

# **ULBS\_05: Process-oriented topics: Modelling and simulation-based design and optimization of manufacturing systems and processes on the ADOxx platform**

## *Training goals*

The participants to this the training will

- Understand and apply methods for the design of manufacturing systems and processes;
- Understand and apply methods for the optimization of manufacturing systems operation;
- Acquire operational skills on the use of ADOxx toolkits for domain specific metamodeling;
- Acquire operational skills on the use of ADOxx toolkits for manufacturing systems modelling and simulation.

## Method

- Case studies;
- Metamodeling stage – the participants define together a Domain Specific Language optimally describing the domain of the studied cases. On its bases, they build the modelling and simulation tools;
- Modelling stage – grouped in team the participants compete in solving manufacturing systems design problem. They must design a system producing a given product assortment;
- Simulation stage – grouped in teams, the participants compete in solving manufacturing system optimization problems. They must find the best schedule for a given product assortment.

## Unit 1 Introduction to Manufacturing Systems and ADOxx

This unit will refresh the knowledge of the participants related to the following topics:

- manufacturing system
- modelling and simulation

### Manufacturing systems [Basnet1994]

A *manufacturing system* is an ensemble of humans and machines that interact in order to transform raw materials in finite products.

*Automation* groups the techniques by which the human contribution to the command and control of a process is reduced or eliminated. The goal of the process control is to maintain the values of the process outputs in the close vicinity of a set of given values (reference values).

*Automated manufacturing system (AMS)* are manufacturing systems that can handle the manufacturing of a finite products with little or no human contribution. This include the transport of material from a workstation to other, the control of each workstation, the synchronization of workstations, the inspection of materials and redirection of nonconforming products.

*Flexible manufacturing system (FMS)* are manufacturing systems able to reconfigure themselves very rapidly in order to produce multiple types of products. A FMS consist of:

- a set of workstations capable each of automatically executing a larger set of operations so that *machine flexibility* is provided. This assures that the system can absorb large-scale changes in volume, capacity, or capability demands.
- an material handling system based on flexible conveyors, automated guided vehicles (AGV) and loading-unloading robots that facilitate the transfer of material and tools from one workstation to other. This must provide *routing flexibility* which is the system's ability to be changed to produce new product types, and to change the order of operations executed on a part.
- a complex command and control system that orchestrates the cooperation of the before mentioned systems.

## **Modelling and simulation**

**Modelling** is the activity of model building. A *model* is an abstracted and simplified representation of a part of the reality. This mental construct allows the reasoning about the represented reality. In the context of this training a systemic view. Constructing a model consist in identifying the states, inputs and outputs that characterize it, and determining the relations that quantify the dependence of the states and outputs from the inputs. In order to handle complex systems, the hierarchical decomposition in a collection of interacting subsystems is also used.

Once the model is defined, **simulation** is the method used to determine the possible evolution of the modeled system starting from a given state. It consists in substituting the values for the starting states and inputs in the model relations and calculating the following states and corresponding outputs. The user, their context – to which modelling library they have access or are loaded by default for example, and the access right ate managed from access rights

## **Presentation of the ADOxx platform**

The ADOxx is a software engineering environment that allows the development of modelling tools. The tools allows the collaborative development and the sharing of the digital artefacts(model and metamodels) by using a multiuser database as the storage backend. As a result an acces control system is implemented and login with user and password is needed.

The two main applications of the environment that will be presented in this tutorial are:

- 1) the ADOxx Development Toolkit – this allows the effective development of the modelling tool in form of a library. This tool is also used to manage the user, their context (what modelling library they use by default for example) and their rights.
- 2) the ADOxx Modelling Toolkit – this allows the creation, editing and simulation(animation) of the models with the use of the library created in the

## **Discussing the workflow metamodeling/modelling/simulation**

The following use cases for the ADOxx ecosystem will be presented in this training:

- 1) Metamodeling – for the development of an Domain Specific Modelling Language (DSML) and of the supporting tools for editing abd simulating DSML models

This workflow will be presented in Unit 2. This is an activity that can be organized with computer and manufacturing engineering trainees in separate or mixed teams. The workflow comprises two steps

1.1) meta-model design it consists in identifying the basic components classes – the classes of basic building blocks from which any model in the domain will be built. For each component class there must be defined:

- the attributes; the values of this attributes describe the state of the component. The state of the model will be described by the ensemble of the values of the attributes of all components
- the possible relations between instances of component classes
- the behavior – how the state of the system is changing as a result of internal and external changes. Practically this means how the value of the attributes change. This allow the simulation of the model
- the graphical representation of both components and relations-this is linked to the values of the attributes
- the animation – this is the visual representation of the state change. To each possible transition between the model states a corresponding change in the graphical representation is associated

This can be performed with the help of any tool that allows schematic metamodel description from simple diagram drawing application or mind mapping tools to more sophisticated tools like for example CoChaCo4ADOxx metamodeling tool.

This is a mixed activity implying both manufacturing engineers that should bring domain specific knowledge and computer scientist which should bring this knowledge in a form usable in the next step.

1.2) meta-model implementation in ADOxx. This is performed with the help of the ADOxx Development Toolkit. Its usually a longer activity so it can be performed during one training session and will be scheduled as a unsupervised independent task for a team of trainees whit competencies in computer science taking part in the training.

## 2) Modelling and simulation on the ADOxx platform

This activity uses a library implementing a DSML developed using the workflow 1). This library is associated to the user environment by the system administrator using the Development Toolkit. At the login, the library is automatically loaded so that the user can start using it without other actions. The available DSML are visible in the Models pan where the user can select the type of the new model he wants to create. The user can perform two types of activities

- visual editing – it can compose the model from visual building block representing components and relations presented in a toolbar. The models can be built from scratch or by modifying pre-existing models. The models can be saved in the intern format for later editing or simulation or exported in XML format for import in other tools.

- simulation – the model is animated by performing step wise the state transition possible. This implies the updating of the attributes values and performing the corresponding changes in the model visual representation.

-

### 3) Modeling and simulation using the stand-alone program BeeUp

Bee-Up is an application developed with the ADOxx environment. It is a single user (no login) standalone simplified version of the ADOxx Modelling Toolkit. It contains the metamodels for 5 largely used modeling languages: Event Driven Process Chain (EPC), Business Process Modelling Notation (BPMN), Entity Relation (ER), Petri Net (PN) and Unified Modelling Language (UML) preloaded. So the user can visually edit models in this 5 language but cannot add supplementary modelling languages.

The Petri nets are modelling formalism of particular interest for the modelling of the manufacturing system because they are well suited to capture the discrete event and concurrent character of such systems. We will provide a short introduction to this formalism in the following.

#### Modelling and simulation with Petri net

In the second part the Petri net modeling theory, language and graphical notation is presented. The tools used for model development, for the simulation of the models and for the formal, qualitative and quantitative analysis of the systems are also presented.

Petri net theory was originally developed by Carl Adam Petri and presented in his Ph.D [Petri1973]. The common definition of PN's introduced by Petri is as follows. A Petri net or place/transition net can be defined as a five-tuple;  $PN = (P, T, I, O, Mo)$  (1) Where:

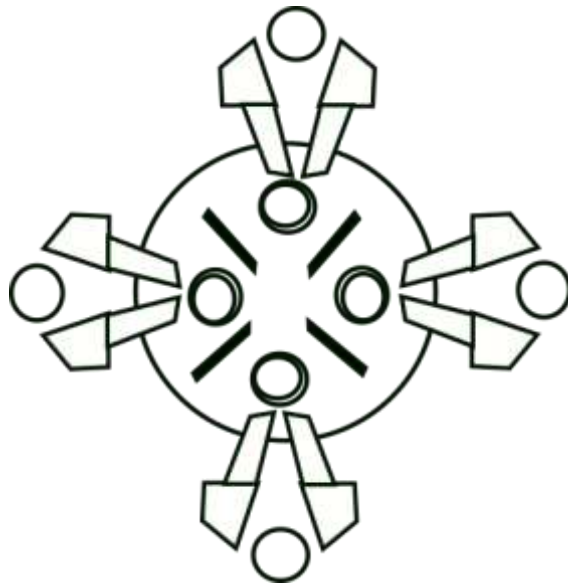
- $P$  and  $T$  are a finite non-empty sets of places pictured through circles and transitions pictured through rectangles, respectively.  $P$  and  $T$  are disjoint sets, and  $P \cup T$  are called nodes with  $P \cup T \neq \emptyset$  and  $P \cap T = \emptyset$
- $I$  and  $O$  are input and output functions that defines the set of directed arcs from  $T$  to  $P$ . More obvious, the input and output functions of Petri net can be represented by arcs with arrows between two different types of nodes.
- $M: P \rightarrow \mathbb{N}$ , is a marking whose  $i^{th}$  component represents the number of tokens in the  $i^{th}$  place  $P_i$ . The initial marking of the net is denoted by  $M_0$ . The Petri nets are assumed to be connected; it means that there is at least one path between any two nodes.

The state of the modelled system at any moment is described by the marking of net. Each transition changes the state of the systems by moving the tokens from some places to other places and so changing the marking.

The modeling using Petri consist in drawing the net by placing places and transitions and connecting them with arcs.

For the simulation, an initial marking is needed describing the initial state of the system. The simulation can be performed step by step with user contribution or automatically. In the step-by-step simulation the application will signal the transition that are ready to fire (in red with “fire” label on them). The user clicks on them and as a result the tokens from the preplace of the transitions are transported in the post places of the transitions. In case of competing transitions (transition sharing preplaces), the user choses which transition to fire. In case of the automatically simulation al transition activated at some steps are fired. In case of competing transition if rules are specified (priority or probability) these rules are applied. If no rule is specified one transition from the competing set is chosen randomly. The result of the simulation is plotted usually as number of tokens in each place vs steps.

We propose following problem to be modeled and simulated using the Bee-Up application



A classical problem in the study of concurrent system is the philosophe's dinner.

Figure 1.1 depicts an instance of this problem. Four philosophers take a common dinner at a round table. They are eating rice with sticks. The problem is they have only 4 sticks. To eat each philosophe needs 2 sticks so he can do two things: eat if both the left and right sticks are available or talk if not both sticks are available.

The problem can be extended to  $n$  philosophy and  $n$  sticks.

Figure 1.1 The philosophers taking dinner

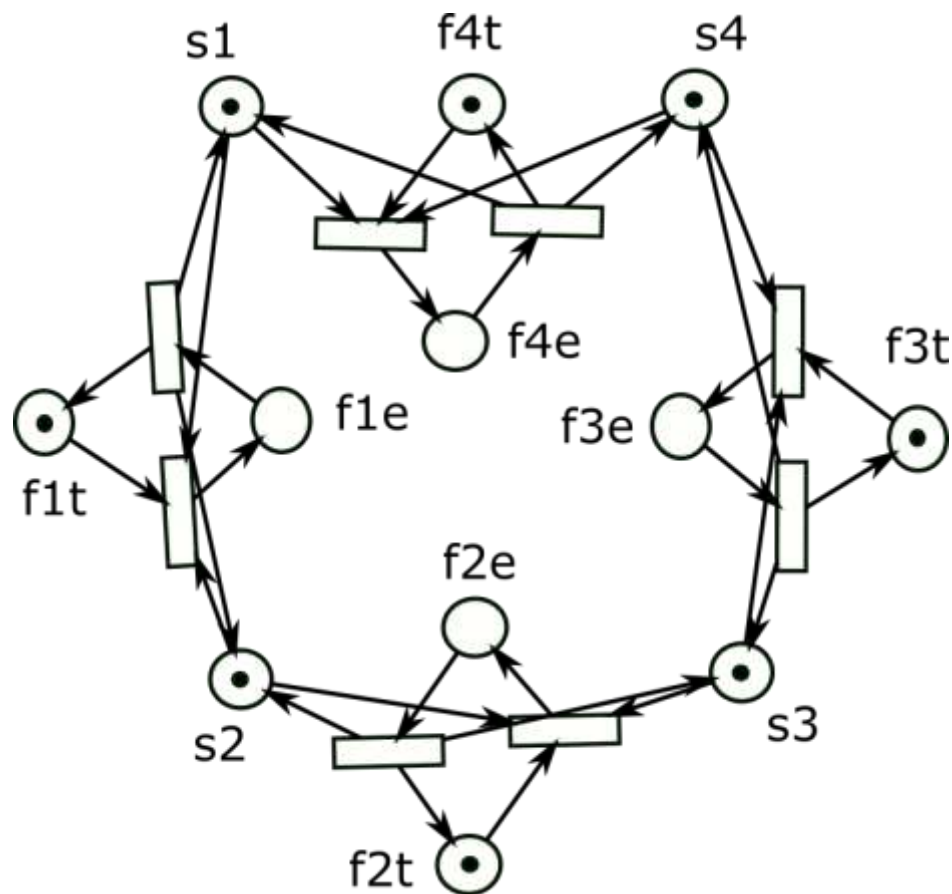


Figure 1.2 presents the corresponding Petri net that models the system.

The behavior of each philosopher ( $f1$ -  $f4$ ) is modeled by a net formed from two places – one representing the philosopher the eating ( $fie$ ) and other the philosopher talking ( $fit$ ) and two transition which allows the change from one state to other. One token representing the philosopher marks the state in which he currently is.

The sticks are represented as token that mark the stick places ( $s1$ - $s4$ ) when the sticks are not used. The sticks in use are not visible in the net. The arcs are connecting the transition with the pre and post places according to the precondition and the effects. The philosopher can transition from eating to talking and back. The sticks are be taken when the philosopher is transitioning to eating and put back when it transitions to talking. To eat the philosopher needs both sticks.

The trainees should simulate the net and identify many situations specific to concurrent systems including deadlocks.

Figure 1.2 The Petri net modelling the philosopher's dinner

## Unit 2 Metamodeling

Using the example provided below design and implement a Domain Specific Modelling Language for the modelling of manufacturing systems.

The manufacturing system presented in figure 1.1 is producing chocolate truffles packaged in aluminum foil bags that are inserted in cardboard boxes.

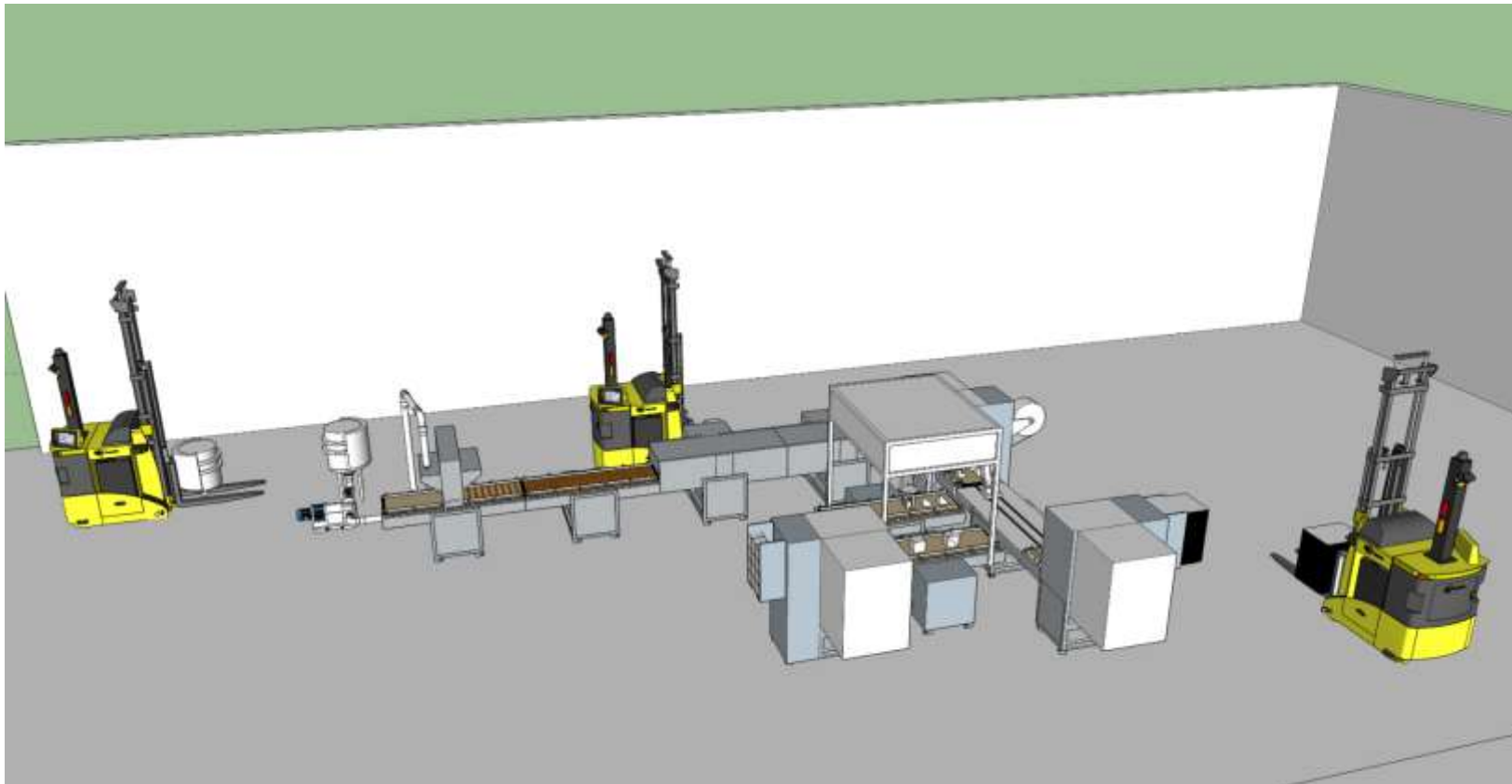


Figure 1.1 3D representation of a chocolate producing manufacturing line



The line has the following components:

- an AGV transports the buckets with chocolate ganache mass from the storage area to the loading buffer of the truffle forming machine;
- an AGV transports the cardboard packaging material from the storage area to the loading buffer of the box forming machine;
- On the first line, the chocolate ganache mass is fed from the loading buffer in the tank of the truffle forming machine; inside the machine the chocolate is melted and chocolate truffles are formed. The truffles are loaded on a transport belt which functions as a buffer for the transport line with freezing areas. The chocolate truffles are transferred on the transport line where they are cooled to solidify and preserve the shape and then loaded in the buffer of the aluminum foil packaging machine. Two processes occur here: the formation of the bags from the aluminum foil taken from a roll and the filling of the aluminum bags with chocolate truffles. The resulted aluminum bags filled with chocolate truffles are buffered in a dispenser;
- On the second line, cardboard packaging material from the loading buffer, is transformed by the box forming machine in cubic cardboard boxes. The boxes are loaded in the machine output buffer;
- On the assembly line, a robotic arm transfers alternatively the aluminum bags and the cardboard boxes on the feeding transport belts which function as a buffer for the final packaging machine. The packaging machine has a robotic arm with flex gripper that introduces the aluminum bags with truffles in the cardboard boxes. The final product - cardboard boxes having aluminum bags with truffles - is discharged in a storage buffer from which it is then later transported to the finite product storage area

## **The methodology for the definition and implementation of a domain specific language in ADOxx**

### **Defining the elements of the metamodeling language**

Therefore, the atomic concepts of the language will be Buffers, Workstations, Transport Machines and Ports, which will be represented graphically as the nodes of a graph, and the relations between them will be represented by the arcs of the graph.

Buffers are temporary warehouses in the manufacturing flow and are characterized by the type of material they can store, by the maximum amount of materials they can store and by the amount stored at a time. These features will be syntactically denoted by

attribute names. Buffers are components that store material without transforming it, so it must have all material ports of the same type. The maximum buffer capacity is fixed and cannot be extended (constant attribute). Their variable attribute is the current content, which can vary between 0 and the maximum capacity. The buffer cannot be loaded above the maximum capacity and cannot be unloaded if it is empty. Buffers are passive components, they are filled and emptied by other components with which they are connected.

Workstations are the components that perform the operations of assembling, subassemblies or transforming some material entities into other material entities. These concepts are components in the manufacturing flow characterized by the types and quantities of input material and by the types and quantities of output materials for each operation that can be performed in that workstation.

Workstations are components that transform materials. So, they must have at least one input and one output port of different types. They must allow the definition of operations that describe how many units of which materials are needed and which number of units based on which materials are produced through this operation.

Transport machines are components of the manufacturing flow that transport material entities between workstations, in principle from one buffer to another. These concepts are components in the manufacturing flow characterized by the types and quantities of materials that can be transported from one buffer to another.

Transport machines are components that transfer material without transforming it. They only change the position of material from the input buffer to the output buffer. So, they will have at least one material input port and output port for each material that is transported and the mass balance must be respected on the same material – number of entering units = number of exiting units.

The conveyers transport only one material so all ports are of the same type and the defining characteristics is the throughput – number of material units transported in the time unit.

The automated guided vehicle (AGV) is practically a mobile buffer. It has all attributes and behavior of a buffer, but it can connect and disconnect its ports from the corresponding ports of buffers and can move between preprogrammed positions. Also, it is an “active” component, initiating the loading unloading actions as soon as it is docked on buffer. So, all considerations concerning the interaction between buffers and components apply here.

The manipulator is a flexible transporter that can transfer multiple types of materials between different in and out ports of the same type. The most usual example is a manipulator that can handle different types of material moving them between different sub lines. There must be at least an in out pair of ports for each material type that is handled by the manipulator. At one moment the manipulator works only between one pair of ports of the same type.

Ports are the components in the manufacturing flow that connect the flow of output material entities from the components with corresponding inputs to other components. These concepts are components in the manufacturing flow characterized by the types of materials and the direction of entry or exit.

The control elements are components that transfer only information, so that they have only information ports - not necessarily all of the same information type. The control element is the ideal point for interfacing the model with other modules. The control element executes the control algorithm. The program reads feedback messages from the process from the input ports or commands from other command items. Depending on the current status and inputs, commands are generated that are placed at outputs.

By structuring this concept, a met model was constructed for the language. The Class representation of this metamodel is presented in figure 1.2.

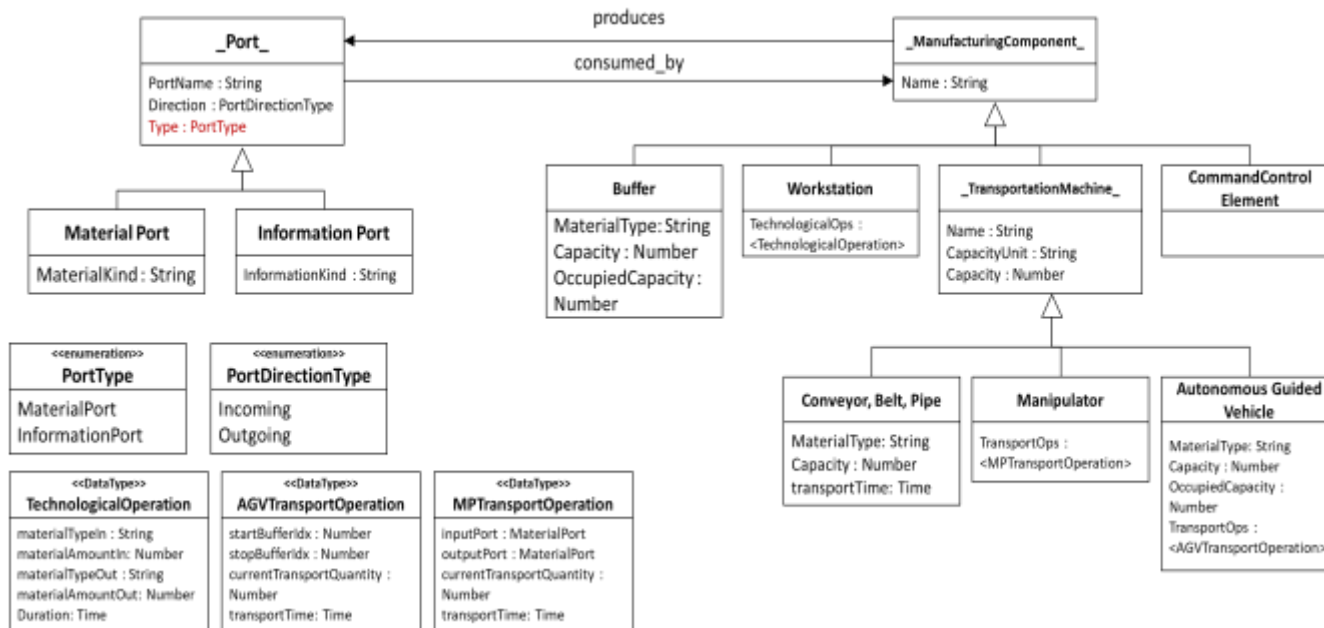


Fig. 1.2 The metamodel constructed from the categorial sketch

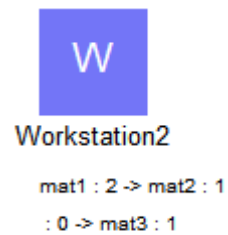
## Defining the graphical representation of elements



In figure 1.3 we can see the symbolic notation that we attributed to this type of component. Attribute names are: Name, MaterialType, Capacity, OccupiedCapacity.

In Fig. 1.4 we can see the symbolic notation that we attributed to this type of component. Attribute names are: Name, Duration, OperationCode, and a set of component records (MaterialType, MaterialAmountIn, MaterialAmountOut).

Figure 1.3. Buffer representation



In figure 1.5 we can see the symbolic notation that we attributed to the three types of conveyors, namely Fig. 1.5a contains the notation for the conveyor type, Fig. 1.5 b contains the notation for the AGV type, and Fig. 1.5c contains the notation for the manipulator type. The attribute names for the conveyor type are: Name, MaterialType, CapacityUnit, Capacity, TransportTime, OperationCode, and for the manipulator and AGV types there is a simple Name attribute and a set of structured records (MaterialType, TransportQuantity, TransportTime, OperationCode).

Figure 1.4. Workstation representation

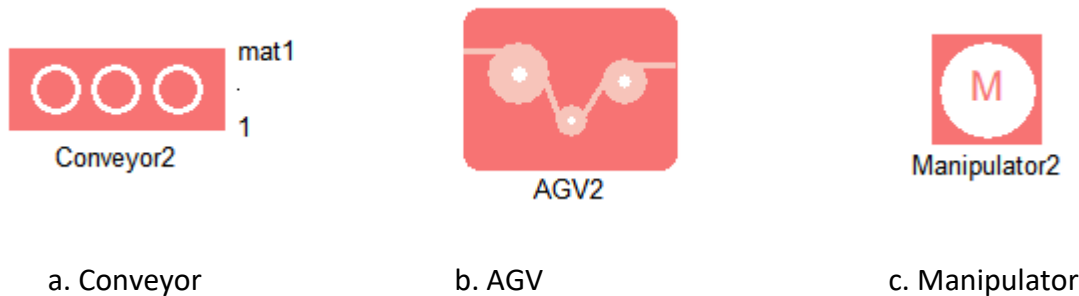
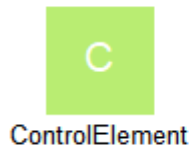


Figure 1.5. Transport machines notations



In figure 1.6 we can see the symbolic notation that we attributed to this type of component. Attribute names are: MaterialKind, PortName, PortDirectionType.

Figure 1.6. Material port notation



In figure 1.7 we can see the notations used for the control elements (figure 1.7 a) and the information ports (figure. 1.7b).

a. Control element

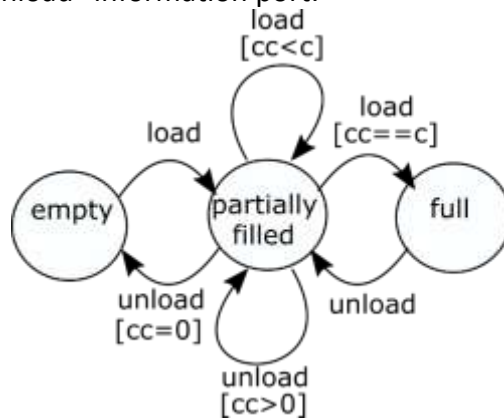
b. Information port

Figure 1.7. Control element and information port notation

## Defining the behavior of each element

### Buffers

The buffer (figure 1.8) has 3 states: empty ( $cc = 0$ ), partially loaded ( $0 < cc < c$ ), full ( $cc = c$ ). The buffer responds to 2 commands on the "load" and "unload" information port.



If the buffer is partially full, "Load" increments cc. "Unload" decrements cc. "Cc" is output on the out port

If the buffer is empty and "unload" is received, nothing is incremented, it is only issued on the out port - "empty".

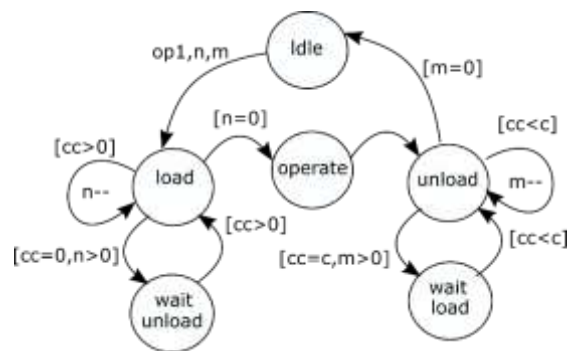
If the buffer is full and "load" is received, nothing is loaded, only it is issued on the out port - "full".

Each upload operation has a duration. It can be implicit and the same for all such elements for simplicity or it can be entered as an additional attribute.

Figure 1.8. State automaton modelling the buffer behavior

### Machines

The machines (figure 1.9) take through materials from the buffers to which they are connected to the ports and deposit materials through the output ports into the buffers to which they are connected.



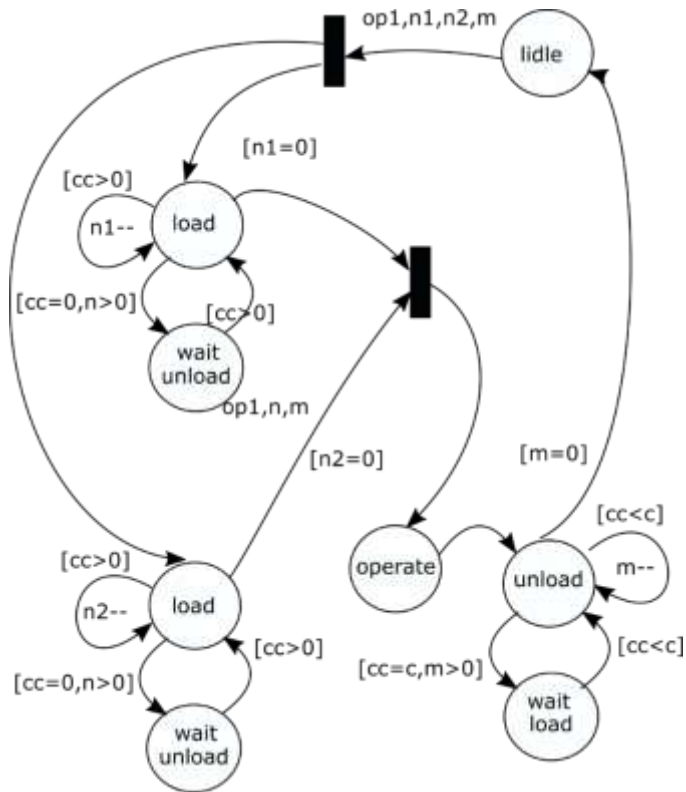
The machine transmits the “unload” command to the input buffer / buffers and the “load” command to the output buffer / buffers.

In the image we have a general machine with a single input buffer and an output buffer, Operation op1 requires  $n$  units from the input buffer and produces  $m$  units in the output buffer. When the op1 command appears on the information input port, the machine tries to load  $n$  units from the input buffer - going into standby mode when the buffer is completely empty and resuming the loading cycle when it is full. Each time the machine loads, it sends the “unload” command to the buffer and reads the  $cc$  attribute of the buffer.

Figure 1.9 State automaton modelling the generic machine behavior

After being loaded into units, the machine enters a processing state that has an associated duration. After the processing time has elapsed, the machine tries to unload  $m$  units into the output buffer - entering the standby state when the buffer is completely full and resuming the unloading cycle when it is empty. Each time the machine loads, it sends the “load” command to the buffer and reads the  $cc$  attribute of the buffer. After  $m$  units are loaded, the machine enters a state of inactivity in which it can receive the next command.

## Workstations



The workstations (figure 1.10) operate at a given time according to a current operation.

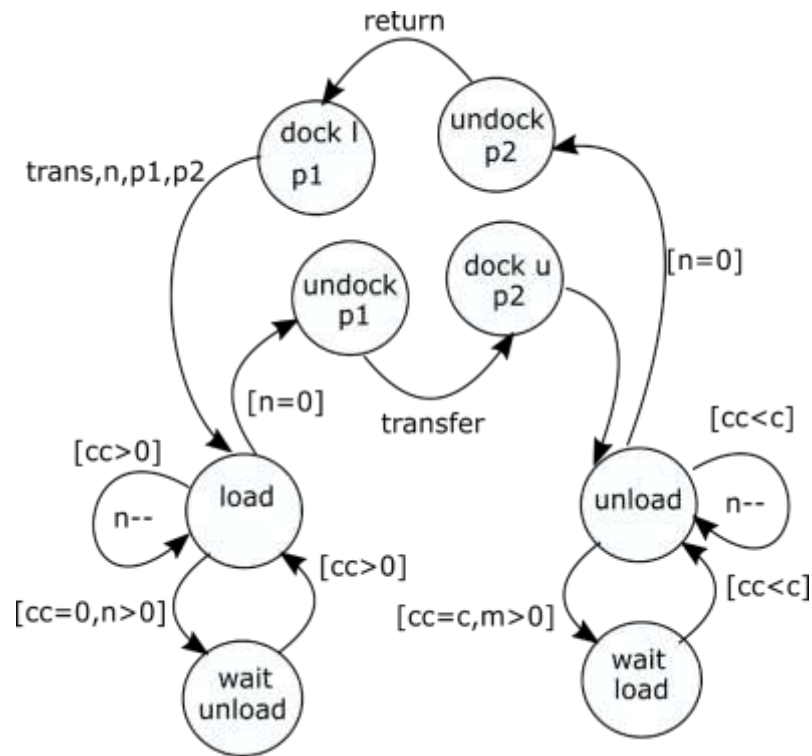
The peculiarity is that a machine can have several input buffers because of which several loading processes can take place in parallel. Only after all the materials are loaded in the appropriate quantities does it move to the processing phase. In the figure a mix with the UML notation (join fork on successive lanes) for a machine with two input buffers and one output buffer. Also, flexible machines can be connected to several input buffers, but for a specific operation they only take materials from some of them.

Figure 1.10 State automaton modelling the behavior of a workstation with two input buffers and one output buffer



## Transport machines

### Autonomus Guided Vehicles (AGV)



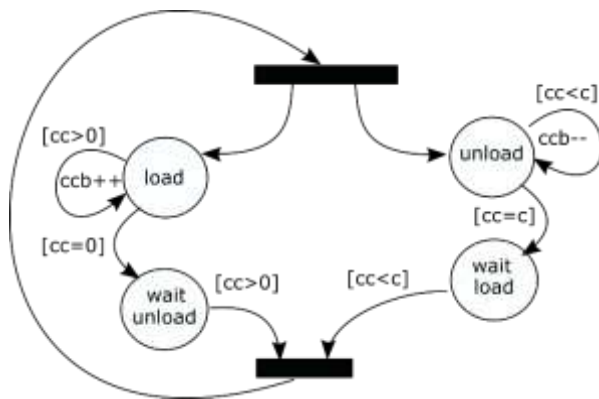
The peculiarity of AGVs is that they have:

- a single input and output buffer
- the same quantity taken from the departure buffer and unloaded at the arrival buffer.

The transported quantity is an attribute and also the position. The processing can be decomposed into at least 2 states (if only the start and end positions are considered) undocking and docking. If several positions are considered, each of them will be a state. The transition between them will be made on the basis of an order coming on the information port to allow the control and synchronization of several AGVs. In the figure 1.11 we have an AGV that oscillates between the buffers p1 and p2. It docks for loading at p1 (dock l). After loading it detaches from p1 and when it receives the transfer command it docks for download at p2 (dock u). After being emptied it detaches from p2 and when it receives the return command it is transferred again to p1 where it is docked for loading

Figure 1.11. State automaton modelling the behavior of an AGV

### Conveyer/belt/pipe



In this case the particularities are the specific start / stop command and the capacity. The quantities of material will be taken from the input buffer / buffers and will be deposited with the delay corresponding to the speed selected in the input buffer. If they have more than one input port, the streams will be cumulated. If they have more than one output port, the streams will be split.

In figure 1.14 is a conveyor with an input port and an output port. The system loads and unloads at the same time. Any stop of loading or unloading stops the transport. Only if both processes are unlocked does the travel process resume. CCB is the current capacity of the bank. The system can be refined but complicates the overall scheme.

Figure 1.12 State automaton modelling the behavior of an conveyer/belt/pipe

### Manipulator

Each command specifies the quantity, the port / Buffer from which they are taken and the port / buffer in which the materials are placed. The manipulator only works if the source and destination are available. The figure 1.13 shows a manipulator that can transfer materials from buffer b1 to buffer b2 and alternatively from b3 to b4.

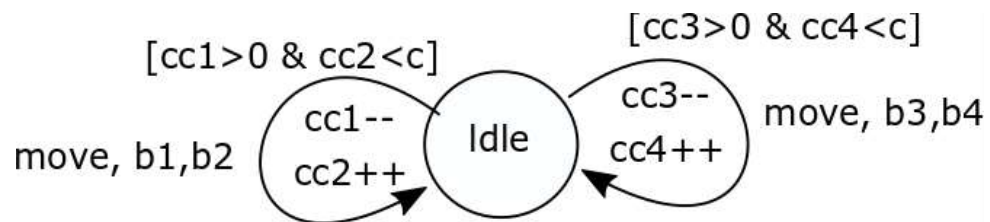


Figure 1.13. State automaton modelling the behavior of an Manipulator

## **Unit 3 Model based design and analysis of manufacturing system**

### **The design problem**

Using the available design tools construct the model of a solution for the manufacturing line design problem described in Unit 2. In this context the design problem consists in finding the right and optimal configuration of such a manufacturing system that can respond to a customer order. A customer order specifies an assortment of products characterized by type of material and quantity that must be produced under some time and cost constraints. In this case the customer order is of chocolate truffles packaged in aluminum bags and in cubic cardboard boxes

### **Design methodology**

The design activity should be divided in three sub-activities presented below (A1-A3) [Mironescu2020].

**A1. Layout design** - in which the number, types, connections, and placement of the components of the line are established. In this phase the designer selects from a toolbox the elements that compose the line. He follows the order of operations (that transform the raw materials in final products) and the organization of the production line: flow shop, job shop, open shop etc. He connects the components with lines, representing each the flow of a specific material between 2 components. This activity will be addressed in this training Unit and the Unit 4

**A2. Operation planning/Scheduling** - At this stage the designer solves the following optimization problem for the production line designed in A1: For a given number of jobs each consisting of a number of product units find a job execution order that minimize one or more aspects of the manufacturing process (for example completion time, and/or delays (due time – job completion time) and/or number of delayed job. This activity will be addressed in Units 5,6 and 7

### **A3. Control system design**

At this stage the designer develops the control system that steers the line to perform the operations corresponding to a given schedule without user intervention. The control system should ensure that the jobs are executed in the prescribed order and with the prescribed timing and that the concurrent processes are coordinated so that no undesired events occur. This activity is addressed in another Training Course.

### **Using the DSML developed in Unit 1**

Figure 3.1 presents the modelling tool generated from the metamodel for MSML designed in Unit 2. The user can compose graphically the model by selecting the graphical representation of the components from the toolbar and placing them on the modelling canvas. The component can be connected by selecting one of the relations present also in the tool bar and picking the components to be connected. In the MSML case the components are exchanging information through ports. The ports are themselves components so they must be placed between the components that should be connected. Connection line modelling the transfer path from machine to port and from port to machine are available. The port can be oriented in 4 directions (up, down, left, right) to indicate the material flow direction. To make the design functional and comprehensible the flow direction should be presented by orienting accordingly the ports. The layout should respect the alignment and orientation of a real manufacturing line. In a real manufacturing line, the material inlets and outlets of the machines are placed so that they allow chaining with short connections.

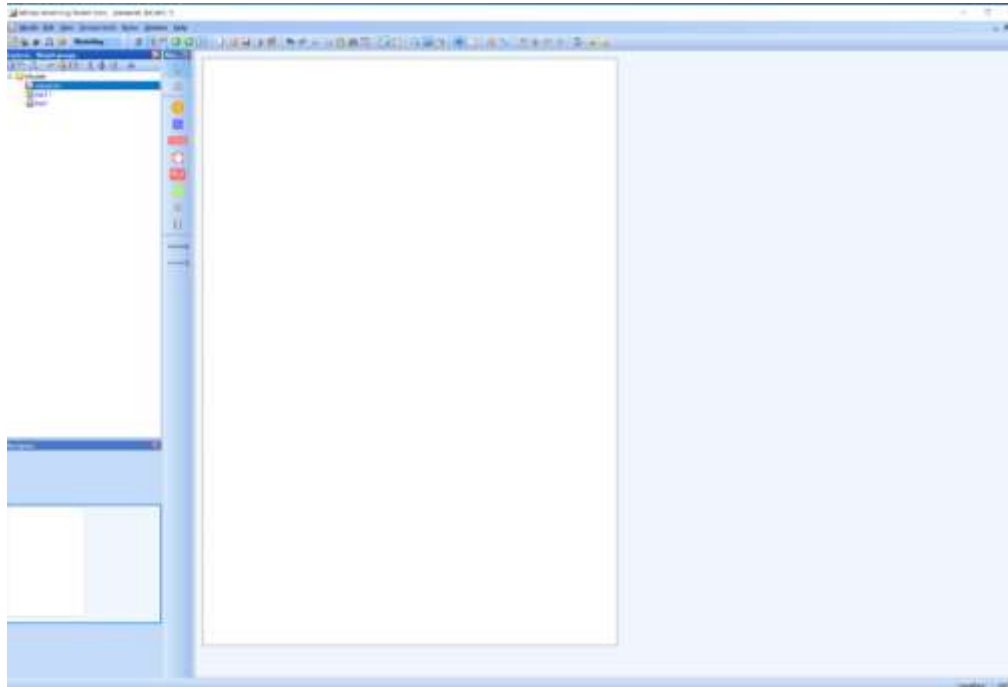


Figure 3.1 Modeling environment for the MSML

After placing the components, the components should be labelled accordingly, and the values of the attributes should be also filled in.

For the Buffers and AGVs the material Type and the Capacity must be at least specified.

For the Workstations, the possible operations must be defined with the type and quantity of materials need as input and the type and quantity of resulting materials.

For Conveyers, Belts, Pipes beside the capacity and material type , a transport time related to the capacity must be also specified

For Manipulators the posible transfer paths must be specified

Figure 3.2 presents the finished model.

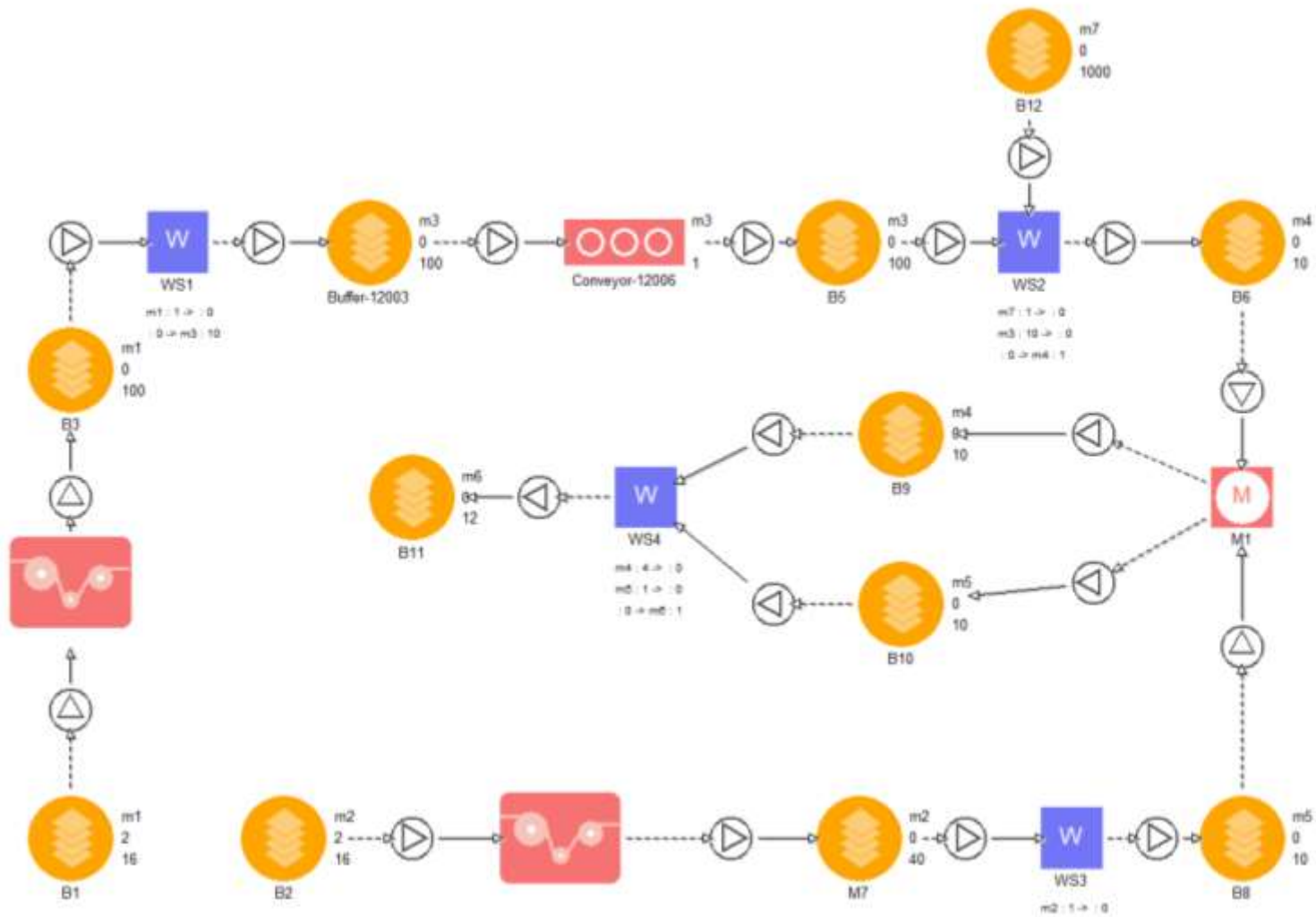


Figure 3.2 Modeling environment for the MSML

## Using Petri Net in Bee Up

In order to extend the possibilities of model (and design) analysis we will transform the MSML model in an equivalent Petri net model. For this an equivalent model for each component must be designed. This is done considering the structure, port connection and also the behaviour of each element described in Unit2

Figure 3.3 presents the Petri net equivalent of the Buffer component. To implement the limited capacity (number of material units) of this component two variants can be employed:

- bounded place can be used if they are supported by the Petri net modeling, simulation and analysis toolchain (e.g. BeeUp) (fig.3.3a)
- a supplementary control place (as employed in the Petri net-based control theory) is provided with an initial marking corresponding to this capacity. (fig 3.3b))

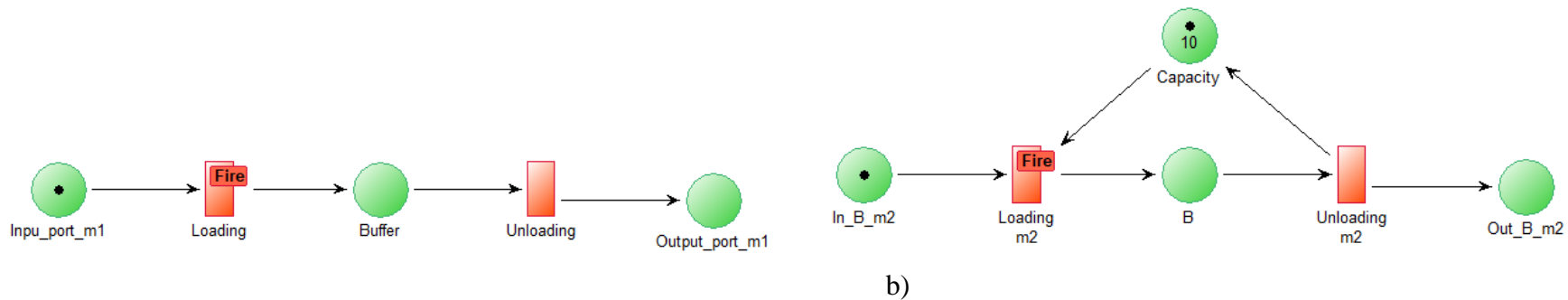


Figure 3.3 Petri Net implementation of a Buffer (B)

a) using bounded places b) using a control place

Figure 3.4 presents the Petri net model corresponding to a component from the Conveyor, Belt and Pipe subclass of the transportation machines class. The Capacity of the transport line is implemented in the arc weight. In the presented example 2 units of the transported material are transferred from the input port In\_t\_m3 to the output port Out\_m3 in each step which corresponds to the TransportTime. The presence of the command place C allows the control of the transport – blocking the transfer if it has a token

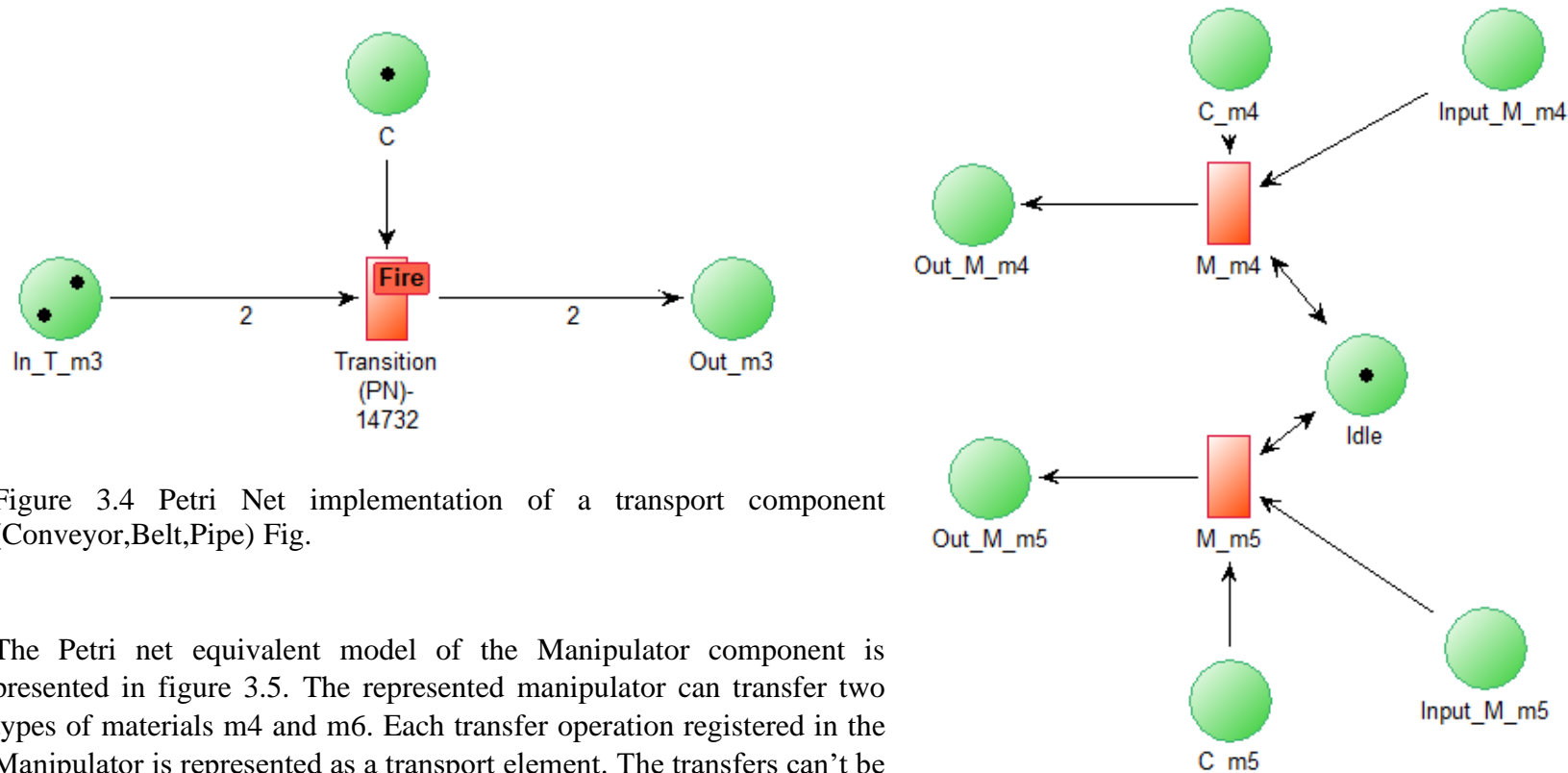
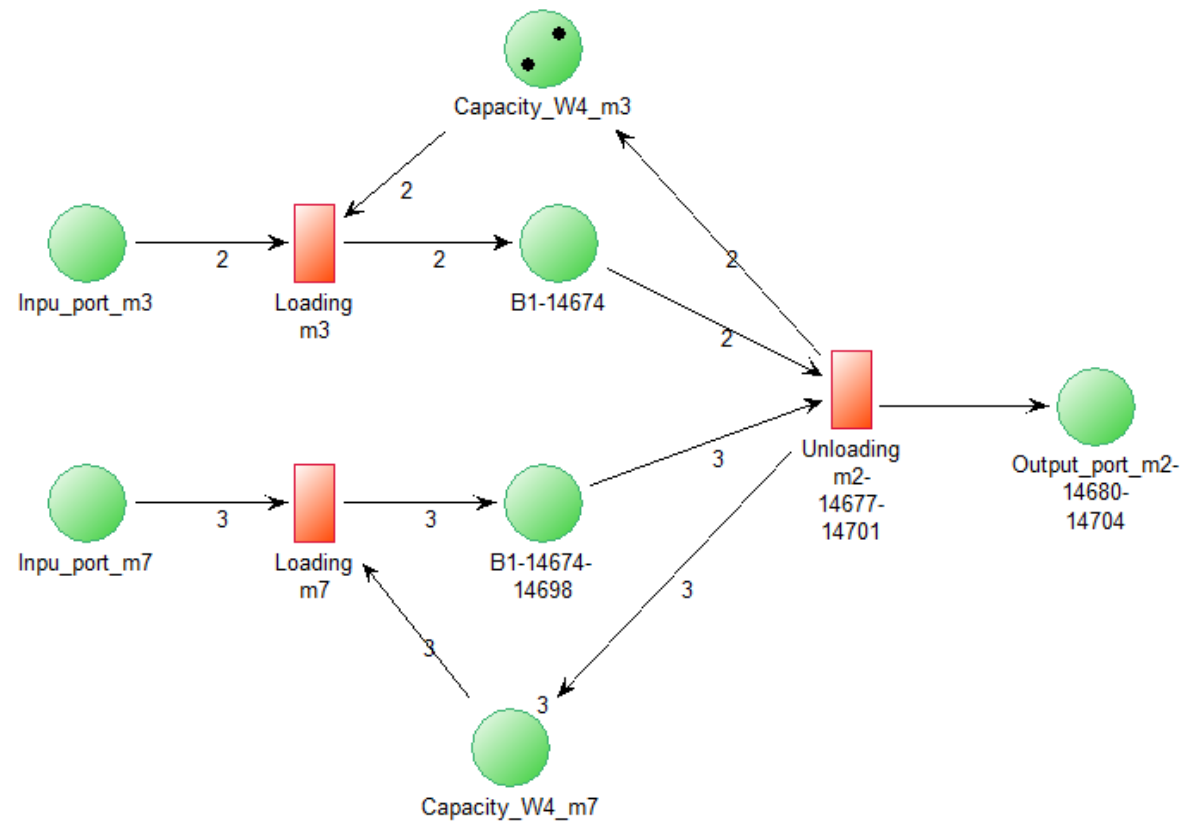


Figure 3.4 Petri Net implementation of a transport component (Conveyor,Belt,Pipe) Fig.

The Petri net equivalent model of the Manipulator component is presented in figure 3.5. The represented manipulator can transfer two types of materials m4 and m6. Each transfer operation registered in the Manipulator is represented as a transport element. The transfers can't be performed simultaneously because there is only one robotic arm performing it that is represented by the token in the place Idle.

Figure 3.5 Petri Net implementation of a Manipulator





For the Workstation the equivalent model is presented in Figure 3.6. The transformation rule is that for each incoming ingredient needed by the operation a net equivalent to a buffer is created complete with limiting capacity place. All these nets are connected through the unloading transition which has as many post-places as resulting materials from the operation. The represented workstation takes 2 pieces of m3 and 3 pieces of m3 and produces 1 piece of m2.

The individual models are the connected by substituting the places representing disconnected input and output ports with one single place representing the connection port.

The resulting Petri net is presented in figure 3.7

Figure 3.6 Petri Net implementation of a Workstation

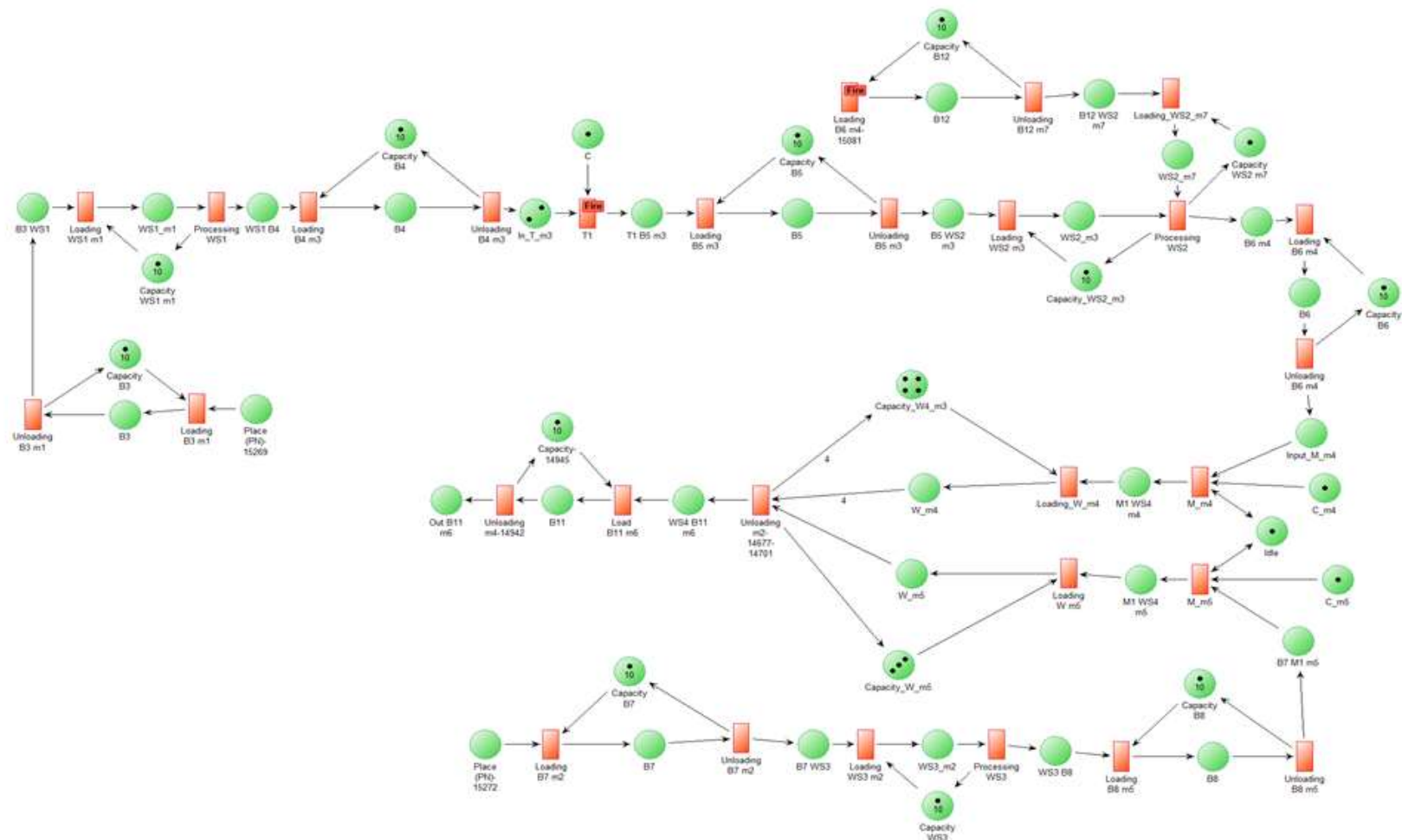


Figure 3.7 Petri net model of the chocolate manufacturing line

## **Analysis tools and methods**

The transformation of the model in the equivalent Petri net allows the qualitative analysis of proprieties of the modeled system using formal methods because Petri nets have a sound formal definition. This allows the proof of the presence or absence of some properties of the net through formal analysis. If we can map these properties to the properties of the of the modelled system, we can prove that this system presents or not this properties. This can be used to verify and validate the model or to validate and verify the system.

For example, one structural property in the Petri net is the connectivity that can be associated to the real connectivity of system components machines in the real system. By analyzing the structure of the net it can be revealed if two places have a direct or indirect connection so that a token can travel from a places to another

The boundedness in Petri net is determined by the maximal number of token that can accumulate in a place. Usually, tokens model resources so the unbounded accumulation of tokens in a place indicate that can be caused by a blocked flow (which is a design flow that conducts to an unsafe accumulation of the modeled resource) or by an unnatural multiplication of the tokens (which is a modelling flow)

The concept of liveness of the net is correlated with the presence or absence of deadlocks in the modelled concurrent system so a structurally live Petri Net is a model for a deadlock-free system.

In order to perform more sophisticated analysis of the state space of the system a suitable equivalent representation must be found As the model in both MSML or untimed Petri net form is a finite state system the best suited structure for a formal analysis is a Kripke structure.

A state of the structure is represented as a tuple of the current material content of all buffers (Bi.Capacity).

The set of atomic propositions AP is the set of proposition constructed with the increment (inci) and decrement (deci) operation on each buffer Bi.

As the initial states determine the reachable states because the transitions are enabled or disabled depending on the buffer content, a Kripke structure will be build for each intended initial state.

An algorithm that constructs the Kripke structure corresponding to a given initial state of the model is given in the following. The algorithm generates the structure iteratively from the initial state.

```
S={ }
L={ }
TR={ }
C={s0}
for each s in C
    for each s' to which it can transition from s conforming to the above rules.
        add s' to C
        add label l(s,ap) to L
        add TR(s,s') relation to TR
        remove s from C
    add s to S
```

The resulted structure can be exported in the .ktz format used by many model checkers. The TINA (Time Petri Net Analyser) toolbox that will be used also in conjunction with Timed Petri nets simulation, contains a A State/Event LTL model checker named selt. Selt can be used to express some properties and testing them for Kripke structures constructed for the manufacturing system model with different initial states.

## **Unit 4 Simulation based analysis of manufacturing system**

The simulation allows a quantitative analysis of the performance of the system in the case in which the number of states of the system is very big or infinite so that a state space analysis is not feasible anymore. We will explore the possibilities of two type of simulation that can be performed with the available tools:

- a) step simulation – can be performed in the ADOxx based modelling and simulation tools
- b) time stimulation - must be performed in an exterior tool as both MSML and Bee-up doesn't have a time based simulation engine

### **Step simulation**

Both modelling languages based on the ADOxx environment use a step based discrete event modelling engine. At each step the possible transition of the system is evaluated and the ones that meet the preconditions are performed. When the transition is performed the state is updated also in the graphical interface.

As the Petri net simulation is not bringing significant information in comparison with the MSML information we will exemplify with the later this type of simulation.

The simulation takes place in ADOxx Modelling Toolkit with the MSML library loaded. The model created in Unit3 should be loaded

The transition to simulation mode is seamless. Only the perspective must be changed in the left-up corner of the toolbar from modelling to simulation. The toolbar will then present a next step tool. By clicking the next step all possible transitions are performed.

The materials are transferred from one buffer to other respecting the behavior rules - mass conservation and operation mass balance. The content of the buffers represents the state of the system so this will be the subject of the analysis.

If we simulate the model defined in Unit 3 and we represent the content of buffers (in material units) in each step, we obtain the plot represented in Figure 4.1.

As the transition to editor is seamless, the content of the buffers can be modified during the simulation. The represented plot is for a simulation that was not modified after the start. Only B1 and B2 are filled at the beginning of the simulation.

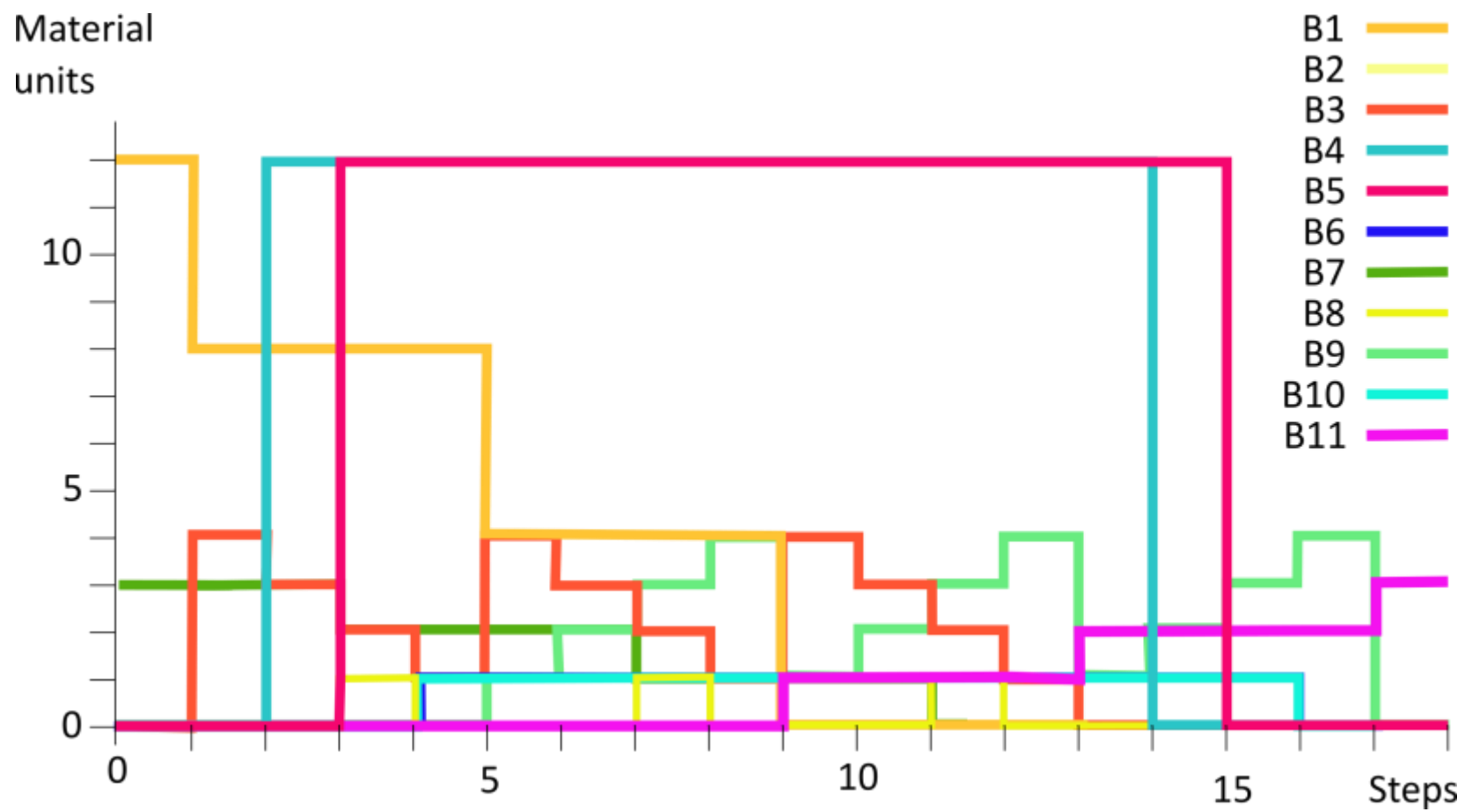


Figure 4.1 Results of the simulation of the manufacturing line

The simulation reveals that:

- the starting values are sufficient for obtaining a final product
- the flow of the production is continuous.
- no unbounded accumulation or intermediary depleting of material take places

In the end all buffers are depleted with the exception of B11 where the final product is accumulated. The whole production cycle takes 17 steps and 3 units of final product are produced in this cycle.

The method is good for preliminary developing and checking the model. For more in depth analysis of the timely behaviour of the design a time based simulation is needed.

The advantage of the ADOxx based development environment is that it provides a simple mechanism of exporting of the model in XML format. The Petri net created from our model in Unit 3 is exported in BeeUp in XML format. This XML is then translated in PNML that can be imported by other tools.

### **Timed simulation**

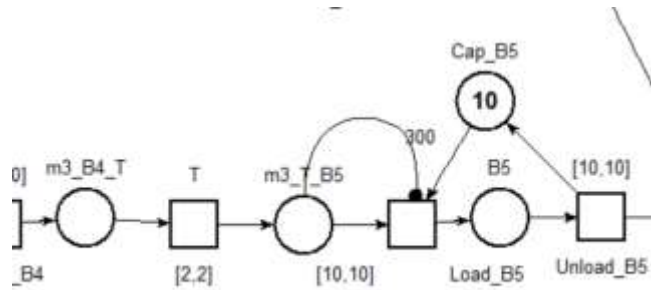
We will use the TINA tool (Time petri Net Analyzer) which contains beside a Petri net editor (nd) also a suite of analysis and simulation tools for timed Petri nets. The Petri nets add the time dimension to the simulation by associating time to the transitions. This time is understood as the time needed by the transition to fire. So when the tokens arrive in the preplaces of the transition, the transition is activated but doesn't fire. Instead a countdown clock is started with the time value associated with the transition. When this clock reaches zero, if the activation tokens are still in place the transition fires as usual. If at least one of the activation tokens were stolen by a concurrent transition, the transition is deactivated and doesn't fire anymore. The particular implementation in Tina introduces time intervals (windows) associated to transitions. If an transition has associated an interval  $[a,b]$  the value of the firing time is an random real variable in the interval  $[a,b]$ . This makes the possibilities of simulating nondeterministic (stochastic) Petri net.

The Petri net elaborated in Unit 3 is the basis for the timed Petri net model of the system that we will use in our simulation. We will start by importing the PNML transformation of the net. The imported net is an untimed net so the first step is to associate to each transition a time interval following the rules:

- if the event is a very short one (can be abstracted as instant) the time interval is  $[0,0]$
- if the event has a fixed duration  $t$  (very tight timed operations) the time interval is  $[t,t]$
- if the event has a variable duration between  $a$  and  $b$  the time interval is  $[a,b]$

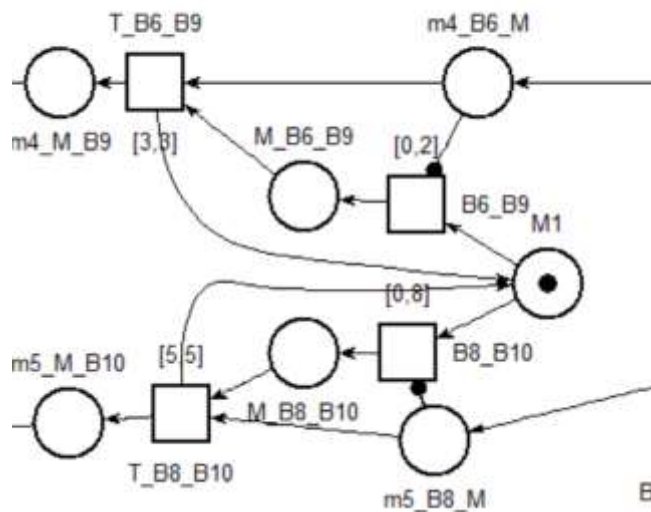






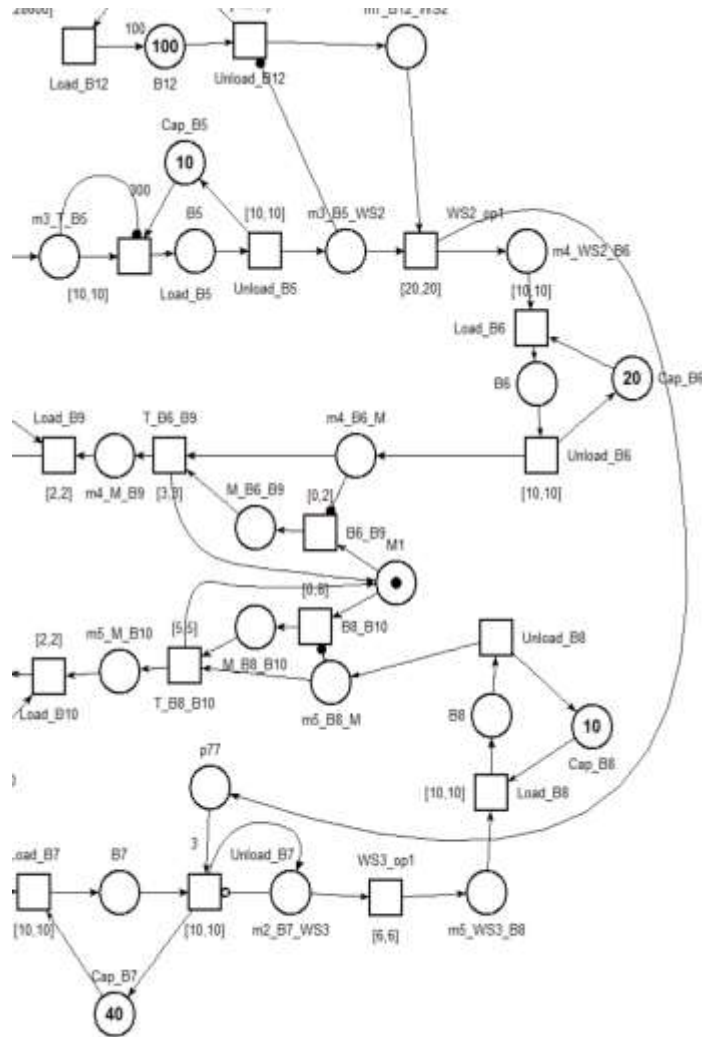
We then refined the behavior of the transport band that is actually a cooling tunnel that transports and cools the mold containing the truffles. The model is presented in figure 4.3. This machine is functioning continuously recirculating the moduls. From the start of the cycle until the first filled mould comes at the end of the transport machine it takes 1h or the equivalent of 300 moulds entering the transport machine. So the read arc will stop the first 300moulds to passing. After that as a mould enters the transport another one is loaded in B5.

Figure 4.3 New model for cooler transport machine



The manipulator needs special attention. The more in depth automation of such a machine needs special attention and a more time. We implemented here a simple solution that approximate enough the behavior of this machine represented in figure 4.4. The token representing the robotic arm when idle is present in place M1. So it has the choice to select between transferring materials from B6 to B9 or between B8 to B10. As the two intervals are  $[0,2]$  and  $[0,8]$  the first choice is more probable as the second. This fits with the fact that the arm must transfer 4 truffle bags in the first case and 1 cardboard box in the second. The separate way also assures that after the choice is made the manipulator cane not be hijacked by the concurrent transition. This is a common solution for such situation in the context of the current implementation of time in Petri nets.

Figure 4.4 New model of the manipulator



Further synchronization is needed between the two lines – the truffle aluminum bags producing line and the cardboard box line. As the cardboards are produced more quickly than the 4 aluminum bags with truffles they will accumulate if the production of the 2 lines is not synchronized. Figure 4.5 presents the proposed solution. A command place (p77) is linked to the unload transition of the buffer B7. Every time a aluminum bag is produced a token is also placed in this place. When 3 tokens are accumulated in the place – signalling that 3 bags were already produced – the machine WS3 is fed with the cardboard sheet to form the box in parallel with the packaging of the 4th bag.

A further simplification was operated. To allow simulation that are longer than a shift, the feeding of B1, B2 and B7 was abstracted. The places corresponding to the ports where eliminated and the transition where “programed” to fire after 8 hours, simulating the replenishing of the stock at the next shift beginning. The model of the final product buffer B11 was similarly modified to allow the emptying of the buffer at the end of shift.

Figure 4.6 presents the complete network.

Before starting the simulation, the initial marking corresponding to the initial state at the beginning of the shift was placed in the net. The B1, B2 and B7 were filled with the number of tokens corresponding the necessary raw materials for a shift. The capacity places (Cap\_B..) were all filled with the number of tokens corresponding to the maximum capacity. The tokens for the AVGs were placed in the docking positions at B1 and B2 and the token for the robotic arm was placed in the idle position M. All other places were left empty modelling empty machines.

Figure 4.5 Synchronization between the lines

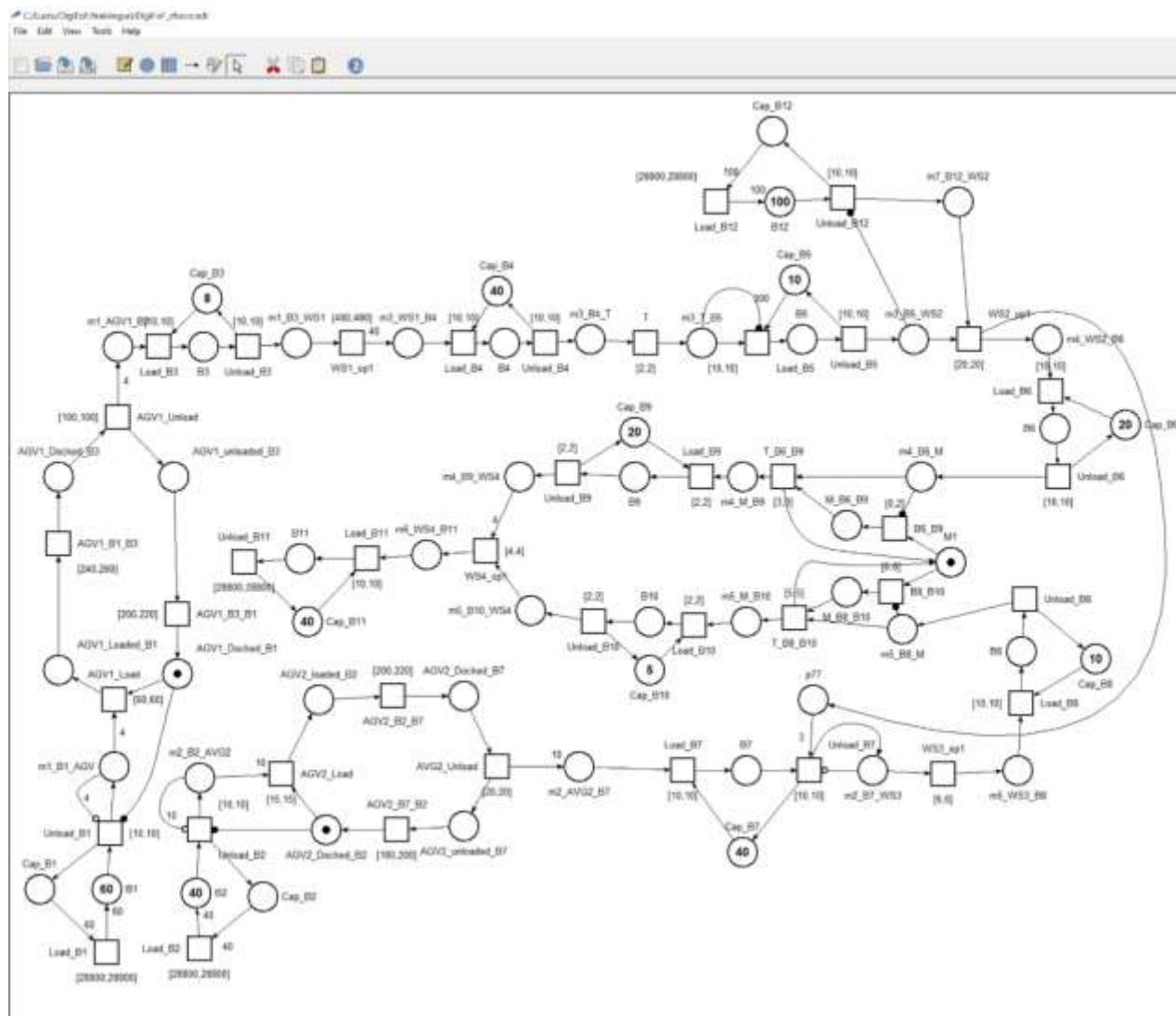


Figure 4.6 Complete time Petri net model of the manufacturing line

With the initial marking the simulation can be started. This can be done interactively using the stepper simulation accessible in the tools menu or in the command line with the tool play.

The simulation windows is presented in figure 4.7. The upper toolbar has controls for driving the simulation: a player like set of buttons for stepping forward (>), backward (<), rewinding to the begin or fast forwarding to the end. These buttons control the navigation of the current simulation path.

There is also a button that starts the automatic (without user interaction) performing of simulation.

The status bar below has on the left the control panel for the next step and on the right the simulation time display.

In a step by step simulation the user is always in control. The program evaluates the initial markings. When a transition has all preplace occupied by sufficient tokens it is displayed either in red (if it can fire immediately) or in gray (it is activated but can fire only after a delay). If the transition is only activated the associated time interval is displayed in red. The user can click on any red transition and trigger it. After all red transition are fired the user has the possibility to advance on a local time scale the time between 0 and the earliest firing time from all the gray transitions - the minimal value from the lower bounds of their firing time windows. In the moment it reaches this firing time the corresponding transitions change to red. The total simulation time is also advanced. The time can be also advanced automatic to the minimal value from the higher bound firing time of their firing time windows by clicking the max button. All the red transition can be fired manually as explained before. The random character of the simulation is determined only by the user choices. He chooses in which order to fire the red transitions or at which position of their firing time window to fire them.

If the rand button is clicked the program advances automatically the simulation. It selects randomly from the red transitions the one to fire. For the grey transitions it generates a random number in the bounds of the firing time window. From all generated random number it selects the smaller and advances the simulation to this point and fires the corresponding transition.

The two simulation types can be mixed by stopping the automatic simulation at a certain point and continuing manually from there or starting manually for the initial phase and continuing automatic for the stable phase or alternating the types.

At any point in the simulation, the user can navigate on the trajectory described by the system until this point using the navigation buttons and change its choices exploring alternative paths in simulation.

Figure 4.7 presents a simulation in performing



By analyzing the results, qualitative information about the capabilities of the designed system can be acquired:

- the productivity of the system is given by the ratio between the number of units of final product present at the end of the simulation in the corresponding buffer (B11) and the simulation time. This can be analyzed globally (one shift) or on certain time periods (for example at beginning of the shift until the line is filled, or at the end of the shift). If randomness is incorporated many runs are needed and the distribution of results can give information about the variability of the process
- by maximizing the material input and performing the simulation until the depletion of the initial material the maximal capacity and productivity in final product can be determined
- the accumulation of tokens in buffers indicates a desynchronization between the different stages of the manufacturing line that can be solved like in reality by modifying the speed of material flow or by introducing control links.
- the buffers depleting too fast or long time empty is a sign that they are under dimensioned or are under feed. The capacity of the buffers and the material flows can be adjusted in the model and verified by a new simulation.
- the time spent by tokens in places can be used to identify resource utilization either by only monitoring the time spent in idle places or by extending the monitoring to all states.

## Unit 5 Scheduling in manufacturing system

### Key concepts in scheduling

Scheduling in a manufacturing line concept is the activity of elaborating a production schedule. A production schedule is a short-term execution plan. We consider a manufacturing line composed of machines that can handle at most one task at a time. A fixed number of jobs with some given characteristic (i.e., tasks, the necessary sequential constraints, the time estimates for each operation and the required resources, no cancellations). The jobs must be processed by the manufacturing line in a way that satisfy some (mostly time but not limited to) constraints. The performing of the task needs beside the corresponding machines also other resources (input material, human labor, energy, etc.) that are limited and must be shared by all jobs. The scheduling is then an assignment problem. The schedule is then a list of tuples of the form (resource, task, begin time, end time). Normally the resource is the machine to which the tasks map naturally, all other resources being mapped to the task.

A feasible schedule is a schedule that allows the completion of all jobs without violating any constraints.

A particular schedule determines particular values for the starting and completion times of the jobs, idle time of resources, lateness of jobs (difference between job completion time and Job due times)

An optimal schedule is feasible solution that minimize some of this values in the context of multiple jobs (for example the completion time for all jobs, the total idle time for the resource).

In this view a scheduling problem is an optimization problem. We present the usual mathematical description of such a problem.

A scheduling problem is represented by a triple  $\alpha | \beta | \gamma$ , in which resources are allocated to orders (with the above properties). The design of the resources ( $\alpha$ ), the running properties and constraints ( $\beta$ ) and the target function to be minimized ( $\gamma$ ) is specified by the triple.

We use following notation

$n$	$\in \mathbb{N}$	Number	of	orders
$m \in \mathbb{N}$	number of machines (= resources)			
$i \in \{1, \dots, m\}$	(mostly) index for a machine			
$j \in \{1, \dots, n\}$	(mostly) index for an order			
$S_i$	schedule (flow chart) on machine $i$ , permutation of a (partial) set $\{1, \dots, n\}$			



$S_i(j) \in \{1, \dots, n\}$  position of order  $j$  in process on machine  $i$   
 $p_j \in \mathbb{N}_0$  processing time of order  $j$  on one machine  
 $p_{ij} \in \mathbb{N}_0$  processing time of job  $j$  on machine  $i$   
 $r_j \in \mathbb{N}_0$  arrival time of job  $j$ , i.e. earliest time at which a job can be scheduled (release date)  
 $d_j \in \mathbb{Z}$  delivery date of job  $j$ , i.e. Time at which an order should be completed (due date)  
 $w_j \in \mathbb{N}_0$  weight of job  $j$ , e.g. Value, cost, priority etc. (weight)  
 $C_{ij} \in \mathbb{R}^+$  completion time of job  $j$  on machine  $i$ . This value depends on the schedule  $S$  to be selected and, if applicable, on the arrival times  $r_j$ .  
 $C_j \in \mathbb{R}^+$  completion time of job  $j$ . If there is more than one machine ( $m > 1$ ),  $C_j := \max \{C_{1j}, \dots, C_{mj}\}$  is defined.

$\alpha$  can take only one of the following values

- 1 – only one machine
- $P_m$  -  $m$  identical machines that run in parallel. A job needs (and may only be processed on) one machine.
- $F_m$  -  $m$  machines in a flow shop, every job must be carried out on every machine. The order of the machines is predetermined and the same for all jobs
- $J_m$  -  $m$  machines in a job shop. The order is prescribed but not necessarily the same for all jobs.
- $O_m$  -  $m$  machines in an open shop. The order in which the processing happens is arbitrary.

$\beta$  can have multiple entries or no entry.

- $p_j = p$  - all orders have the same processing time.
- $d_j = d$  - all orders have the same delivery date.
- $r_j$  - arrival times must be taken into account, i.e. that an job  $j$  may not be processed before time  $r_j$ .
- pmtn (preemption) - if present in field  $\beta$ , orders may be interrupted and resumed as often as required without increasing the processing time for this order. It is generally assumed that orders must be processed on one machine without interruption. (In the same way, it can be assumed that an order may be interrupted, but that it will need all of its processing time to be completed once it has been resumed).
- prec - there are precedence constraints. A priority relationship  $i \rightarrow k$  means that order  $k$  may only start after order  $j$  has ended. Corresponding conditions can occur in single-machine models as well as in multi-machine models.
- $s_{jk}$  - sequence-dependent setup times (setuptimes) between job  $j$  and job  $k$ . If job  $k$  is executed on a machine after job  $j$ , the machine must remain unoccupied for  $s_{jk}$  time units.  $s_{0k}$  describes the set-up time when job  $k$  is first executed on a machine. If there are set-up times that are independent of the sequence, these can be integrated into the processing time in the modeling and do not require any explicit consideration.

pmu - only affects flow shops. Forces that all jobs are executed in the same order on the machines

nwt – no wait time- the processing on the next machine must start immediately after is finished on the current machine

$\gamma$  can have one of the following values signifying the value that should be minimised.

$C_{\max}$  - the total duration (makespan) of a schedule corresponds to completion time of the last completed job  $C_{\max} := \max \{C_1, \dots, C_n\}$ .

$L_{\max}$  -  $L_{\max} := \max \{L_1, \dots, L_n\}$  where  $L_j$  the delay in an job  $j$  is  $L_j := C_j - d_j$  (lateness).

$\sum C_j$  - the total completion times. By minimizing this sum (interim) inventory costs can be reduced. As  $\min \sum C_j$  is equivalent to minimizing the mean lead time ( $\sum C_j / n$ ) and so minimizing the "waiting time" (= Storage time).

$\sum w_j C_j$  - the sum of the weighted completion times, e.g. weight can be storage cost rates.

$\sum T_j$  - the sum of missed deadlines  $T_j := \max \{C_j - d_j, 0\} = \max \{L_j, 0\}$ .

$\sum w_j T_j$  - the sum of the weighted missed deadlines, in which the deadline for individual Orders weighs worse than others.

$\sum U_j$  - the number of missed deadlines  $U_j := (1 \text{ if } C_j > d_j \text{ } 0 \text{ otherwise})$ .

$\sum w_j U_j$  - Even if deadlines are missed, the Define the weighted number of missed deadlines.

$\sum Y_j$  - the sum of late work  $Y_j =$  time units of an order  $j$  that are after the delivery date  $d_j$ ,  $Y_j := \min \{T_j, p_j\}$ .

$\sum w_j Y_j$  - the sum of weighted late work

### **A single machine process model**

For example, we consider a packaging machine that can pack different food products in the same type of package. A job is characterized by the type of packaged product and the number of units (processed or produced). The time needed for operation depends on the number of units, but it is also important to reduce the job completion time so that the product has short intermediary storage time. In this case a weight is associated with each job – according to the product type.

As we don't have other constraints, we can express the scheduling problem as

$$(1 \mid \mid \sum w_j C_j)$$

The parameters of the jobs are presented in Table

Table 5.1 Parameters of jobs for the example

j	1	2	3	4	5
$p_j$	3	6	5	7	6
$w_j$	10	12	3	8	1

### Elaborating a schedule for the proposed model

A helpful graphical tool for building schedules is a variant of Gant chart. The diagram has the time on abscissa and the resource (machine) on the ordinate. Each job is represented as a colored rectangle. Each job  $j$  has the length equal to  $p_j$ . The height is not associated by default to any parameter. Building a schedule means placing the rectangles representing the jobs on the diagram so that the constraint specific to the scheduling problems are maintained.

So from  $(1 \mid \mid \sum w_j C_j)$  following rules are derived

- single machine case - we have a single row on the ordinate. The rectangles representing jobs are all aligned to this row as they are all executed on the same machine.
- no pre-emption - the rectangle are not split.
- the machine can execute one job at the time so the rectangle can't overlap.
- the rectangles are aligned to the starting times of the jobs
- no setup time so the rectangle can be aligned to the end of the previous rectangle = the next job can start immediately as the current job is finished
- weight are used so they are printed on the task

In this training unit we will build only a feasible schedule, so the only condition is to put all jobs on the schedule without overlapping and chain them one to another. The most obvious way to make this is to place them in their numbering order (that can be interpreted also as their arrival order). The result is presented in figure 5.1

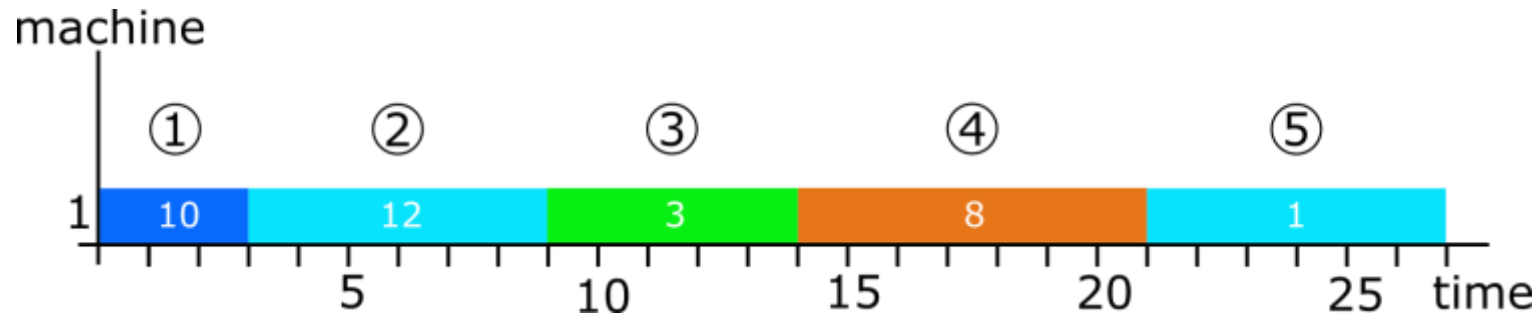


Figure 5.1 A feasible schedule for the example problem

For this schedule the completion time of task 1 is 3 ( $=p_1$ ), the completion time of task 2 is 9 ( $=p_1 + p_2$ ).

Generally for the job  $j$  the completion time is  $C_j = r_j + p_j$

The release time for the job is the sum of the processing time of the previous jobs  $r_j = \sum_{k=1}^{j-1} p_k$ , with  $k = 1, \dots, j-1$

$C_j = \sum_{k=1}^j p_k$  with  $k = 1, \dots, j$  but also  $C_j = C_{j-1} + p_j$

As a result the completion times are presented in Table 5.2

Table 5.2 Completion time

$j$	1	2	3	4	5
$C_j$	3	9	14	21	27
$w_j$	10	12	3	8	1

The objective function has in this case the value  $\sum w_j C_j = 3 \cdot 10 + 9 \cdot 12 + 14 \cdot 3 + 21 \cdot 8 + 27 \cdot 1 = 375$

### Exercise

1. Try to find a optimal schedule or at least a better schedule than that presented

### A multi-machine model

In the case of multiple machines the organization of the shop (Parallel, Flow, Job, Open) is essential adding specific constraints to the schedule. We consider a manufacturing process with 4 machines (1) heating, (2) moulding, (3) tempering and (4) packaging. The jobs must be executed on all machine, in the same order from heating to packaging so the organisation is of a flow shop. We consider two jobs:

- 1 a big lot of chocolate bars with simple forms.
- 2 a small lot of chocolate products with individual forms and complicated moulds

As a results the heating and packaging time will be bigger for the bigger lot, and the moulding and tempering time will be bigger for the lot with complicated moulds.

The processing time for the jobs on the 4 machines are presented in table 3

Table 5.2 Parameters of jobs from the flow shop example

j	1	2
p <sub>1</sub> j	4	1
p <sub>2</sub> j	1	4
p <sub>3</sub> j	2	2
p <sub>4</sub> j	4	1

Depending of the possibility of interchanging the jobs on a machine the scheduling problem of reducing the total duration of the processing can be formulated either as

$(F4|prmu|C_{max})$  when the jobs can't be interchanged

or as

$(F4||C_{max})$  when the jobs can be interchanged

### **Elaborating a schedule for the proposed model**

So from  $(1||\sum w_j C_j)$  following rules are derived

- multiple machine case - we have a multiple rows on the ordinate.
- Each job is executed on each machine so they are multiple rectangles representing the tasks of the same jobs each aligned to the row of the machine of which they are executed.
- no pre-emption - the rectangle are not further split.
- each machine can execute one job at the time so the rectangle can't overlap.
- the rectangles are aligned to the starting times of the jobs
- no setup time so the rectangle can be aligned to the end of the previous rectangle = the next job can start immediately as the current job is finished
- the next task of the job can start on the next machine only if the current task of the job is finished on the current machine.

The release time of the  $i$  task of a job on a machine  $i$  is equal to the completion time of the precedent task on the preceding machine

$$r_{ij} = C_{i-1,j} \text{ for } i > 1$$

$C_{1j}$  is depending of the job order and on the interchange clause.

Following this rules only two feasible schedules are possible for the  $(F4|pmu|C_{\max})$  case . They are presented in figure 5.2. Figure 5.2a describes the schedule starting with job 1( $S_1$ ) and figure 5.2b the schedule starting with job 2( $S_2$ ).  $C_{\max}(S_1)=12$  and  $C_{\max}(S_2)=13$  so  $S_1$  is the optimal schedule

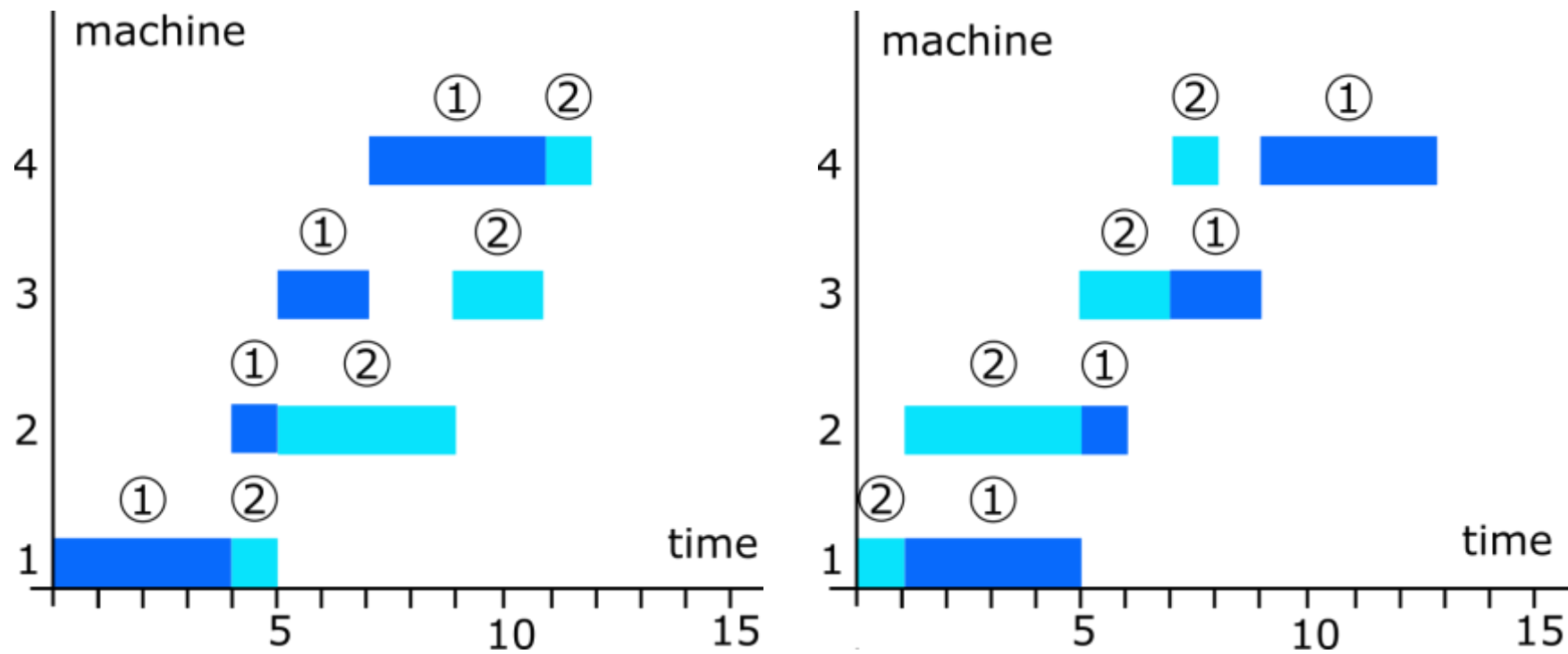
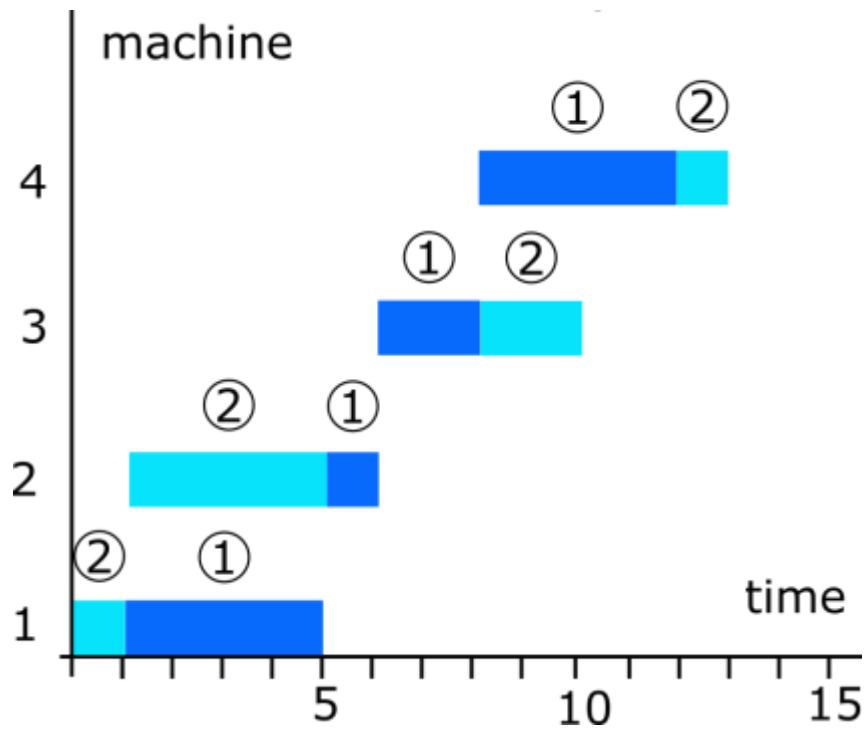


Figure 5.2 The schedules for the example problem ( $F4|prmu|C_{max}$ )

a)  $S_1$  starting with job 1 b)  $S_2$  starting with job 2





For the case  $(F4 || C_{\max})$  theoretical they are 16 possible schedules – including the 2 presented before. In figure 5.4 an example is presented in which an interchange occurred in machine 3.

The  $C_{\max}$  for this schedule is 13.

Figure 5.4 A feasible schedule for the example problem  $(F4 || C_{\max})$

## Unit 6 Single objective optimization of the schedule

### Single objective optimization

In this case only one objective function must be minimized.

### Optimization of the schedule for one machine case

#### Example 1 (Lateness)

For a manufacturing line with one machine we consider the set of 5 jobs described by the parameters (processing times and delivery dates) from table 6.1

Table 6.1 Parameters of jobs for the single objective optimization problem

j	1	2	3	4	5
<b>p<sub>j</sub></b>	5	9	8	4	3
<b>d<sub>j</sub></b>	3	12	7	18	5

We want in this case to find a schedule that minimize the maximal lateness (delay). The problem is then

$(1 | \max L_j)$  with  $L_j = C_j - d_j$

#### Solution

The solution to this type of problems is to sort the jobs according to their ascending delivery time. The sorted list is then the searched schedule.

In this case the table 6.2 summarizes the principal parameters of the solution together with the job parameters for the ease of understanding

Table 6.2 Parameters of the solutions

j	1	5	3	2	4
<b>p<sub>j</sub></b>	5	3	8	9	4
<b>d<sub>j</sub></b>	3	5	7	12	18
<b>C<sub>j</sub></b>	5	8	16	25	29
<b>L<sub>j</sub></b>	2	3	9	13	4

The solution is then  $S=(1,5,3,2,4)$ . As proved in [Jaehne2019] this is the optimal solution – the schedule with the minimal maximal lateness. For this schedule it is the lateness of job 2  $L_2=13$

### Example 2 (Number of late orders)

For the same machine we consider the set of jobs presented in table 6.3

Table 6.3 Parameters of jobs for the single objective optimization problem

j	1	2	3	4	5
<b>p<sub>j</sub></b>	7	3	3	6	2
<b>d<sub>j</sub></b>	8	9	11	13	16

An order  $j, j \in \{1, \dots, n\}$  is late if  $C_j > d_j$ . We want in this case to minimize the number of late orders, i.e. minimize  $|\{j \in \{1, \dots, n\} \mid C_j > d_j\}|$ .

### Solution

In this case the goal is to reduce the number of late order not to reduce the latencies of the order that are late. As a result the orders that are inevitably late can be scheduled at the end. A method was developed by Moore in 1968 based on this idea. We start with a schedule in which the jobs are already sorted and renumbered in the order of not descending delivery times  $d_j$ .

Algorithm 1 (Moore method):

1. Initialization: Sort the jobs non-descending according to delivery dates, so let  $d_1 \leq \dots \leq d_n$  be assumed. Let the number of late order  $U := 0$ .
2. Stop criterion: If  $C_j \leq d_j \forall j \in \{1, \dots, n\}$  with  $S(j) \leq n - U$ , then stop. There are no more jobs that will be late.
3. Determination of the job to be postponed: Let  $k := \operatorname{argmin}_{j \in \{1, \dots, n\}} \{S(j) \mid C_j > d_j\}$  be the earliest scheduled, late job. Further, let  $l \in \operatorname{argmax}_{j \in \{1, \dots, n\}} \{p_j \mid 1 \leq S(j) \leq S(k)\}$  be an job with the longest processing time among the first  $S(k)$  orders.
4. Postponing the order: Define the following schedule:  
 $S'(j) = n$  if  $j = l$  |  $S'(j) = S(j) - 1$  if  $S(l) < S(j)$  |  $S'(j) = S(j)$  otherwise  
set  $S := S'$ ,  $U := U + 1$  and go to step 2.

To solve our example, we start with the configuration in table 6.4. We have 4 delayed jobs (2-5)

Table 6.4 Parameters of the starting schedule

j	1	2	3	4	5
<b>p<sub>j</sub></b>	7	3	3	7	2
<b>d<sub>j</sub></b>	8	9	11	12	16
<b>C<sub>j</sub></b>	7	11	14	20	22

*Iteration 1*

step 2: there are more the 1 late jobs

step 3: the earliest scheduled late job is 2 so  $k=2$ . The job with the bigger processing time is 1. So the job  $l=1$  will be postponed at the end of the schedule

step 4:  $U=1$  and the new schedule is presented in table 6.6

Table 6.5 Parameters of the schedule after the first iteration

j	2	3	4	5	1
<b>p<sub>j</sub></b>	3	3	7	2	7
<b>d<sub>j</sub></b>	9	11	12	16	8
<b>C<sub>j</sub></b>	3	6	13	11	18

### Iteration 2

step 2: there is 1 late job(4) in the first  $n-U=4$  jobs

step 3: the late job is 4 so  $k=4$ . The job with the bigger processing time is also 4. So the job 4 will be postponed at the end of the schedule

step 4:  $U=2$  and the new schedule is presented in table

Table 6.6 Parameters of the schedule after the 2<sup>nd</sup> iteration

j	2	3	5	1	4
<b>p<sub>j</sub></b>	3	3	2	7	7
<b>d<sub>j</sub></b>	9	11	16	8	12
<b>C<sub>j</sub></b>	3	6	13	20	27

### Iteration 3

step 2: there are no more late jobs in the first  $n-U=3$  jobs. The search stops and the current schedule is the optimal schedule with only 2 jobs that are late.

### Example 3 (Weighted completion time)

In the same context let consider the set of 4 jobs presented in table 6.7

Table 6.7 Parameters of the jobs for the example 3

j	1	2	3	4
p <sub>j</sub>	3	5	6	8
w <sub>j</sub>	5	10	9	1

Let us consider the scheduling problem  $(1 \mid \mid \sum w_j C_j)$

### Solution

We start from the schedule  $S(j)=j$  presented in table 6.8

Table 6.8 Parameters of the jobs for the starting schedule

$j$	1	2	3	4
$p_j$	2	2	3	9
$w_j$	6	9	11	2
$C_j$	2	4	7	16

The objective function is  $\sum w_j C_j = 6*2 + 9*4 + 11*7 + 2*16 = 157$

The method used here is to sort the jobs in the ascending order of their  $p_j / w_j$  values. This sorting rule is called WSPT (weighted shortest processing time) or Smith rule.

Following the rule the resulting schedule is presented in table 6.9

Table 6.9 The optimal schedule for the example 3

$j$	2	3	1	4
$p_j$	2	3	2	9
$w_j$	9	11	6	2
$p_j/w_j$	2/9	3/11	1/3	9/2
$C_j$	2	5	7	16

The objective function is  $\sum w_j C_j = 9*2 + 11*5 + 6*7 + 2*16 = 147$

## Optimization of the schedule for multi machine case

### Example

We consider the two machine flow shop problem ( $F2 \parallel C_{\max}$ ) the only scheduling problem that can be solved exactly in polynomial runtime

For this we consider the set of jobs from table 6.10

Table 6.10 Parameters for the jobs

j	1	2	3	4
p <sub>1</sub>	4	1	3	2
p <sub>2</sub>	1	4	2	6

### Solution

In the solving of the problem the so-called "Johnson rule" is used, which is a priority rule based on modifications of the Shortest Processing Time (SPT) and Shortest Processing Time (LPT) rules. While the SPT rule and the LPT rule are based on the entire processing time of an job on all machines, the STT rule (shortest task time) is based only on the processing time of an job on one machine. Similarly, the LTT rule (longest task time) selects the next job on a machine based on which job has the longest processing time on that machine. Because only two machines are used only one permutation of the job is required to solve the problem.

### Solving the problem

Algorithm (Johnson's algorithm):

1. Initialization:

Let  $J$  be the set of all orders and  $J_1 := \{j \in J \mid p_{1j} < p_{2j}\}$  the set of orders whose processing time on machine 1 is shorter than on machine 2

2. Determining

the

order:

First schedule all jobs from  $J_1$  according to the STT rule (related to machine 1).  
Then schedule the remaining orders according to the LTT rule (based on machine 2).

For the example above we have  $J_1 = \{2, 4\}$

So the order of scheduling is

2 ( $p_{12} < p_{14}$ ) following the STT rule on machine 1

4 last from  $J_1$

3 ( $p_{23} > p_{21}$ ) following the LTT rule on machine 2

1 last job to be scheduled

The table 8.11 contains the schedule The objective function is  $C_{\max} = 14$ .

Table 6.11 Parameters for the jobs

j	2	4	3	1
$p_{1j}$	1	2	3	4
$C_{1j}$	1	3	6	10
$p_{2j}$	4	6	2	1
$C_{2j}$	5	11	13	14



## Unit 7 Multiple objective optimization of the schedule

Because even the single objective optimization problems are in the category of hard NP-complete problems and can be solved only using heuristic, in order to solve multiple objective optimization problems multiple upper-level multi-objective methodologies (i.e., *meta-heuristics*) must be employed. Metaheuristic' search methods can be defined as upper level general methodologies guiding strategies in designing heuristics to obtain optimization in problems.

We will present one class of such problems and the methodology use to solve them. These problems are closely related to the manufacturing line presented in the previous modeling and simulation units

### The PFSP (Permutation Flow Shop Problem) with Unlimited Buffers

The PFSP consists of  $m$  machines arranged in series in which, a set of  $n$  jobs must be processed in the same sequence. Supplementary to the standard description and notation of Flow Shop problems there are infinite capacity buffers between machines. The setup times are included in the routing times of operations and the process cannot be interrupted at any time.

Let:  $\Pi = (\pi(1), \pi(2), \dots, \pi(n))$  be a solution of the permutation flow shop scheduling problem, where  $\pi(i)$  corresponds to the job in position  $i$  in the sequence  $\Pi$ ,  $N$  the number of jobs,  $M$  number of machines,  $t_{\pi(i),j}$  the processing time of the job in position  $i$  in the sequence on machine  $j$ ,  $d_{\pi(i)}$  the due date of job in position  $i$  in the sequence,  $T_{\pi(i)}$  the tardiness of job in the permutation and  $C_{\pi(i),j}$  the completion of the job in the position  $i$  on the machine  $j$  that can be recursively computed as follows.

$$C_{\pi(1),1} = p_{\pi(1),1}$$

$$C_{\pi(i),1} = C_{\pi(i-1),1} + p_{\pi(i),1}$$

$$C_{\pi(1),j} = C_{\pi(1),j-1} + p_{\pi(1),j}$$

$$C_{\pi(i),j} = \text{Max}(C_{\pi(i-1),j-1}, C_{\pi(i),j-1}) + p_{\pi(i),j}$$

Where,  $i=2 \dots n$  and  $j=2 \dots m$ , noticeably, the total amount of time required to accomplish processing of all jobs denoted makespan  $C_{\max}$  is given by  $C_{\pi(N),M}$  that represents the completion time of the last job on the last machine

Moreover, the tardiness of job in position  $i$  in the permutation is obtained by the following equation

$$T_{\pi(i)} = \text{Max}(C_{\pi(i),M} - d_{\pi(i)}, 0) \quad (6)$$

Correspondingly, the maximum tardiness is expressed as

$$T_{\max} = \max_{1 \leq i \leq N} (T_{\pi(i)})$$

### The PFSP with Limited Buffers

The blocking flow shop scheduling problem (BSFP) is the permutation flow shop scheduling problem (PFSP) with a supplementary constraint - the capacity of storage between two successive machines is limited or zero. As modeled in the MSML, if the next buffer is full or with null capacity the current job is blocked in the current resource until the next resource is available for its execution. Given that  $R_{\pi(i),j}$  represents the start times of the job in the position  $i$  in the machine  $j$ , the make span and maximum tardiness are computed as follows:

$$R_{\pi(i),j} = \text{Max}(C_{\pi(i-1),j}, C_{\pi(i),j-1} + R_{\pi(i-1),j+1})$$

$$C_{\pi(i-1),j} = R_{\pi(i-1),j} + t_{\pi(i-1),j}$$

$$C_{\pi(i),j-1} = R_{\pi(i),j-1} + t_{\pi(i),j-1}$$

$$T_{\pi(i)} = \text{Max}(C_{\pi(i),M} - d_{\pi(i)}, 0)$$

$$C_{\max} = C_{\pi(N),M}$$

$$T_{\max} = \max_{1 \leq i \leq N} (T_{\pi(i)})$$

### Multi Objective Optimization Problems

The problem that will be analyzed is that of simultaneously minimization of makespan  $C_{\max}(\pi)$  and maximum tardiness  $T_{\max}(\pi)$ . As the measures of performance often conflict with themselves we try to find a objective a set of optimal solutions called Pareto optimal comprising all non-dominated solutions. In fact, solution  $\pi$  dominates  $\pi^*$  if and only if

$$C_{\max}(\pi) \leq C_{\max}(\pi^*) \text{ and } T_{\max}(\pi) \leq T_{\max}(\pi^*)$$

$$C_{\max}(\pi) < C_{\max}(\pi^*) \text{ and } T_{\max}(\pi) < T_{\max}(\pi^*)$$

### Metaheuristic

We will use the Non-dominated Sorting Genetic Algorithm II (NSGA-II) which is an enhanced version of the Non-dominated Sorting Genetic Algorithm (NSGA) proposed by (Deb et al., 2002) for solving non-convex and non-smooth multi objective optimization problems.

The genetic algorithms use a mechanism inspired by the evolution in order to find the solution to a given problem. Each potential solution of a problem is an individual encoding the characteristic of the problem in a way that is similar to the encoding of genetic information in the chromosomes. New solutions are produced similar to the real life by mutation (changing of the encoded information with new information) or crossover (exchange of encoded information between the individuals). The individuals are tested for fitness (how good the solution is for the given problem) and ranked accordingly to this. This ranking is used further to direct the process of generating new individuals with the goal of obtaining better and better solution with each new generation. The different methods differ in the way they encode the information, they apply mutation and crossover and they select the parents for the new generation.

### The NSGA II

- uses a fast non-dominated sorting procedure to rank all the individual in accordance with their level of non-domination;
- integrates elitism that preserves all best solutions (non-dominated) from the parents and children population.
- eliminates the difficulties of the sharing function approach by implementing the crowding distance technique with the aim to maintain diversity and the propagation of solutions.
- incorporates a crowded-comparison operator ( $<_n$ ) in order to orientate the selection at different steps of the approach.

In every iteration, new off springs are created through the classical crossover and mutation operators applied to the current population. Thereafter, the different operators of the algorithm including crowding distance, fast non-dominated sorting, and crowded comparison selection are carried out on both of parents and generated off springs. We will use the method developed by [Lebbar2017] to solve the PFSP and the BFSP scheduling optimization problem using a modified variant of NSGA 2. This method uses some specific versions of the NEH heuristic for the initialization of the populations

## Solution Encoding

The solution is presented as a vector of values ranging from 1 to the number of jobs  $n$ . This solution representation is known as a jobs permutation that indicates the order of jobs in the sequence.

## Initializing Population

The method uses a revised variant of the NEH heuristic proposed by [Nawaz1983] called RNEH for the solving of PFSP multi- objective problem and a revised variant of the NEH-WPT heuristic [Wang2010] called RNEH-WPT for the solving of BFSP multi- objective problem,

In RNEH priority is given to jobs that have a high operating time

For RNEH the algorithm is the following:

Set  $l=1$  and  $i=0$  ( $i$  refers to the number of jobs in the sequence and  $N$  to the number of jobs

Start

Step 1 Calculate for each job the sum of processing times on the  $m$  machines

Step 2 Sort the list of jobs in descending order of the total processing time

Step 3  $i=2$  extract the first 2 jobs from the sequence  $\pi$ , compute the value of  $F = 0.5 * C_{\max}(\pi) + 0.5 * T_{\max}(\pi)$  for the possible sequences of this jobs. Set the sequence with the minimal  $F$  as the current sequence  $\pi^*$

Step 4  $i = i + 1$  take the  $i$  element from  $\pi$  and insert it in every of the  $i$  position before, between and after the elements of the the current sequence  $\pi^*$ . Calculate for each such sequence the value  $F$  and set the sequence with the minimum  $F$  as the new  $P^*$

Step 5 If  $i=N$  let  $\pi_l = \pi^*$ ;  $\pi = \pi^*$ ; and go to step 6, otherwise go to step 4

Step 6  $l=l+1$ ; if  $l=4$  stop otherwise go to step 3

At the end of the algorithm we have 4 candidate solutions  $\pi_1, \pi_2, \pi_3, \pi_4$

The RNEH-WPT is similar only the step 2 is changed Step 2 the list of jobs in ascending order of the total processing time

The rest of the method (fast non-dominated sorting technique, diversity guarantee and selection procedure, genetic operators) is performed as in the standard NSGA 2 described in [Deb2002]

The algorithm is

1 initialize population of size P

1.1 apply the RNEH or RNEH\_WPT to generate 4 individuals

1.2 generate the remaining P-4 individuals randomly

2. Evaluate each individual in the population and assign a rank according to Pareto dominance sorting procedure.

3. Generate O offsprings population:

3.1 apply the binary tournament selection

3.2 apply the two point crossover operator with a probability  $P_c$  to generate offspring

3.3 use the shift change mutation operator with a probability  $P_m$

4 For  $i = 1$  to  $Gen_{max}$  do:

4.1 For each individual of the combined offspring and parents population:

4.1.1 assign rank based on Pareto dominance sorting procedure

4.1.2 determine the crowded distance of individuals in each front

4.2 select the best P solutions of the combined population using crowded comparison operator

4.3 generate the Next Generation:

4.3.1 apply the binary tournament selection

4.3.2 apply the two points crossover operator to generate offspring

#### 4.3.3 use the shift change mutation operator

### Implementing the solution in Matlab

As the candidate solution of schedule can be represented as vectors of permutation indexes and all genetical mutations and crossover operation can be expressed as operation on vectors (slicing and concatenating). So the steps in implementing the algorithm in Matlabs are the following:

- establish a format for the job parameter matrix
- write the function *cmax* that takes a job parameter matrix and a vector containing a job permutation and calculates the makespan for the given permutation
- write the function *tmax* that takes a job parameter matrix and a vector containing a job permutation and calculates the tardiness for the given permutation
- write the functions that implements the RNEH and RNEH\_WPT algorithms. The functions take as parameters a job parameter matrix and a vector containing a job permutation. Internally they use slicing to construct the partial candidates and return a 4 row matrix containing the generated first solutions
- Integrate the defined functions with a Matlab package implementing NSGA 2 of your choice available on Internet  
<https://de.mathworks.com/matlabcentral/fileexchange/10429-nsga-ii-a-multi-objective-optimization-algorithm>  
<https://de.mathworks.com/matlabcentral/fileexchange/49806-matlab-code-for-constrained-nsga-ii-dr-s-baskar-s-tamilselvi-and-p-r-varshini>  
<https://de.mathworks.com/matlabcentral/fileexchange/62487-nsga-ii-to-scheduling>  
<https://de.mathworks.com/matlabcentral/fileexchange?q=NSGA>

If no prior experience with Matlab is available the implementation can be also performed in the mlanguage of your Choice

### **Unit 8 Final assessment**

The students will receive the task of solving teams a similar task to the complex task presented as example without receiving guiding. The functioning of the system should be proved through simulation.

## References

- [Basnet1994] CHUDA BASNET & JOE H. MIZE (1994) Scheduling and control of flexible manufacturing systems: a critical review, International Journal of Computer Integrated Manufacturing, 7:6, 340-355,
- [Deb2002] Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. A. M. T. 'A fast and elitist multi objective genetic algorithm: NSGA-II'. IEEE transactions on evolutionary computation, Vol.6, No.2, (2002). 182-197.
- [Jaehne2019] Jahene, F., Pesch, E., Ablaufplanung, Springer Gabler, 2019
- [Lebbar2017] Lebbar, Ghita & Ikram, El Abbassi & El Barkany, Abdellah & Jabri, Abdelouahhab & Moumen, Darcherif. (2018). Solving the Multi Objective Flow Shop Scheduling Problems Using an Improved NSGA-II. International Journal of Operations and Quantitative Management. 39. 211-230.
- [Kaid2015] Kaid, Husam & A.M., El-Tamimi & Abouel Nasr, Emad & Al-Ahmari, Abdulrahman. (2015). Applications of Petri net based models in manufacturing systems: A review.
- [Nawaz1983] Nawaz, M., Ensore, E.E.J. and Ham, I. 'A heuristic algorithm for the m-machine, n-job flow shop sequencing problem', OMEGA – International Journal of Management Science, Vol. 11, (1983). 91–95.
- [Petri73] Petri C. A., *Concepts of Net Theory*. [MFCS 1973](#): 137-146
- [Wang2010] Wang, L., Pan, Q. K., & Tasgetiren, M. F. 'Minimizing the total flow time in a flow shop with blocking by using hybrid harmony search algorithms'. Expert Systems with Applications, Vol. 37, No. 12, (2010). 7929-7936.