true" classPath="jmt.engine.NetStrategies.
    ServiceStrategy" name="ServiceStrategy">
     [insert SERVICE STRATEGY CLASS SPECIFIC STRING for all the
    classes defined in the station]
</parameter>
</section>
```

Listing 4.20: service strategy station string

```
<refClass>CLASS</refClass>
<subParameter classPath="jmt.engine.NetStrategies.ServiceStrategies.
    ServiceTimeStrategy" name="ServiceTimeStrategy">
<subParameter classPath="jmt.engine.random.Exponential" name="
    Exponential"/>
<subParameter classPath="jmt.engine.random.ExponentialPar" name="
    distrPar">
<subParameter classPath="java.lang.Double" name="lambda">
<value>PROC_TIME</value>
</subParameter>
</subParameter>
</subParameter>
```

Listing 4.21: service strategy class specific string

Regarding the "service strategy class specific string" it is shown the exponential distribution coding with PROC_TIME equal to the lambda.

The last section of this element is the "Router". It defines the destination of products leaving the station. The routing strategy is set as probabilities for all the class, even if the system is a flow shop in order to keep the modelling as much generic as possible.
The generation of this section is possible by having all the connections between machines for each class. As defined in section 4.2.4.7, for each class it is possible to define its path in the different stations. By using this list it is possible to assign the correct routing to each class.
First it is defined the station specific string (Listing 4.22, then, with a simple cycle of the class specific strings (Listing 4.23) for all classes in the system and a substitution of the key work in CAPS LOCK it is possible to generate the router section. It's important to notice that in case of multiple downstream station for one class, there is the need to repeat the routing string for all the following stations (Listing 4.24) . The level of cycling can be understood by looking at the text right alignment.

```
<section className="Router">
<parameter array="true" classPath="jmt.engine.NetStrategies.
    RoutingStrategy" name="RoutingStrategy">
    [insert ROUTING CLASS SPECIFIC STRING for all the classes defined
    in the station]
</parameter>
</section>
</node>
```

Listing 4.22: routing section string

```
<refClass>CLASS</refClass>
<subParameter classPath="jmt.engine.NetStrategies.RoutingStrategies.
    EmpiricalStrategy" name="Probabilities">
<subParameter array="true" classPath="jmt.engine.random.
    EmpiricalEntry" name="EmpiricalEntryArray">
    [insert ROUTING DOWNSTREAM STATION SPECIFIC STRING for all the
    classes defined in the station]
</subParameter>
</subParameter>
```

Listing 4.23: Routing class specific string

```
<subParameter classPath="jmt.engine.random.EmpiricalEntry" name="
    EmpiricalEntry">
<subParameter classPath="java.lang.String" name="stationName">
<value>NEXT_STATION</value>
</subParameter>
<subParameter classPath="java.lang.Double" name="probability">
<value> CONN_PROB</value>
```

```
</subParameter>
</subParameter>
```

Listing 4.24: routing downstream station specific string

The queue definition process is summarized with reference to XML strings of code in Figure A.1.

4.3.4 Sink

in Appendix, Chapter A The last element to be analysed it the sink. For our model it has been decided that there is a specific sink for each class of products.

```
<node name="SINK_NAME">
<section className="JobSink"/>
</node>
```

Listing 4.25: sink string

To define the the system sinks, it is created a list that for all the class contains the following information:

- type name;
- sink name;
- last machine visited from the product;

This list is used also in define the connection between the last machine and the sink.
It is needed to replace the word SINK_NAME with the name of the sink. The last machine information is used in the next step.

4.3.5 Connections

Once all the system elements have been defined there is still the need to connect them according to the physical connections defined in the ontology.
The only information required is the name of one station and the name of the connected one. With the use of the list "physical_Mergedconn" (section 4.2.4.8 the connections are defined easily. For open class systems, in addition to the physical stations, also source and sink have to be considered. While for closed system,it is needed to consider the additional connections given by the link between the last machine of a process plan and the first one.
The XML structure for the connection is structured as reported in Listing 4.26:

```
<connection source="SOURCE" target="TARGET"/>
```

Listing 4.26: connection string

This structure really fits with the definition of couple of connections decided. With a simple cycle for all the number of connection and by replacing the SOURCE and TARGET variables, the generation of this part of the XML input can be easily done.

4.3.6 Performance to be evaluated

It is possible to set the performance to be analysed also through the XML input. The definition is easier and reflects the structure present in the user interface. The XML is reported in Listing 4.27

```
<measure alpha="ALPHA" name="NAME" nodeType="STATION" precision="
    PRECISION" referenceNode="NODE" referenceUserClass="CLASS" type="
    TYPE" verbose="false"/>
```

Listing 4.27: performance evaluation string

The variables are defined as follow:

- ALPHA = 1-Confidence Interval;
- NAME : name of the performance, it is not the performance to be analysed but a way for JMT JSIM to classify the performance. In the model it has been defined as “Station name + class to be evaluated + type of performance”, for example “M1_Class1_Throughput”;
- STATION: specify if the performance is station specific or global, in the first case insert “station” otherwise leave it empty;
- PRECISION: the value for the maximum relative error;
- NODE: station where the performance has to be evaluated, if it is a system performance leave it empty;
- CLASS: if the performance is class specific insert here the class name;
- TYPE: kind performance to be analysed, for example throughput or residence time;

For the XML generation, it is needed to cycle between all the machines and all the classes processed in them to obtain all the performance which are station/class specific.

4.3.7 Load of population for closed system

In case of closed class product, it is needed to place the existing population inside the production system. In particular, in the station where is assigned the first process step. This assignment of the population is possible through the listing reported in Listing 4.28

```
<preload>
    <stationPopulations stationName="ORIGIN_STATION">
        <classPopulation population="POP_NUMB" refClass="CLASS"/>
    </stationPopulations>
</preload>
```

Listing 4.28: preload string

It is needed to define the loading for each station that is the reference station for a closed class, and for each station it is important to declare the class population for all the part types. This solution requires a cycle within a cycle.

4.3.8 Ending

After the definition of the production system in its entirety the XML generation can be concluded with the following string:

```
</sim>
</archive>
```

Listing 4.29: end string

With these 2 lines of code the XML input can be considered completely modelled. Before concluding it is important that the reader understand that this XML input codification is the results of analysis on the output and continuous testing of input generation. Some information can be omitted during the input but once the file has been saved, the new .jsimg file presents more parameters than the one generated. This is because through the user interface inside each station is possible to see the parameter specific for all classes, also those have no process step assigned to that machine. For this reason, JMT JSIM accept the XML file where the parameters not needed are not defined , but once the file is opened from the user interface all the parameters are saved in the file.

4.4 Run Simulation

Once concluded the XML modelling phase the representation of the production system on XML is obtained and it can be opened and analysed through the use of JMT JSIM.

At this point there is the real phase of performance evaluation. In previous Section subsection 4.3.6, it has been defined the possibility to include some predefined performance for the existing stations and the system in general. Moreover the user has the possibility to set the preferred parameter choosing from a wide set.
In this section, we described how the simulation can be run in a more or less automatic approach.
The classic method is the use of the user interface of JMT while the other involve the use of the pc console. The two approach are described and analysed in order to evidence point of strength and weakness. It is taken in hypothesis that all the performance have already been defined in the previous section.

4.4.1 JMT JSIM user interface approach

This approach is the common one since it involve the use of the user interface which is specifically design also for this aim.
Trough the "simulation parameter" setting, depicted in **??** is possible to access to a set of parameter which can look limiting compared to other simulation software. As it can be seen in the picture the user can have access to:

- simulation random seed;
- maximum simulation duration;
- maximum simulated time;
- maximum number of samples;
- animation update interval;

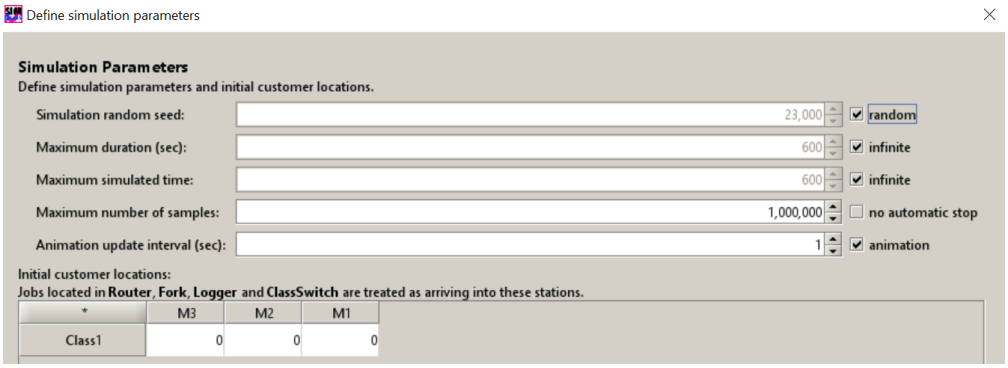

Figure 4.6: JMT JSIM simulation parameter

It can be possible to notice that it is not possible to set any parameter for the analysis of the transient. This is due to the fact that JSIM performs automatically

the transient detection, based on spectral analysis, computes and plots on-line the estimated values within the confidence intervals. This feature release the user who is not practical with these setting from the risk of making error due to the lack of experience.
Once these parameters have been set it can be possible to launch the simulation. Once clicked the button a new interface is opened where it is possible to see the simulation ongoing. For each of the performance previously defined it can be possible to see live the simulation progress. If the simulation constraints (such as maximum processing time) allows the simulation to get a results within the confidence interval and the maximum relative error, the user sees a green check, otherwise in case of yellow or red there is the need to act on simulation parameters or performance precision.
At this point, it is important to remember that the simulation results are not saved on a separate file but on the same XML input. For this reason after a simulation if the data obtained are meant to be kept the file need to be saved. At this point if the original XML file is opened is possible to see the results coded after the closing of the previously defined XML input, so after the "</jmodel>" line.

Going back to the user interface, there is another useful feature for the simulation process, the what-if analysis. It gives the possibility to the user to launch multiple simulation changing in each of them a parameter. It is possible to act on: arrival rates and service rates of a singular station, for just one or all the classes, and on the seed. This last parameter is in particular interesting. In fact, by conducting a what-if analysis changing the seed, it can be possible to set the number of simulation that are to be run with different seed. Since seed has no direct effect on the results this change allows the user to run multiple simulation without the need of launch each of them singularly and save the data.
Also in this case the results are saved in the XML file. In Section (4.5) it is analysed the output format and the procedure to extract these data for an analysis is discussed.

4.4.2 Console simulation run

This second approach is the one that allows for a higher automation. In fact, in the point of view of a completely automatic process which lead from the ontology to their performance evaluation, this path is the one to be followed.
Even if this approach is the best one in terms of future automatic application it has some limits. First of all trough this approach it is only possible to have access to a way more limited set of simulation parameters: the seed and the maximum simulation time. In fact as it can be possible to see through the documentation on the console these are only two settings possible.
In Listing 4.30 the command that need to be launched from the console for the

run of the simulation:

```
java -cp JMT-singlejar-1.0.3.jar jmt.commandline.Jmt sim XML_FILE
  PATH -seed SEED -maxtime MAX_TIME
```

Listing 4.30: console command for the launch of the simulation

Where:

- XML_FILE_PATH represent the location of the file on the PC
- SEED represent the simulation random seed
- MAX_TIME represent the maximum simulation time

It is important that the command is launched from the location of the java executable file, which for example in the computer used for the simulation is: C:/Program Files (x86)/Java/jre1.8.0_211/bin/
Once the simulation is completed it is generated a file which is named as the XML file plus "-result". In this file there are stored only the result of the performance analysis.
Another limit of this approach is represented by the need of multiple simulations. While for the user interface approach there is the what-if analysis with the console there is the need to launch multiple time the console command. The issue in this is due to the fact that it can be possible that if the console is taking a little more time to complete the analysis and the command is launched again, there can be error or at least a smaller number of simulation than needed.
Moreover, if an XML file it is opened trough the JMT JSIM user interface, it can be possible that the tool fills automatically some missing parameter, such as wrong routing policies, or routing missing probabilities, while launching the simulation trough the console requires a perfect input.
After all these consideration, it has been decided that the best way of action is the use of the user interface for the launch of the simulation.

4.5 Output Reading and Results Analysis

In this section it is analysed the process of accessing the results of a user interface launched simulation and how it can be possible to save them. As introduced in previous section section 4.4, the performance are saved on the same XML input file after the closing of the model.
First of all it is analysed how the codification works for a normal simulation and then the analysis is expanded to the what-if analysis.
They data are encoded as Listing 4.31

```
<results elapsedTime="3161" logDecimalSeparator="." logDelimiter=","
    pollingInterval="1.0" xsi:noNamespaceSchemaLocation="Results.xsd">
<measure alpha="0.99" analyzedSamples="20480" discardedSamples="1520"
     finalValue="0.8830009274216856" name="M1_Throughput" nodeType="
    station" precision="0.03" referenceClass="" referenceStation="M1"
    state="1" type="5">
<sample lastIntervalAvgValue="0.8837670231940549" lowerBound="
    0.8564216390206626" meanValue="0.8848770789862674" simulationTime=
    "14226.600076748122" upperBound="0.9152884250373146"/>
<sample lastIntervalAvgValue="0.8828154645686694" lowerBound="
    0.8564216390206626" meanValue="0.8848770789862674" simulationTime=
    "24349.89351539046" upperBound="0.9152884250373146"/>
<sample lastIntervalAvgValue="0.86590315547372" lowerBound="
    0.8601599321025973" meanValue="0.8830009274216856" simulationTime=
    "24915.77665936853" upperBound="0.9070880693266152"/>
</measure>
</results>\newline
```

Listing 4.31: XML result codification

It is possible to see that the encoding involve three different main lines: results, measure and sample.
The first one represents, the opening of the results part by clarifying some parameters; it is closed after all the performance. Measure introduce the single performance analysis by defining some parameter which identify the performance such as the station and classes taken in analysis, the evaluation precision and all the other value set in the XML definition in subsection 4.1.3. Moreover it is defined the final value, which represent the performance value. Moving to sample it is possible to find some intermediate results, this can be seen by looking at "simulated time".
Once this data location is well defined is possible to extrapolate the information needed for a better display and for a future results analysis. The extraction is conducted trough the development of a python script which access the XML file and copy only the valuable information on a Excel file so that it can be possible for the user to have the data ready for use. Moreover it can be also possible to develop some analysis directly in the script trough the use of dedicated library. This last implementation has not been conducted yet.
Once this simpler simulation results have been analysed it can be possible to move to the what-if analysis. with this new kind of simulation the data displayed on the XML file have some differences. In Listing 4.32 it is depicted the new coding.

```
<results elapsedTime="0" logDecimalSeparator="." logDelimiter="," pollingInterval="0.0" xsi:noNamespaceSchemaLocation="Results.xsd">
<measure alpha="0.99" analyzedSamples="10" name="M1_All classes_Throughput" nodeType="station" precision="0.03" referenceClass="All classes" referenceStation="M1" type="5">
<sample lowerBound="0.8684821692048661" meanValue="0.8813253280233613" upperBound="0.8945540389881116" validity="true"/>
<sample lowerBound="0.8667800313806752" meanValue="0.8812530230981305" upperBound="0.8962175454014092" validity="true"/>
<sample lowerBound="0.8725772672511427" meanValue="0.8853356082998474" upperBound="0.898472576199752" validity="true"/>
<sample lowerBound="0.8717453573791785" meanValue="0.8853871588866736" upperBound="0.899462704659873" validity="true"/>
<sample lowerBound="0.8742652996836974" meanValue="0.8866605630077312"
...
...
</measure>\newline
```

Listing 4.32: XML result codification what-if

For this alternative the main change is given to the fact that the sample line represent each simulation launched with the different seed. So in order to define the performance evaluation result it is necessary to make a statistical analysis. In the developed method it simply involve the calculation of the average and the standard deviation which are needed for the phase of validation of the model.
In **??** it can be found the final version of the code

Chapter 5

Validation of the model

This chapter has the aim to apply the modelling defined in Chapter 4 to some defined cases studies in order to verify that the performance evaluated form the model are representing the one of the real system.
In a first phase, we check that the physical system modelled from the ontology exactly represents the real one and respects the hypothesis discussed in Chapter 3. In the second phase, we check that the XML is designed as it is defined according the model.
Once the physical system is confirmed to be as it was designed to be represented, it can be possible to proceed to the validation process. This validation is executed by designing a second simulation model following the hypothesis defined in Chapter 3. The second simulation system is modelled in TECNOMAX Plant Simulation, a commercial software, using a manual creation instead of an automatic one as for the model to be verified.
The performance evaluated from the two systems are collected and compared making the hypothesis that the Plant Simulation system represent the original ontology system.
The validation process is conducted through a hypothesis test. It aims at verifying if the two population under study belong to the same population, thus if the simulation models represent the same production system. The KPIs analysed are listed in the following:

- system throughput;
- product average lifespan;
- throughput for each machine;
- machine utilization for all stations;

The hypothesis test is defined for two population which are independent and with unknown variance [23]. It is assumed $\sigma_1 \neq \sigma_2$.
The test is conducted with p significance level of 0.05 on the following hypothesis:

- $H_0 : \mu_1 = \mu_2$
- $H_1 : \mu_1 \neq \mu_2$

Where μ_1 represents the actual average value for the real system, which is represented from Plan Simulation system, while μ_2 the performance value in the defined model.
The hypothesis zero defines that the two system performance come from the same population, thus, the model represents the original system, while hypothesis one define that the two performances obtained are from different populations so the two systems are not related.
Both the simulation are conducted with 50 experiments.
The simulation model is validated on the following types of production systems:

- flow shop: single and multi class
- hybrid flow shop: single and multi class
- job shop

Moreover, in addition to the three production systems, it is defined the modelling for the Cosberg assembly line industrial case study is addressed.
For all the case study, the model is able to define both open and closed system, but in this validation only the open definition is analysed.
Regarding the simulation parameter, for Plant Simulation it is considered to start the performance analysis after 72h which is hypothesised to be enough time to exclude effect of the transient period. The simulation is run for 144 hours. In JMT JSIM, it is not possible to set these parameter so it has been set a confidence interval of 0.95.
For each hypothesis test, the results are collected in a table which represent the following information for each performance evaluated:

- $\overline{X}$: mean value of the performance
- S: standard deviation of the performance
- t: t-student obtained from the test
- P: p-value obtained
- Result: if is "Significant" there are strong evidence to reject the H_0; if it is "Not Significant" there are weak evidence to reject the H_0

The JMT JSIM queue system obtained for all the case studies are defined in **??**.

5.1 Flow shop

This case study is designed as the most basic one.It is composed of three machines, each of them with its own buffer upstream. All the products start their process plan in M1 and end it in M3. Moreover, the buffers are designed with a buffer size of 10 units.
The production system is designed as depicted in Figure 5.1, with a production plan represented in Table 5.1.

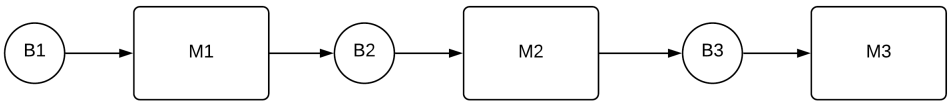

Figure 5.1: Flow shop physical system

PART TYPES	PRODUCTION PLAN	PROCESS STEPS	PROCESSING TIME	TIME DISTRIBUTION	MACHINE ASSIGNED
Class1	ProcessPlan01	P1_01	1.3 s	Exponential	M1
		P1_02	1.4 s	Deterministic	M2
		P1_03	1.2s	Exponential	M3
Class2	ProcessPlan02	P2_01	1.3 s	Exponential	M1
		P2_02	1.1 s	Deterministic	M2
		P2_03	1.2 s	Exponential	M3

Table 5.1: Process plan flow shop case study

5.1.1 Flow Shop single class

For the open system case, deterministic arrival rate of 1.8 second/piece is set. The production system is created in the ontology and then imported on the Stardog database.
After this phase the Python script is launched. At this point, it is needed to define the arrival rate of the parts.
The XML is generated through the steps described in Chapter 4 and the simulation is launched.
For the validation phase, the physical system is manually generated as shown in Figure 5.2.

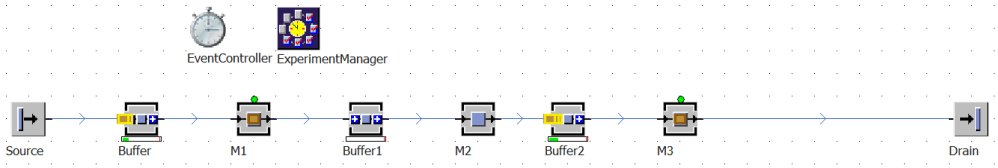

Figure 5.2: Plant Simulation representation flow shop single class

Through the experiment manager, all the KPI defined in the first part of the chapter have been defined and a simulation of 72h is launched.
The results are reportred in Table 5.2.
From the results given from the hypothesis test it not possible to reject the H_0,

	JMT		Plant Simulation		t	P	Result
	X	S	X	S			
Avg. Lifecycle	10.10562871	0.134591826	10.1471	0.0688	1.9400	0.0553	Not Significant
TH tot	0.555537892	0.000179303	0.55555409	1.62357E-05	0.6362	0.5262	Not Significant
TH M1	0.555528053	0.000124524	0.555557639	6.29418E-06	1.6779	0.0966	Not Significant
TH M2	0.555638886	0.000172923	0.555559028	1.39702E-05	3.2549	0.0016	Significant
TH M3	0.555537892	0.000179303	0.555556481	1.59463E-05	0.7302	0.4670	Not Significant
% M1	0.722019306	0.007629821	0.722410044	0.00247821	0.3444	0.7313	Not Significant
% M2	0.779729853	0.008801027	0.7776741	0.002002292	1.6105	0.1105	Not Significant
% M3	0.66686127	0.006035616	0.666597409	0.001693463	0.2976	0.7666	Not Significant

Table 5.2: Hypothesis test results for flow shop single class

thus $\mu_1 = \mu_2$ for all the performances except the throughput of station M2. All the KPIs can be considered representative of the original system.
Once this test is executed, it can be useful to make another test with the production system under high utilization condition. For the open system case, exponential arrival rate of 1.5 second/piece is set. Moreover, the buffers are reduced to a buffer size of 5 units.
The simulation is launched and the data are collected giving the results reported in Table 5.3

	PS		JMT		t	P	Result
	X	S	X	S			
Avg. Lifecycle	10.1471	0.0688	10.10562871	0.134591826	1.9400	0.0553	Not Significant
TH tot	0.55555409	1.62357E-05	0.555537892	0.000179303	0.6362	0.5262	Not Significant
TH M1	0.555557639	6.29418E-06	0.555528053	0.000124524	1.6779	0.0966	Not Significant
TH M2	0.555559028	1.39702E-05	0.555638886	0.000172923	3.2549	0.0016	Significant
TH M3	0.555556481	1.59463E-05	0.555537892	0.000179303	0.7302	0.4670	Not Significant
% M1	0.722410044	0.00247821	0.722019306	0.007629821	0.3444	0.7313	Not Significant
% M2	0.7776741	2.00E-03	0.779729853	0.008801027	0.2313	0.8181	Not Significant
% M3	0.666597409	1.69E-03	0.66686127	0.006035616	0.2976	0.7666	Not Significant

Table 5.3: Hypothesis test results for flow shop single class in blocking situation

The results of the new hypothesis test proves that H_0 must be rejected, defining that the two test population are not the same for all the KPIs. For this reason it is possible to claim that the system analysed is is not representative of the original. This validation evidences the inability of the model to represent system with high utilization of machine combined with a variability of arrival time and process step due to stochastic time distribution. This issue is probably a consequence of the blocking dynamics and a different definition of drop strategies that have not been analysed.
After this first set of tests, new flow shop systems are designed in order to understand at which level of saturation and variability the modelling can be representing. The first important change is done by changing the arrival time from exponential distribution to deterministic, in this way the first origin of uncertainty can be excluded. This new hypothesis can also be feasible by considering that the flow shop has a feeding system which is always ready to launch the product in the line according to a deterministic arrival rate.
Trough multiple tests it is possible to deduce that this modelling creates validity issues when the machines have a utilization rate over around 80%. This analysis confirm the previous hypothesis that defines the lack of validation in case of multiple blocking situations.

5.1.2 Flow Shop multi class

The introduction of an additional class in the system, with respect to the single class case, is very important in order to understand if the coding of XML input file has been done correctly since there are lot of parameters and policies that required specific definition for each class.
The physical system is defined as the single class case (Section 5.1.1).
The buffer size set to 10 units and the deterministic arrival rate for classes is set to 3.2 seconds/piece. All the steps defined in the single class system are followed to get the simulation results
For the validation phase, the physical system is manually generated as shown in Figure 5.3

For the KPI evaluation, the hypothesis test is defined and it is possible to accept the hypothesis H_0 for all the performance excluding the residence time for both the classes where H_0 must be rejected. With these results is possible to claim that excluding the analysis of the time spent in the system, the model defined is representing correctly the original model in case of flow shop multi class. The test results are reported in Table 5.4.

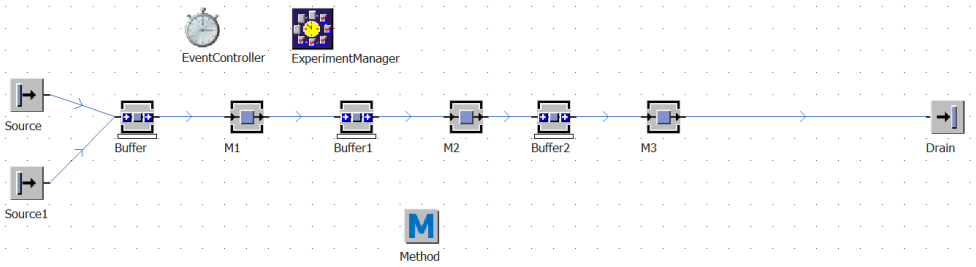

Figure 5.3: Plant Simulation representation flow shop multi class

	JMT		Plant Simulation		t	P	Result
	X	S	X	S			
Avg. LifecycleCl1	3.32892328	0.200414556	3.153	0.0632	5.9196	0.0000	Significant
Avg. LifecycleCl2	3.219436661	0.111485052	3.3019	0.0582	4.6366	0.0000	Significant
TH Cl1	0.71427388	9.27506E-05	0.714286111	5.45382E-06	0.9309	0.3542	Not Significant
TH Cl2	0.714259007	0.000104372	0.714286188	5.85824E-06	1.8386	0.0690	Not Significant
TH M1	1.428495245	0.00033491	1.428569907	6.63393E-06	1.5761	0.1182	Not Significant
TH M2	0.71384861	0.003085645	0.714284799	3.64674E-06	0.9996	0.3200	Not Significant
TH M3	0.714713034	0.003711223	0.714284877	3.63589E-06	0.8158	0.4166	Not Significant
TH M4	1.42852105	0.000317255	1.428572299	1.08957E-05	1.1416	0.2564	Not Significant
% M1	0.714850847	0.003795642	0.714148202	0.001182145	1.2498	0.2144	Not Significant
% M2	0.535291585	0.00274613	0.535053227	0.007030371	0.2233	0.8238	Not Significant
% M3	0.535622751	0.002714064	0.536373955	0.007029775	0.7049	0.4825	Not Significant
% M4	0.713251123	0.00383822	0.714056798	0.001101641	1.4267	0.1569	Not Significant

Table 5.4: Hypothesis test results for flow shop multi class

5.2 Hybrid flow shop

Next case study to analyse if the hybrid flow shop. This system is a variant of the traditional flow shop due to the introduction of parallel machining. In fact there is the first introduction of multiple path due to the presence of two parallel machines. The physical system is depicted in Figure 5.4 with a production plan reported in Table 5.5. Moreover, the buffers are designed with a buffer size of 10 units.

5.2.1 Hybrid flow shop single class

This first version of the case study is developed to test the probability routing policy developed in the Section 4.3. In fact as anticipated in the introduction of the chapter the existence of more machines in parallel requires the definition of the probability routing policy which needs to be tested with first this case study and later with the job shop one (Section 5.3).
The product enter in the system with an arrival rate of 0.8 second/piece with a deterministic time distribution.

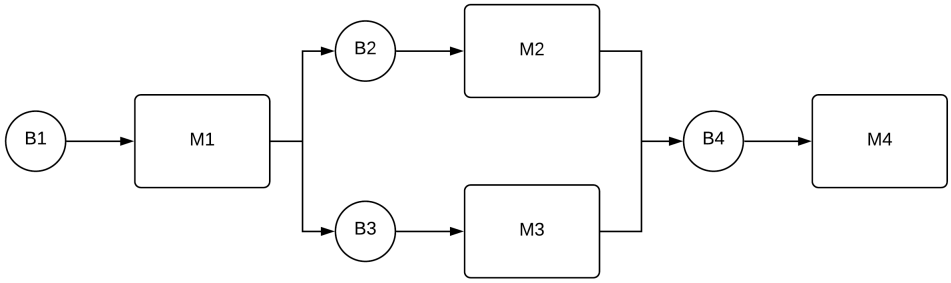

Figure 5.4: Hybrid flow shop physical system

PART TYPES	PRODUCTION PLAN	PROCESS STEPS	PROCESSING TIME	TIME DISTRIBUTION	MACHINE ASSIGNED
Class1	ProcessPlan01	P1_01	0.5 s	Exponential	M1
		P1_02	0.7 s	Deterministic	M2, M3
		P1_03	0.5 s	Exponential	M4
Class2	ProcessPlan02	P2_01	0.5 s	Exponential	M1
		P2_02	0.8 s	Deterministic	M2, M3
		P2_03	0.5 s	Exponential	M4

Table 5.5: Process plan hybrid flow shop study

Once also this control is done, it can be possible to move to the performance evaluation process through the what-if analysis simulation. When this step is completed, the results extraction script is used to collect the statistical data needed for the validation.
The Plant Simulation is depicted in Figure 5.5. The simulation is launched providing

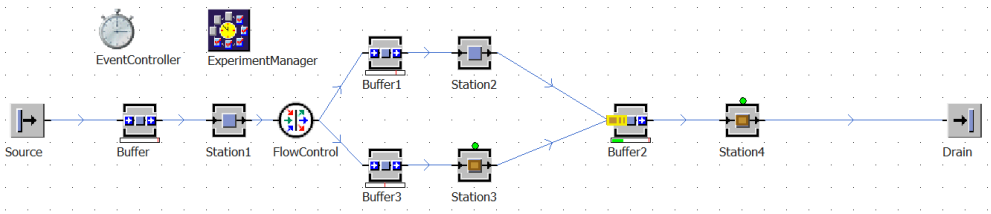

Figure 5.5: Hybrid flow shop Plant Simulation system

the results reported in Table 5.6
Given the results of the test, it is possible to accept the hypothesis H_0 for all the performance excluding the residence time for both the classes where H_0 must be rejected. With these results, it is possible to claim that,as the flow shop multi class case (Section 5.2.1), the time spent in the system of the model is note representative

	JMT		Plant Simulation		t	P	Result
	X	S	X	S			
Avg. Lifecycle	2.913016265	0.03319615	2.5811	0.0065	69.3835	0.0000	Significant
TH tot	1.250064675	0.000251148	1.250000926	8.30608E-06	1.7939	0.0759	Not Significant
TH M1	1.249987056	0.000149556	1.250001929	4.62766E-06	0.7029	0.4838	Not Significant
TH M2	0.625557713	0.003977853	0.625001929	4.21546E-06	0.9880	0.3256	Not Significant
TH M3	0.624981158	0.004328469	0.625002932	3.17184E-06	0.0356	0.9717	Not Significant
TH M4	1.250064675	0.000251148	1.250003472	8.14692E-06	1.7223	0.0882	Not Significant
% M1	0.624041111	0.006873223	0.625140348	0.001079496	1.1172	0.2666	Not Significant
% M2	0.437093633	0.005538254	0.437942924	0.001220323	1.0589	0.2922	Not Significant
% M3	0.436511067	0.006290487	0.437528077	2.02E-05	1.1432	0.2557	Not Significant
% M4	0.62670151	0.007130288	0.624878484	0.001082145	1.7874	0.0770	Not Significant

Table 5.6: Hypothesis test results for hybrid flow shop single class

of the original system. For the other KPIs the model is verified.

5.2.2 Hybrid flow Shop multi class

For the hybrid flow Shop multi class case, the arrival rate is set to 1.4 second/pieces with a deterministic time distribution.
The Plan Simulation system is represented in Figure 5.6 and the performances of the two different model are evaluated according to the methods developed.
From the KPIs obtained, the hypothesis test is completed obtaining the results

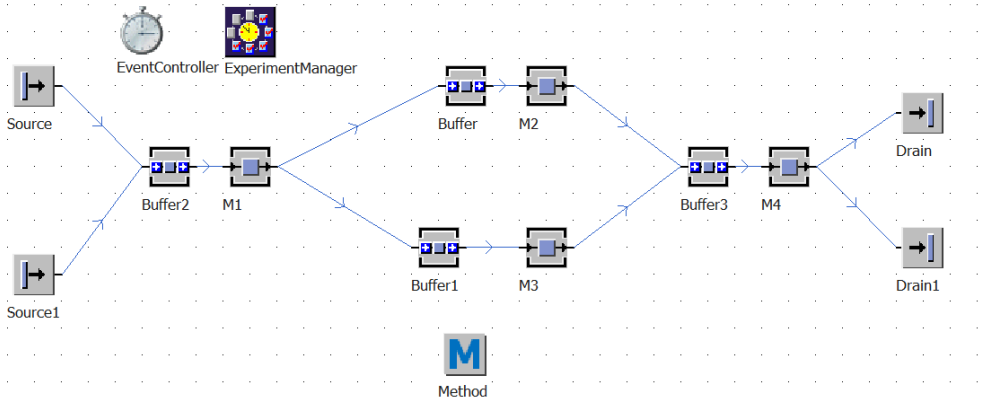

Figure 5.6: Plant Simulation hybrid flow shop multi class system representation

reported in Table 5.7

From the KPI evaluation, the hypothesis test is defined and it can be possible to accept the hypothesis H_0 for all the performance excluding the residence time for both the classes where H_0 must be rejected. With these results is possible to claim

	JMT		Plant Simulation		t	P	TestResult
	X	S	X	S			
Avg. LifecycleCl1	3.32892328	0.200414556	3.153	0.0632	5.9196	0.0000	Significant
Avg. LifecycleCl2	3.219436661	0.111485052	3.3019	0.0582	4.6366	0.0000	Significant
TH Cl1	0.71427388	9.27506E-05	0.714286111	5.45382E-06	0.9309	0.3542	Not Significant
TH Cl2	0.714259007	0.000104372	0.714286188	5.85824E-06	1.8386	0.0690	Not Significant
TH M1	1.428495245	0.00033491	1.428569907	6.63393E-06	1.5761	0.1182	Not Significant
TH M2	0.71384861	0.003085645	0.714284799	3.64674E-06	0.9996	0.3200	Not Significant
TH M3	0.714713034	0.003711223	0.714284877	3.63589E-06	0.8158	0.4166	Not Significant
TH M4	1.42852105	0.000317255	1.428572299	1.08957E-05	1.1416	0.2564	Not Significant
% M1	0.714850847	0.003795642	0.714148202	0.001182145	1.2498	0.2144	Not Significant
% M2	0.535291585	0.00274613	0.535053227	0.007030371	0.2233	0.8238	Not Significant
% M3	0.535622751	0.002714064	0.536373955	0.007029775	0.7049	0.4825	Not Significant
% M4	0.713251123	0.00383822	0.714056798	0.001101641	1.4267	0.1569	Not Significant

Table 5.7: Hypothesis test results for hybrid flow shop multi class

that excluding the analysis of the time spent in the system, the model defined is representing correctly the original model in case of hybrid flow shop multi class.

5.3 Job shop

Last case study defined for the validation of the model is a job shop. This is the most complex system due to the fact that each part type has its own routing. The physical system is depicted in Figure 5.7. Each buffer of the system has been designed with a capacity of 10 units, and all the arrival rates of the three classes are set to a value of 2.5 second/piece, following a deterministic time distribution. The process plan is defined in Table 5.8.

The simulations on both tools are launched and the performance evaluated are

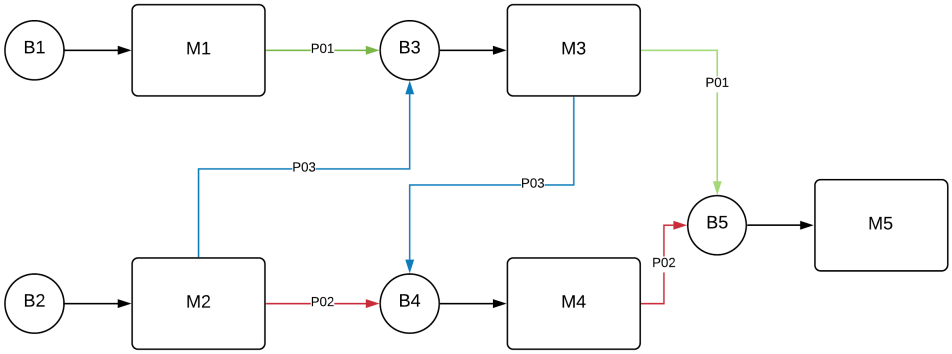

Figure 5.7: Jobshop system representation

PART TYPES	PRODUCTION PLAN	PROCESS STEPS	PROCESSING TIME	TIME DISTRIBUTION	MACHINE ASSIGNED
ProductJS01	ProcessPlanJS01	PJS01_01	1.5 s	Exponential	M1
		PJS01_02	0.7 s	Exponential	M3
		PJS01_03	0.7 s	Exponential	M5
ProductJS02	ProcessPlanJS02	PJS02_01	0.6 s	Exponential	M2
		PJS02_02	0.8 s	Exponential	M4
		PJS02_03	0.7 s	Exponential	M5
ProductJS03	ProcessPlanJS03	PJS03_01	0.6 s	Exponential	M2
		PJS03_02	0.7 s	Exponential	M3
		PJS03_03	0.8 s	Exponential	M4

Table 5.8: process plan job shop case study

collected and analysed. The result are reported in Table 5.9

The results of the hypothesis test on the case study shows that, excluding the residence time, it is possible to accept H_0, so to claim that the KPIs from the model represent of the original system performance.

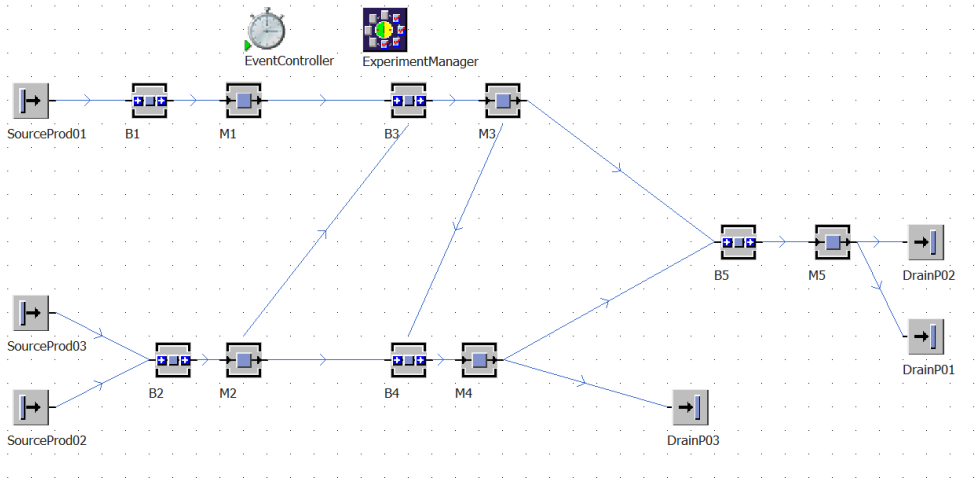

Figure 5.8: Plant Simulation Jobshop system representation

	JMT		Plant Simulation		t	P	Result
	X	S	X	S			
Avg. LifecycleCl1	4.731905105	0.035843824	4.7588	0.0168	4.8042	0	Significant
Avg. LifecycleCl2	4.04364859	0.019447675	4.5327	0.2909	11.8612	0	Significant
Avg. LifecycleCl3	3.429976029	0.019216564	4.3714	0.2772	23.9572	0	Significant
TH Cl1	0.400006546	3.76E-05	0.400000772	6.25E-06	1.0702	0.2872	Not Significant
TH Cl2	0.39999624	4.45E-05	0.400000386	6.67E-06	0.651	0.5166	Not Significant
TH Cl3	0.400002903	2.72E-05	0.4	5.14E-06	0.7411	0.4604	Not Significant
TH M1	0.399999951	3.90E-05	0.399998843	4.47E-06	0.1997	0.8422	Not Significant
TH M2	0.800011519	4.89E-05	0.800003858	6.56E-06	1.0976	0.2751	Not Significant
TH M3	0.800029413	8.48E-05	0.800000772	9.42E-06	2.3736	0.0196	Significant
TH M4	0.800013836	0.000111162	0.800000772	9.06E-06	0.8283	0.4095	Not Significant
TH M5	0.800002412	0.000147352	0.800001157	8.14E-06	0.0601	0.9522	Not Significant
% M1	0.599201655	0.004271045	0.600319716	0.001674158	1.7234	0.088	Not Significant
% M2	0.479837102	0.002889523	0.64035372	0.000999362	371.2309	0	Significant
% M3	0.560127685	0.004151241	0.560166901	0.001556616	0.0625	0.9503	Not Significant
% M4	0.639262597	0.004161988	0.639987164	0.001074138	1.192	0.2362	Not Significant
% M5	0.559532056	0.003236309	0.559575437	0.001777352	0.0831	0.934	Not Significant

Table 5.9: hypothesis test results on job shop production system

5.4 Real Industrial case

The case study addresses on the analysis of the assembly process of hinge for furniture doors. The hinge is depicted in Figure 5.9. This case differs from the previous ones due to the assembly process involving the presence of different component in the system. In particular, this process of the hinge assembly considers the assembly of some sub-assemblies and some components in different stations of an assembly line.
The assembly line is composed of a linear conveyor capable of handling pallets, and

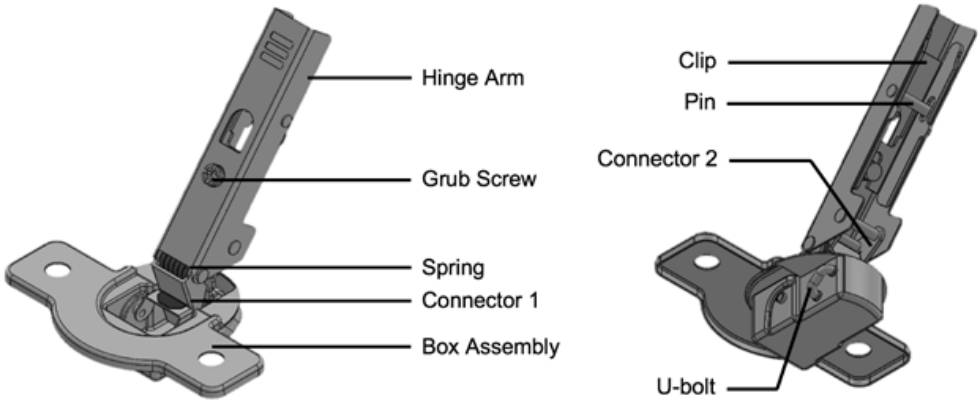

Figure 5.9: representation of the assemble hinge

a set of assembly stations. In each one a component is added to the hinge, while it is blocked into the pallet. Each assembly station is equipped by with a feeding system conveying the needed component. Each feeding group is designed with a vibratory bowl feeder, a linear rail and a pacing selector to compose the component. This solution grants the feeding application for a wide range of components of different sizes and materials. An example of an assembly line is shown in Figure 5.10.

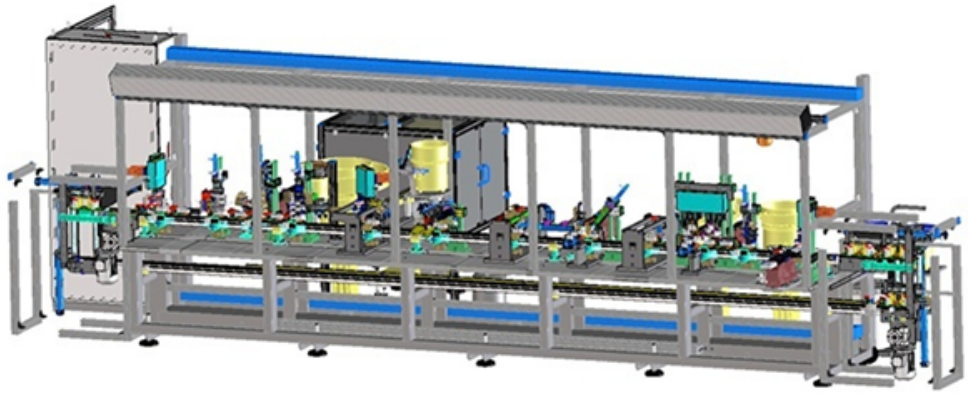

Figure 5.10: Model of an assembly line similar to the case study one

In this specific case, the pallets, moved by the conveyor, pass 9 stations corresponding to 8 (+1) assembly operations. These operations are described below:

1. Arm feeding: an arm is conveyed to the assembly line and tightened into the pallet.
2. Grub screw screwing: a grub screw is conveyed to the station, aligned to the corresponding hole on the arm and screwed. Processing time: 0.9 s

3. Clip alignment: a clip is conveyed to the line and inserted in the arm, hooking it to the grub screw. Its holes are aligned with those of the arm.
4. Clip riveting: insert rivet into arm and clip and riveting holes.
5. Rod/spring alignment: the rod/spring sub-assembly is conveyed to the line. Its holes are aligned with those of the arm.
6. Assembly riveting: inserting rivet into the holes of the arm and assembly and riveting.
7. Box alignment: a box is conveyed on the line and aligned with the arm.
8. U-bolt insertion: a U-bolt is inserted in the holes of arm and box.
9. Control: a vision system controls the presence of all components in the assembly. If the assembly is not complete or has defects, a slide is opened in which the assembly to be discarded is conveyed.

All the operations are defined with an exponential time distribution, the processing time is defined in In Table 5.10

OPERATION	PROCESSING TIME
Arm feeding	0.7 s
Grub screw screwing	0.9 s
Clip alignment	0.7 s
Clip riveting	0.5 s
Rod/spring alignment	1 s
Assembly riveting	0.5 s
Box alignment	0.7 s
U-bolt insertion	0.6 s
Control	0.8 s

Table 5.10: processing time of assembly system

At the end of the line the pallet is brought to a lift where it lowered to a conveyor that flow under the assembly conveyor. This second conveyor redirects all the pallet in a buffer at the start of the assembly line where they can start again the cycle. From there it is delivered to the beginning of the line where it is ready to start a new assembly process. Since the sub-assembly is moving on the whole line on a pallet, it can me modelled as the pallet itself. In this way instead defining a pallet that is moving from station to station with the sub-assembly, it is considered a pallet that at each station get a component through an operation and at the end of the last process (Control) a fully assembled hinge is obtained.

A system representation is depicted in Figure 5.11
The use of a pallet defined in a fixed number in the system would required the

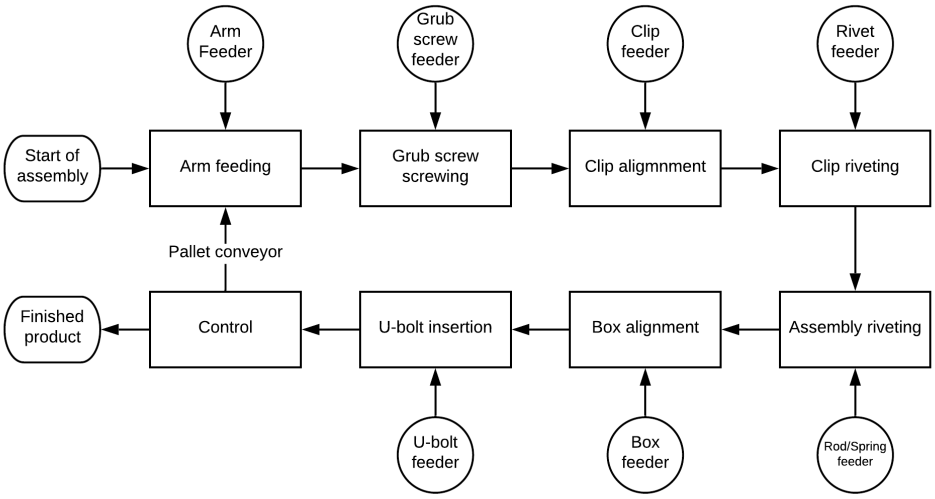

Figure 5.11: cosberg assembly line representation

designed of a closed system, but due to their high number, their presence is not having influence on the performance of the system. Therefore the assembly process can be designed as an open one.
The second issue of this particular system is the assembly process. In OntoGui the implementation of assembly process is possible, but only in "OWL Individual Manager". This means that the definition of the system is more complicate and different from the one defined through "System Design" where it is not possible to set relationship between part types. Moreover, an assembly system does not satisfy the hypothesis defined in Chapter 3. For these reason it cannot be represented through the model defined, so in order to have a JMT JSIM representation it has been defined manually.

5.4.1 JMT JSIM representation

In JMT JSIM, , the process of modelling two or more products merging into a single one is a delicate step due to the structural limit of the tool itself.
The software presents two elements that can be used for representing such a process: the fork and the join. The first one let the user define the creation from one class of multiple class, the second one merge again the class created. For technical details please refers at the JMT user manual [8]. These two elements allow the definition of component but have the limit that in order to merge two or more components

with the join, these components must have been created from the job in the fork element. This means that in order to represent the flow of component, they must be generated in the fork from an original class; it is not possible to represent their arrival in the system in another way. This limitation defines constraints in terms of path definition: if in the real system the components enter in the system from the feeders, in this representation they must be generated at the start of the line and and convey them in the feeding stations. Moreover, due to the join constrains that requires that only part type created from the original one in the fork can merge together again, it is not possible to change the part type definition during the processing, for example by changing in the assembly station from semi-finished-part01 to semi-finished-part02. For this reason, it is not possible to define the change of the product to be assembled during the assembly system. In Figure 5.12, the component structure generation and merging through the fork and join element is depicted.

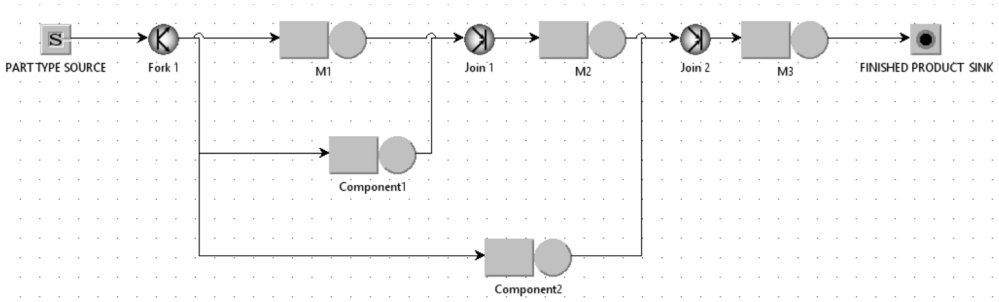

Figure 5.12: Fork and Join component structure

With these constraints an assembly process can be defined as follow. The finished product is generated from a source and is directed to a fork where all the components using "Multi-Branch Class Switch" strategy. The original part is directed to the first assembly machine and each component is conveyed to its own dedicated feeding system. Each feeding system is represented as a queue station with really high or infinite buffer size and with a processing time lower than the line throughput in order to have the component always available for the arrival of the part to be assembled. Upstream each assemble station, a join element has to be defined. In to that join are directed the component for the assembly operation and the semi-finished product. With a "Quorum" join policy, the component from the feeding system waits for the semi-finished part in the join element and once it arrives they are merged into the original part type. By repeating this process for all the assembly process, it is possible to define the whole system. With these hypothesis is possible to represent the assembly system as depicted in Figure 5.13

In order to evaluate if the system designed can represent the assembly process

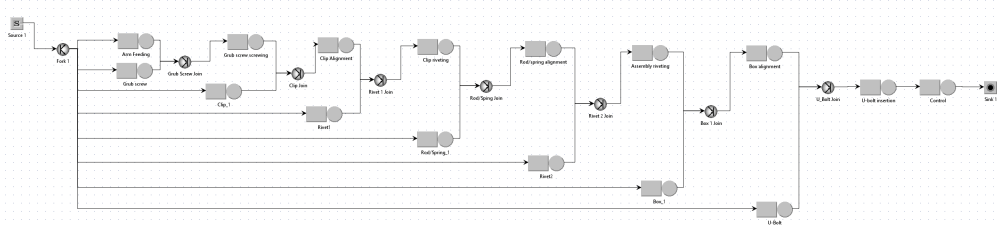

Figure 5.13: JMT JSIM representation of Industrial case study

correctly, we also defined a simplified case where the whole line can be approximate to a flow shop. The hypothesis made is that the feeding system of the component to the main line can be excluded from the analysis. This is due to the fact that the buffers of the various feeders have such capacity that there is no starvation risk. Moreover, the availability of the feeding sub-system is such high that its effect on the performance of the line is that low to justify that the effect are minimal. Each station of the flow shop represents an assembly operation even if the component are no more present.
The flow shop approximation is depicted in Figure 5.14
Once both the system are designed, the performance can be evaluated setting

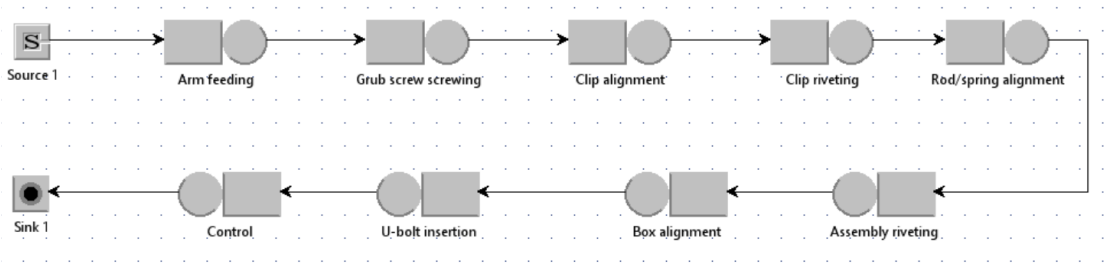

Figure 5.14: flow shop approximation for the assembly line

an exponential arrival rate of 1.5 second/piece. It is decided to compare the system throughput and system residence time. Due to the high complexity of the component system the simulation could not be concluded multiple time due to pc memory limit. For this reason the residence time performance can not be defined as the previous case study and as the the counterpart in the simplified system. The previous analysis cannot be used as statistical proof for the validation of the model due to the lack of statistic samples. For this reason it is is decided to use the approximate flow shop system for the performance evaluation and its following hypothesis test. The assembly line is defined with buffers with a capacity of 4 units and the product entering the system with an arrival rate of 1.5 second/piece with a deterministic time distribution.

The Plant Simulation model is depicted in Figure 5.15
The simulation is launched for both model and the performance evaluated are

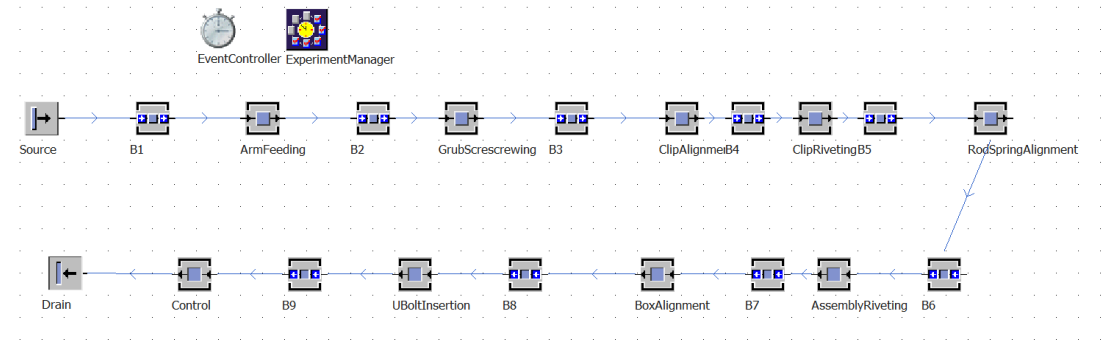

Figure 5.15: Plant Simulation approximation for the assembly line

collected for the hypothesis test. The results of the test are represented in Table 5.11. In the table, the name of the machines have been approximated to with the position of the process step in the process plan, for example" Grub screw screwing" process is M2 since it is the second process.

	JMT		Plant Simulation		t	P	Result
	X	S	X	S			
Avg. LifecycleCl1	10.8979683	0.059751504	10.9137	0.0344	1.613426231	0.10986755	Not Significant
TH Cl1	0.66666945	5.68081E-05	0.666667284	8.58659E-06	0.26655819	0.790369413	Not Significant
TH M1	0.666655736	2.25934E-05	0.666667438	2.33833E-06	3.643097996	0.00043292	Significant
TH M2	0.666656601	3.49013E-05	0.666670293	5.69631E-06	2.737788449	0.007346897	Significant
TH M3	0.666655327	3.06473E-05	0.666669599	6.54728E-06	3.220274927	0.001737897	Significant
TH M4	0.666662424	4.07467E-05	0.666668904	7.95219E-06	1.103796651	0.272384311	Not Significant
TH M5	0.666647397	5.88546E-05	0.666671373	8.29617E-06	2.852403155	0.005292875	Significant
TH M6	0.666653818	4.67096E-05	0.66667037	8.95371E-06	2.46090704	0.015607571	Significant
TH M7	0.666663197	4.92391E-05	0.666670139	1.11667E-05	0.972163724	0.333362527	Not Significant
TH M8	0.666659987	5.47193E-05	0.666669367	1.01701E-05	1.191778267	0.236226314	Not Significant
TH M9	0.66666945	5.68081E-05	0.66666929	8.79903E-06	0.019637693	0.984372322	Not Significant
% M1	0.466610944	0.00243742	0.466674667	0.001257711	0.164280538	0.869848657	Not Significant
% M2	0.600074581	0.003991912	0.600175062	0.001293321	0.16932081	0.865893322	Not Significant
% M3	0.4662784	0.002198178	0.466541836	0.001317062	0.726922592	0.469006264	Not Significant
% M4	0.332980303	0.001838139	0.33327762	0.000906156	1.025856341	0.307484805	Not Significant
% M5	0.666105573	0.003609182	0.666781846	0.001636544	1.206688356	0.230456956	Not Significant
% M6	0.333484388	0.001785565	0.333291618	0.00083986	0.690795223	0.491327225	Not Significant
% M7	0.4668604	0.002614847	0.46662704	0.001372498	0.558757774	0.577601587	Not Significant
% M8	0.39953295	0.001819914	0.399962049	0.001104303	1.425337925	0.157236806	Not Significant
% M9	0.533881962	0.00234743	0.533544877	0.001516017	0.852971164	0.39575522	Not Significant

Table 5.11: hypothesis test results on simplified assembly line

From the table it is possible to evidence that for the throughput for M1, M2, M3, M5 an M6 the results prove that there are enough evidence to reject H_0 so for these performance the model is not representative of the Plant Simulation system.

For the other performance it is possible to affirm that H_0 cannot be rejected so they are representative of the system.
Overall, the model proves to be valid to represent the system on the most general performance such as the average life cycle of a product in the system and the system throughput. For the unverified performance, it is possible to affirm that they are a consequence of the buffer size. In fact in the flow shop case study validation it has been proven that small buffer size creates more blocking situations even with a low utilization of machines which reduce the ability of the model to represent the original system

Chapter 6

Conclusion

In this thesis it has been develop a model to represent production system in order to evaluate their performances. This model is necessary for developing a way to create a way of exchange information from the physical system and the performance evaluation representation. The choice of the tools in support of the process has been determined by the need of interoperability in order to have a solution which could be, in the future, integrated in a deeper network of connections. For this reason the data of the system are based on an ontology structure and the performance evaluation tool is an open source queuing network based on a XML layer.
First, the model has been defined through the analysis of the two data structure available, on one side the ontology formalization and on the other one a queuing network. In order to have a reliable model for the representation of the original system, some hypothesis have been defined in order to limit its field of application and be able grant an accurate modelling. The main hypothesis is the possibility to apply this model only to flow shop, hybrid flow shop and job shop. Moreover, it is also defined that each machine in the system can be connected upstream only with one buffer, and that buffer has to be dedicated only to the use of the machine.
After this step it has been defined the approach for the resolution model which can be divided in three main phases: data extraction from the ontology, generation of the queuing network simulation model and reading and anlysis of the results. For the data extraction it has been made use of SPARQL query to interrogate the ontology on a remote server and later the information obtained have been refined in order to make them suitable for the following phase of the generation of the model. In this phase, the XML input structure of the queuing network has been analysed to learn how to define the needed element and generate a complete system. The performance model has been generated through the use of Python language to code an algorithm for the automatic conversion from the data obtained from the ontology to the XML input representation. Last phase involved the reading of the results in order to collect the data on the performance evaluated.

Last step of the thesis is the validation of the model. Even if the pure conversion from ontology to queuing network was defined correctly it must be verified that the model defined, on which the conversion process was built, could approximate the performance of the original system accurately. For this reason case studies have been defined, and their performance have been analysed through the defined model and a commercial software which was approximate to be representing of the original system. The results have been compared through a hypothesis test to verify if the model could be representative of the original production system.
From this last analysis it was defined that the model developed is able to correctly represent the system under normal machine load. The only KPI that is not representative of the system is the time spent in the system and as consequence the other pure time indicators. This issue can be traced back to the different data structure of the two system, in fact the performance evaluation tools is queuing network based, which has a different behaviour in certain situation. Regarding this last point it has in fact been observed that under high machine load (more than 80%),which leads to blocking situation in the system, the model defined is no more representative of the system, having all the KPIs failing the hypothesis test. The issue is probably generated by the different queuing management within the software structure. It is required in the future a more detailed analysis on the blocking situation in order to understand what created this and how it can be possible to solve it, by for example developing more hypothesis.
Regarding assembly systems, it has been identified a way to manually integrate it into the performance evaluation tools. In order to validate the model it has been designed a flow shop model which could approximate the assembly line and from its performance it has been conducted the validation. The model has proved to be reliable for the system performance such as average life cycle of a product in the system and system throughput but some machine throughput cannot be considered reliable. Also in this case it is a consequence of the blocking events due to the low buffers size. Future work on this topic is required so that it can be possible to validate the representation defined, and, moreover, have a complete process to start from the ontology defined, and obtain a system representation. At this very moment research on the ontology side are being conducted so soon it will be possible to work with a more accessible way to define assembly systems. Moreover, a progress in the blocking situation analysis is going to give a real benefit also for the representation of system with buffer with low capacity as the industrial case defined.

Appendix A

XML generation

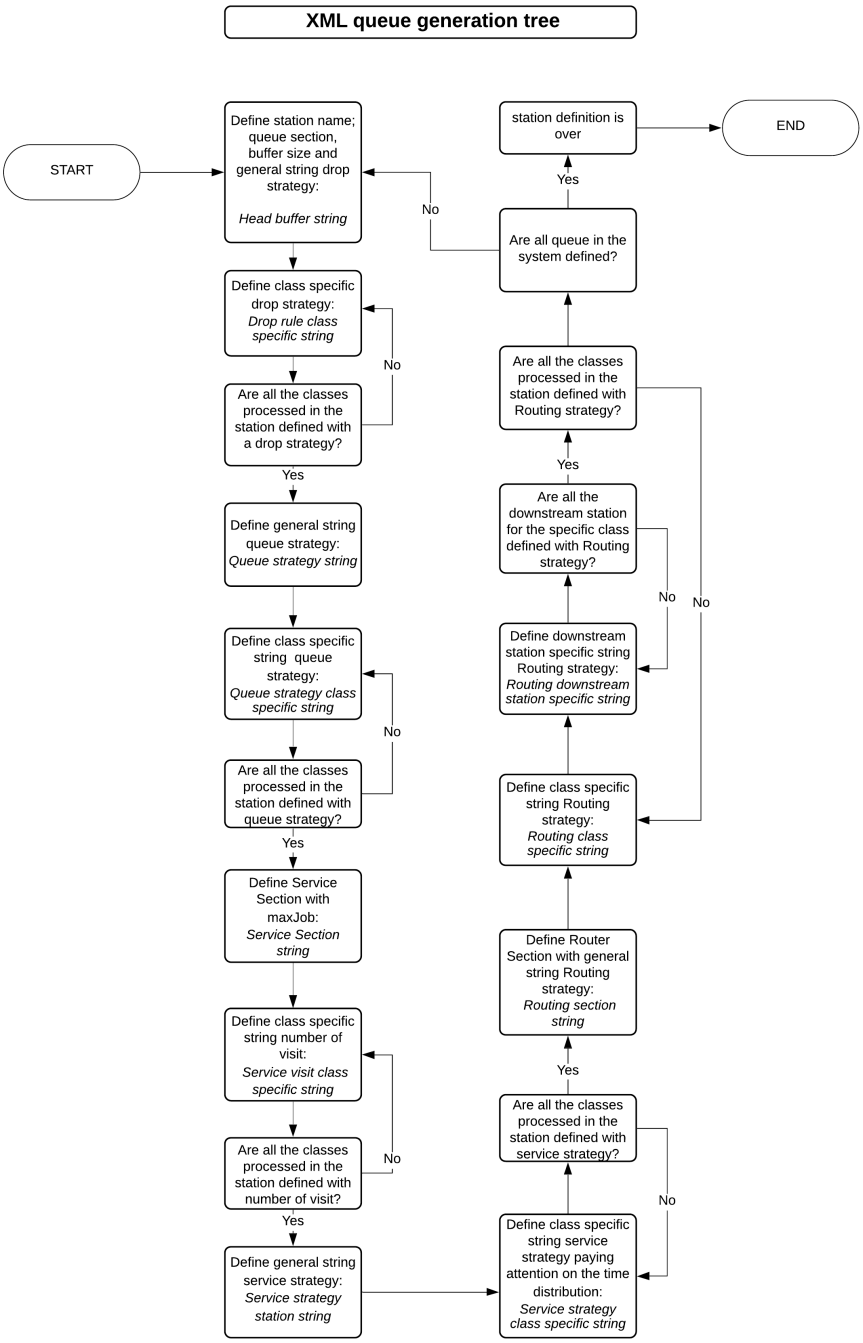

Figure A.1: XML queue specific generation process

Bibliography

[1] Walter Terkaj and Marcello Urgo. «Ontology-based Modeling of Production Systems for Design and Performance Evaluation». In: (Nov. 2014), pp. 748–753 (cit. on pp. 2, 5, 7).

[2] Walter Terkaj. «OntoGui: A Graphical User Interface for Rapid Instantiation of OWL Ontologies». In: (2017) (cit. on pp. 2, 6).

[3] *VLFT Project Website*. 2019. URL: https://www.vlft.eu/ (visited on 12/02/2019) (cit. on p. 3).

[4] International Society of Automation. «ISA-95: the international standard for the integration of enterprise and control systems». In: () (cit. on p. 5).

[5] National Institute of Standards and Technology. «Process Specification Language (PSL)». In: (2008) (cit. on p. 5).

[6] buildingSMART. *IFC Overview*. URL: http://buildingsmarttech.org/specifications/ifc-overview (cit. on p. 5).

[7] M. Shahbaz H.K. Lin J.A. Harding. «Manufacturing system engineering ontology for semantic interoperability across extended project teams». In: *Int. Journal of Production Research* 42.24 (), pp. 5099–5118 (cit. on p. 6).

[8] M.Bertoli G.Casale G.Serazzi. *JMT User manual*. 2018. URL: http://jmt.sourceforge.net/Papers/JMT_users_Manual.pdf (visited on 12/02/2019) (cit. on pp. 7, 20, 72).

[9] Swee K. Leong Charles R. McLean. «A Process Model for Production System Engineering». In: (1995) (cit. on p. 9).

[10] URL: https://www.w3.org/community/lbd/2014/12/12/ifcowl-ontology-file-added-for-ifc4_add1/ (cit. on p. 10).

[11] URL: https://www.stardog.com/ (cit. on p. 25).

[12] URL: https://www.w3.org/TR/rdf-sparql-query/ (cit. on p. 25).

[13] G. Pedrielli W. Terkaj. «ProRegio Deliverable VFF». In: (2017), p. 1 (cit. on p. 25).

[14] URL: http://www.ontoeng.com/statistics (cit. on p. 25).

[15] URL: http://www.learninglab.de/~dolog/fsm/fsm.owl (cit. on p. 25).

[16] URL: http://www.w3.org/ns/sosa/ (cit. on p. 25).

[17] URL: http://www.w3.org/ns/ssn/ (cit. on p. 25).

[18] URL: http://www.ontoeng.com/expression (cit. on p. 26).

[19] URL: http://www.ontoeng.com/osph (cit. on p. 26).

[20] URL: https://w3id.org/list (cit. on p. 26).

[21] URL: https://w3id.org/express (cit. on p. 26).

[22] URL: https://rdflib.github.io/sparqlwrapper/ (cit. on p. 36).

[23] Erich Leo Lehmann. *Testing statistical hypotheses*. 1986 (cit. on p. 60).