

Project Title:
THE FOF-DESIGNER:
DIGITAL DESIGN SKILLS FOR FACTORIES OF THE FUTURE

Project Acronym:
DigiFoF



Grant Agreement number:
2018-2553 / 001-001

Project Nr. 601089-EPP-1-2018-1-RO-EPPKA2-KA

Subject:
D3.4 Design (modelling) tool for the Factory of the Future

Dissemination Level:
Public

Lead Organisation:
ULBS

Project Coordinator:
ULBS

Contributors:
UNIBIAL, BOC, OMiLAB

Reviewers:
OMiLAB

| Revision | Preparation date | Period covered | Project start date | Project duration |
|----------|------------------|----------------|--------------------|------------------|
| V1 | October 2020 | Month 14 -22 | 01/01/2019 | 36 Months |


This project has received funding from the European Union's EACEA Erasmus+ Program Key Action 2 - Knowledge Alliances under the Grant Agreement No 2018-2533 / 001-001 

Table of contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 3 |
| 2 | The MLMP Language | 4 |
| 2.1 | MLMP Syntax..... | 5 |
| 2.2 | MLMP Semantics | 9 |
| 3 | Conclusions..... | 17 |
| 4 | References..... | 18 |
| 5 | Annex A. List of Abbreviations..... | 19 |

1 Introduction

The flexibility of manufacturing processes is one of the major conditions for increasing the competitiveness of the factory of the future, in the current conditions of competition. The essential support for the design of flexible production lines is a combination of modelling and simulation. Modelling and simulation provide the facility to create, test, compare and optimize manufacturing processes with minimal costs, before they are physically built. Also, the models are primary artefacts for the generation of software components that coordinate and supervise the physical manufacturing process. Thus, the manufacturing process becomes an arrangement of workstations and transportation units coordinated and supervised by a computer system generated from an optimized model, which is able to adapt, with a minimum of human intervention in order to manufacture a range of products.

The efficient specification of a model, to be a primary artefact, requires a description of the properties of the system in an appropriate modelling language. Therefore, the language chosen to specify a model must facilitate the representation of this model in a clear, simple form that concentrates all the information necessary to achieve the goal. Contemporary models are characterized by a great diversity and complexity, which makes it impossible to specify them only with existing languages. Thus, there is a need to build new modelling languages, i.e. there is a need to develop domain-specific modelling languages (DSML), which would allow the efficient specification of all concepts in the field of modelling [Fowler2010]. The development of DSMLs thus becomes an integral part of the modelling process.

For a modelling language to be usable, it must be integrated into a modelling tool. The reference model of a modelling tool is, in our case, the concept of a modelling method that abstracts and integrates the components of a modelling tool.

A modelling method however is a concept [Karagiannis&Kühn2002, Bork2019] that consists of two components: (1) a modelling technique, which is divided in a modelling language and a modelling procedure, and (2) mechanisms & algorithms working on the models described by a modelling language. The modelling language contains the elements with which a model can be described. The modelling procedure describes the steps applying the modelling language to create results, i.e., models. Algorithms and mechanisms provide “*functionality to use and evaluate*” models described by a modelling language. Combining these functionalities enables the structural analysis, as well as the simulation of models.

The modelling tool that implements the concept of modelling method presented in [D3.3] we called it the Digital Production Planner Tool (DPPT) and we implemented it on the ADOxx metamodeling platform. The main component of this tool is the Modelling Language for Manufacturing Processes (MLMP), which we will present in the following.

2 The MLMP Language

We will define a Modelling Language for Manufacturing Processes (MLMP) designed specifically for specifying and optimizing manufacturing processes. MLMP is a language for modelling manufacturing processes to represent the logical and functional dependencies of the activities of a manufacturing process. The objective of the MLMP language is the conceptual integration of the functional perspective of the manufacturing processes in order to optimize them. The language is addressed to the Digital Production Planner (DPP).

The tool should support the user to find a solution to a design problem. A design problem is formulated as an assortment of products characterized by the type of material and quantity that must be produced under some time and cost constraints.

Because the most intuitive form of a model is the diagrammatic one, we chose to build a diagrammatic language.

Diagrammatic models are, generally, graphs with certain constraints on their components, in which nodes and arcs are interpreted as a concept in the field of modelling [Wolter2015, Karagiannis2016]. Designing a new visual DSML involves defining the syntax and semantics of the modelling domain. Defining the syntax of a DSML involves associating atomic concepts in the modelling domain with suggestive visual symbols and defining formal rules for composing these atomic concepts into complex concepts. Defining the semantics of modelling languages involves defining a semantic domain and mapping syntactic constructions to this semantic domain. Therefore, the syntax of a language is a way of expressing and manipulating semantics.

The modelling language will have to provide mechanisms for specifying the static dimension of a model and the modelling instrument will have to include mechanisms and algorithms for simulating the behavioural dimension of the model. Both dimensions of a model, the static dimension and the behavioural dimension, are characterized by syntax and semantics. The atomic concepts in the specific domain of MLMP language modelling are: Buffers, Workstations, Transport Machines and Ports.

In Fig. 1 we have an example of a manufacturing process model specified in terms of the MLMP language integrated in the Digital Production Planner Tool. The specified model contains: two transporters, namely, a conveyor (Conveyor1) and a manipulator (Manipulator1); a workstation (Workstation1) and six buffers and all these components are connected via material ports. The conveyor transports materials from buffer B_1 to buffer B_2 . The manipulator alternately transports materials from buffers B_3 and B_4 to buffers B_5 and B_6 . The workstation is fed from buffer B_2 , processes these materials and deposits the processing result in buffers B_3 and B_4 .

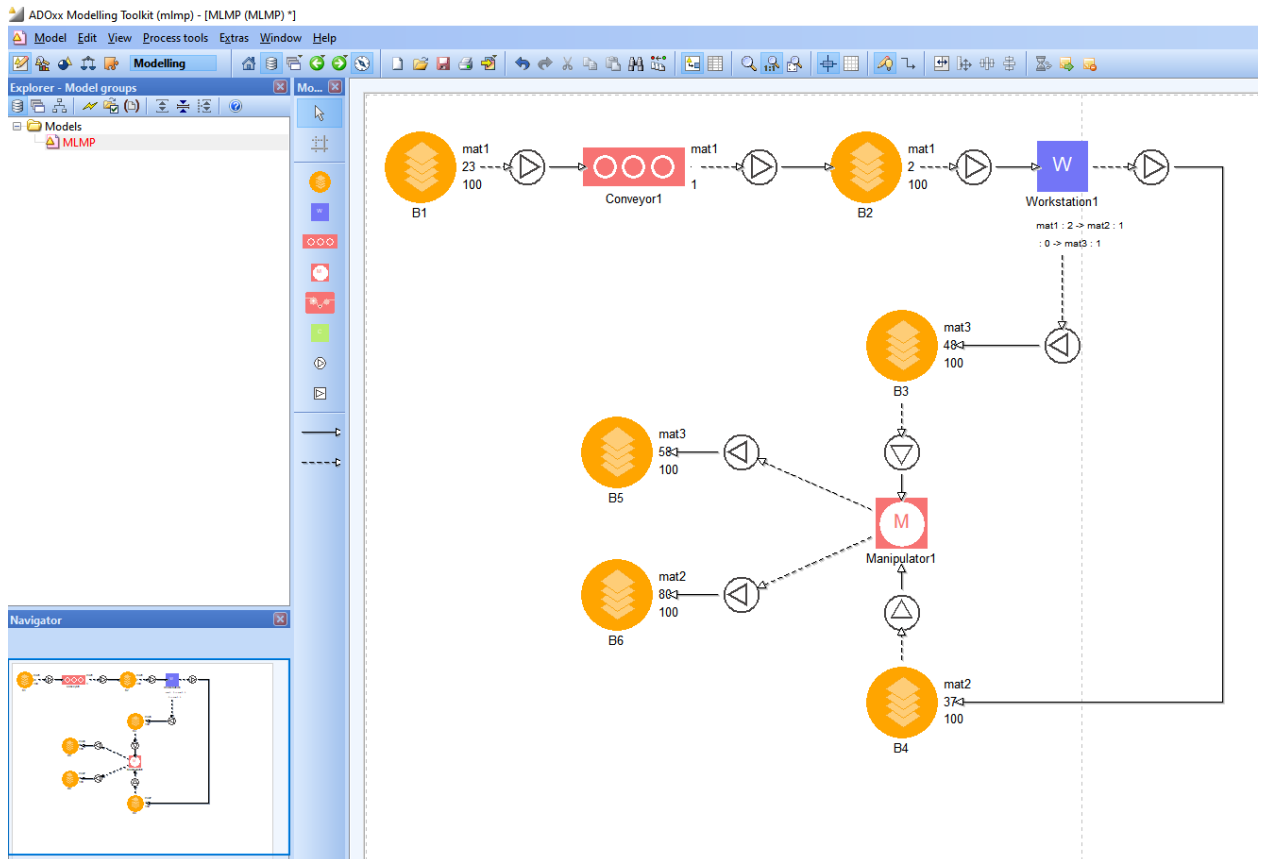


Fig.1. An example of an MLMP model

2.1 MLMP Syntax

A diagrammatic model is characterized by two dimensions, a static dimension and a behavioural dimension. Both dimensions of a model must be specified by its own syntax. In the case of the MLMP language, the behavioural dimension is specified formally, syntactically and semantically, at the metamodel level and is included in the algorithms and mechanisms component of the modelling method concept [D3.3].

Defining the syntax of the static dimension of a model involves associating suggestive notations to atomic concepts in the specific domain of modelling, which will also be the lexical atoms of the MLMP 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.

In Fig. 2 we can see the symbolic notation that we attributed to this type of component. Attribute names are: Name, MaterialType, Capacity, OccupiedCapacity.



Fig. 2. Buffer notation

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.

Workstation are components that transform materials. So, they must have at least one input and one output port of different type. 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.

In Fig. 3 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).

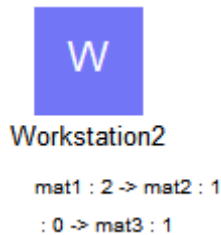


Fig.3. Workstation notation

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 behaviour 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.

In Fig. 4 we can see the symbolic notation that we attributed to the three types of conveyors, namely Fig. 4a contains the notation for the conveyor type, Fig. 4b contains the notation for the AGV type, and Fig. 4c 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).

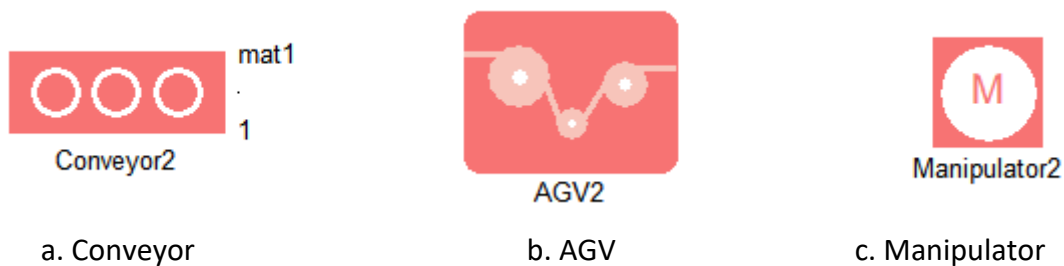


Fig.4. Transport machines notations

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. In Fig.5. we can see the symbolic notation that we attributed to this type of component. Attribute names are: MaterialKind, PortName, PortDirectionType.

MLMP diagrammatic models are graphs $G = (X, \Gamma)$, where X is a set of nodes such as Buffers, Workstations, Transport Machines and Ports and the arcs represent the flow of entities between these components. The syntax of these models imposes constraints such as: the model has to be represented by a connected graph; any two components are connected by at most one arc; between any two components there must be a port component etc. These syntactic restrictions are introduced at the metamodel level [D3.3].

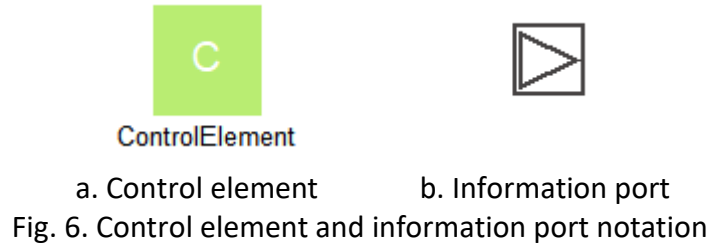


Fig.5. Material port notation

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.

In Fig. 6 we can see the notations used for the control elements (Fig. 6a) and the information ports (Fig. 6b).



MLMP is a graphical language for describing manufacturing processes at the level of manufacturing logic, easy to understand and use.

Not any graph that has the nodes made of concepts specific to a manufacturing process (workstations, transport systems, collection buffers and ports) is a correct manufacturing model. For example, the graph must be connected and may not have more than one arc between two elements, etc.

We will define an MLMP model as a graph with a set of syntactic restrictions.

A MLMP model is a directed graph $\mathcal{G} = (X, \Gamma, \sigma, \theta)$ where

X is a set of objects (concepts in our model) that represent the nodes of the graph.

Γ is a set of arcs (connections in our model).

And which satisfies the following properties:

1. \mathcal{G} is a connected graph
2. There is only one arc between any two nodes.
3. On the set of nodes X we have a partition. This means that each node of type X of the graph will represent in the Set category a distinct set of objects of the same type and these sets are disjoint two by two. If we denote the disjoint union with \sqcup then:

$$X = X_{WS} \sqcup X_{TS} \sqcup X_{BF} \sqcup X_{MP} \sqcup X_{IP};$$

where

X_{WS} is a set of workstations for the primary components;

X_{TS} is a set of transport systems for material components;

X_{BF} is a set of collection buffers for material components;

X_{MP} is a set of material ports;

X_{IP} is a set of information ports.

4. σ and θ are functions $\sigma, \theta: \Gamma \rightarrow X$ which assigns to each arc $r \in \Gamma$ the source and target objects $\sigma(r), \theta(r) \in X$. Each node of type Γ from the graph of the sketch will represent in the Set category a set of arcs between the specific concepts of the models, a set characterized by the source concept and the target concept. Therefore, Γ is a subset of the union of all the pairs of concepts that interact with each other:

$$\Gamma \subseteq (X_{WS} \times X_{MP}) \cup (X_{MP} \times X_{WS}) \cup (X_{BF} \times X_{MP}) \cup (X_{MP} \times X_{BF}) \cup (X_{TS} \times X_{MP}) \cup (X_{MP} \times X_{TS}) \\ \cup (X_{IP} \times X_{WS}) \cup (X_{IP} \times X_{TS}) \cup (X_{IP} \times X_{BF}).$$

The set Γ of arcs of a model is partitioned into disjoint subsets as follows:

$$\Gamma = \Gamma_{WSMP} \sqcup \Gamma_{MPWS} \sqcup \Gamma_{BFMP} \sqcup \Gamma_{MPBF} \cup \Gamma_{TSMP} \cup \Gamma_{MPTS} \sqcup \Gamma_{IPWS} \sqcup \Gamma_{IPTS} \sqcup \Gamma_{IPBF}$$

5. The X_{TS} set is also partitioned into disjoint subsets:

$$X_{TS} = X_{AVG} \sqcup X_{CBP} \sqcup X_{MAN} \text{ where}$$

X_{AVG} is a set of automated guided vehicles

X_{CBP} is a set of conveyors, belts, pipes

X_{MAN} is a set of manipulators

As we can see the syntactic definition of an MLMP model, introduces a series of partitions on the set of concepts and connections, as well as sub partitions on the set of transport systems. In addition, the definition includes connection constraints and a limited number of arcs between different types of nodes.

The static dimension of an MLMP model is a graph with nodes of the types described above, i.e. Buffers, Workstations, Transport Machines and Ports that are endowed with attributes specific to each type. Although the behavioural dimension of a model is dependent on the static dimension, it still reflects the running model, i.e. its transition from one static instant to another. Therefore, each instant of the static dimension represents a state of the model. The transition from one state to another will, in our approach, have to be made by a set of behavioural rules that make up the behavioural dimension of the model. We defined behavioural syntax at the metamodel level by behavioural signatures [D3.3].

2.2 MLMP Semantics

The atomic concepts described in section 2.1 are in accordance with the principles of designing flexible manufacturing systems as structure and operations are integrated from a syntactic point of view. They propose structures based on autonomous, distributed, cooperative and intelligent components, which can be assembled to perform the specific functions of manufacturing processes. The manufacturing process specification mechanism is based on the top-down functional breakdown of the system into specific components, configurable by attribute values.

The semantics of the static dimension is characterized by the values of the attributes and the graph structure of the model, and is therefore defined by mapping the attributes to data domains and the syntactically correct graph structures to known structures such as sequential structures, joins, forks, etc.

The mapping of the attributes to the data domains:

Buffers – Name:string, MaterialType:string, Capacity:integer, OccupiedCapacity:integer

In Fig.7. we can see how to customize a buffer component.

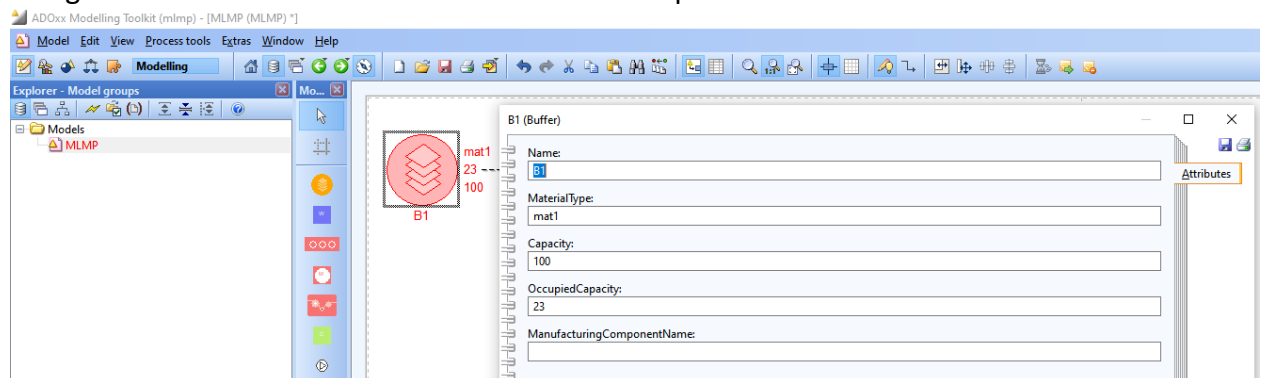


Fig. 7. Buffer customization

Workstations – Name:string, (MaterialTypeIn:string, MaterialAmountIn:integer, MaterialTypeOut:string, MaterialAmountOut:integer, Duration:time):record, OperationCode:longstring

In Fig.8. we can see how to customize a workstation component.

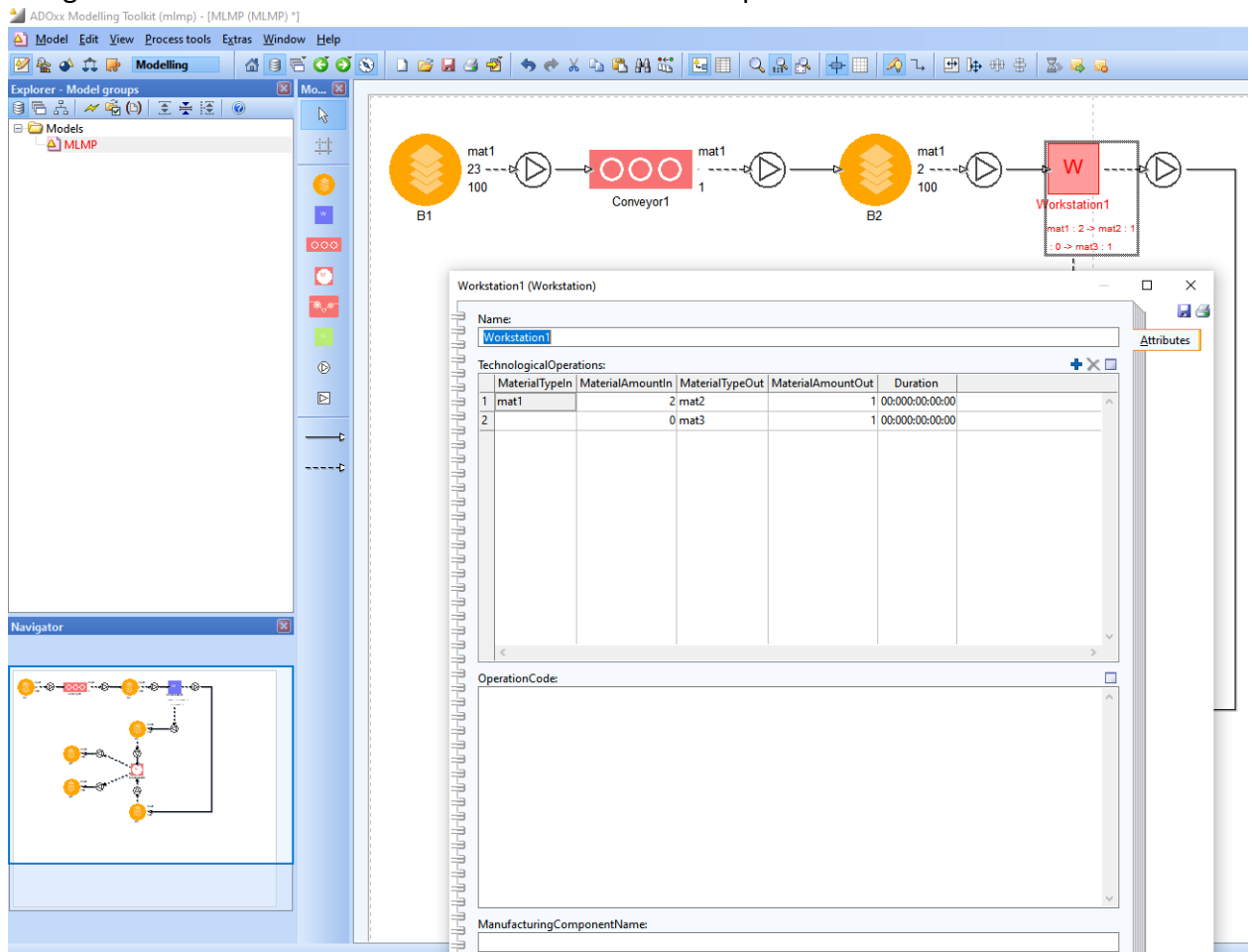


Fig.8. Workstation customization

Transport Machines - Customization is done according to the attributes defined for each type of transport machine.

Conveyors are characterized by the attributes: Name: string, (MaterialType: string, CapacityUnit: integer, Capacity: integer, TransportTime: time, OperationCode: longstring): record

In Fig. 9 we can see how to customize a conveyor component.

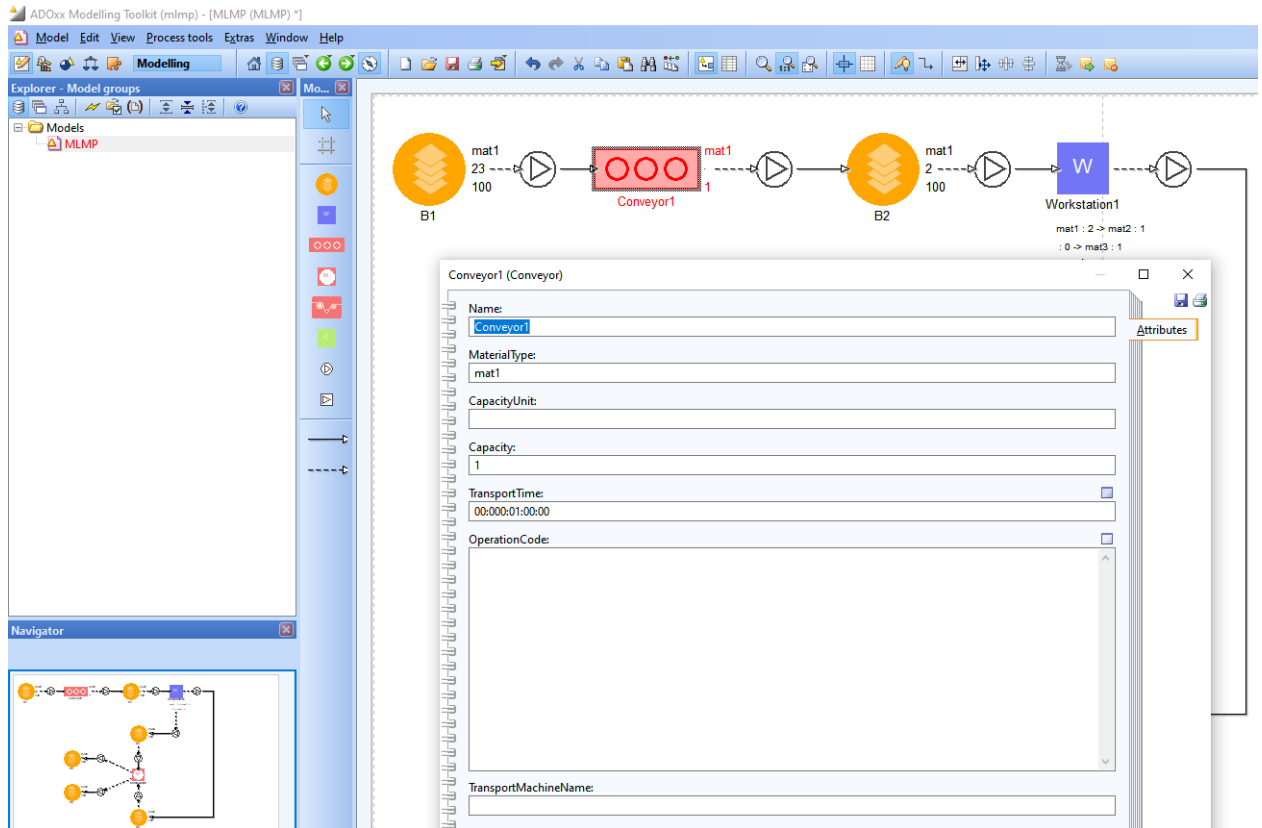


Fig.9. Conveyor customization

Transport machines of type Manipulator are characterized by the attributes: Name: string, (MaterialType: string, TransportQuantity: integer, TransportTime: time, OperationCode: longstring): record

In Fig. 10 we can see how to customize a manipulator component.

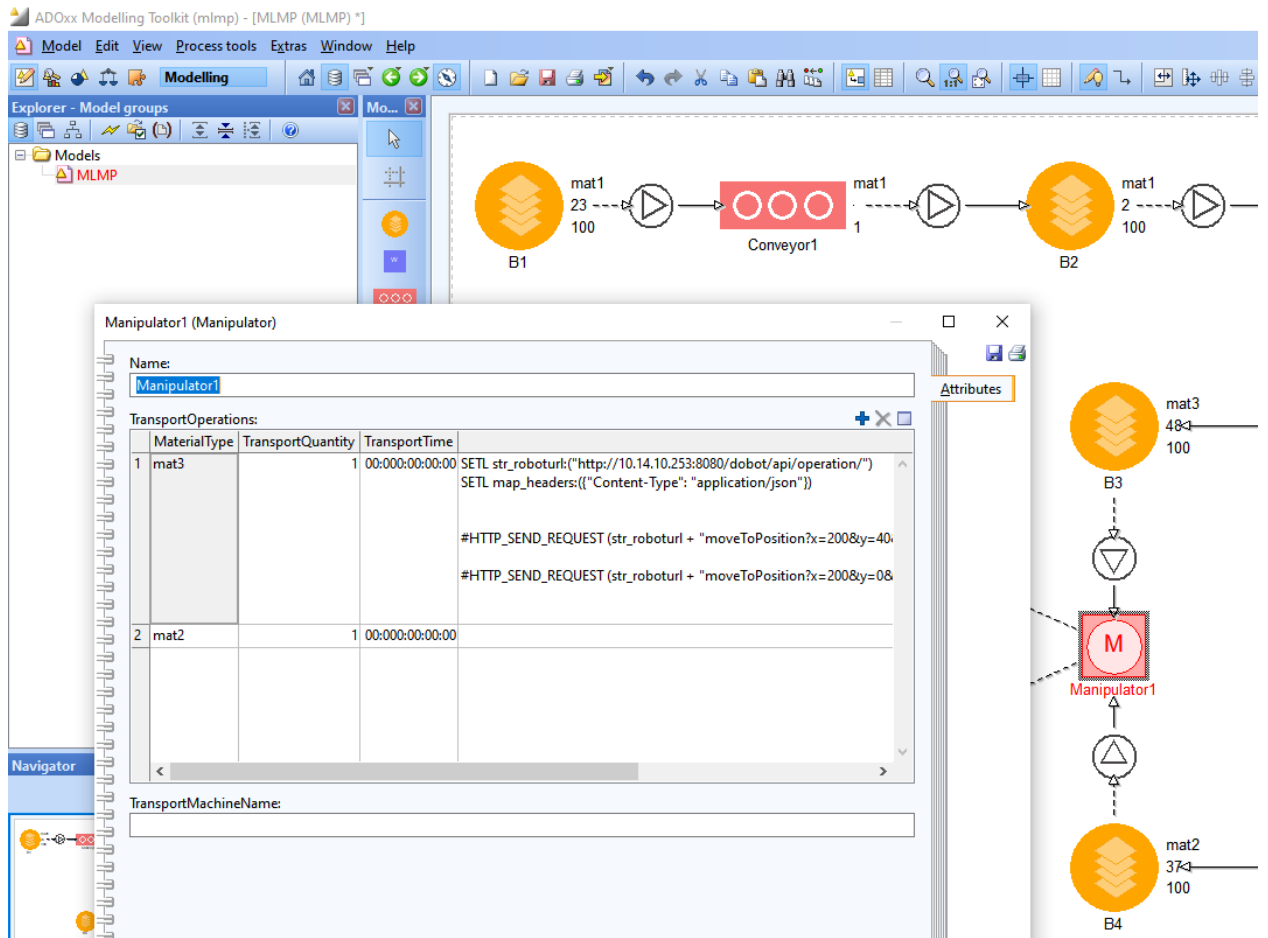


Fig.10. Manipulator customization

Transport machines of type AGV are characterized by the attributes: Name: string, (MaterialType: string, TransportQuantity: integer, TransportTime: time, OperationCode: longstring): record

In Fig.11. we can see how to customize an AGV component.

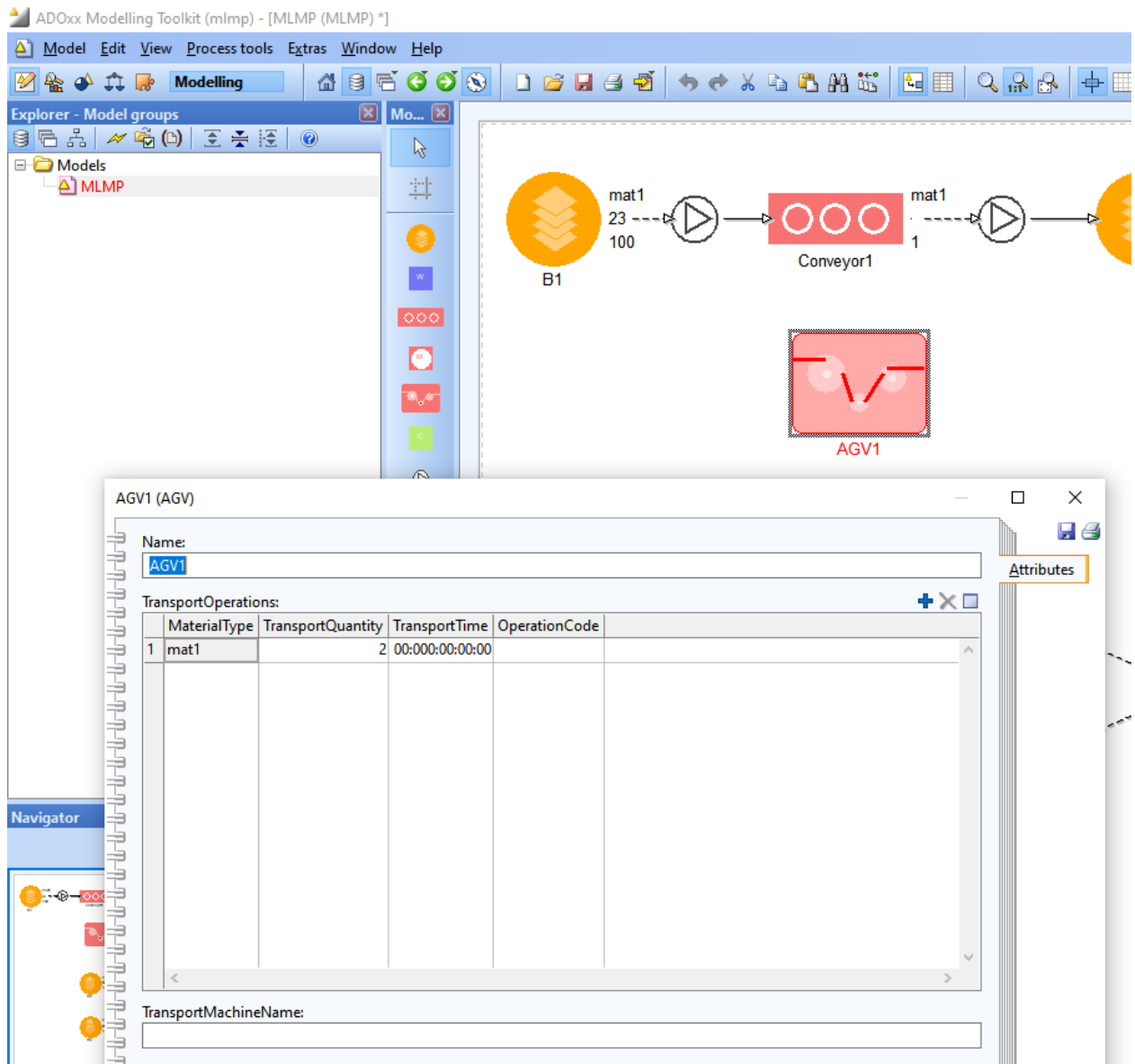


Fig. 11. AGV customization

Ports - Port type components are characterized by the attributes: Name: string, MaterialKind: string, PortName: string, PortDirectionType: enum {Incoming, Outgoing}, Direction {Right, Left, Up, Down}.

In Fig. 12 we can see how to customize a port component.

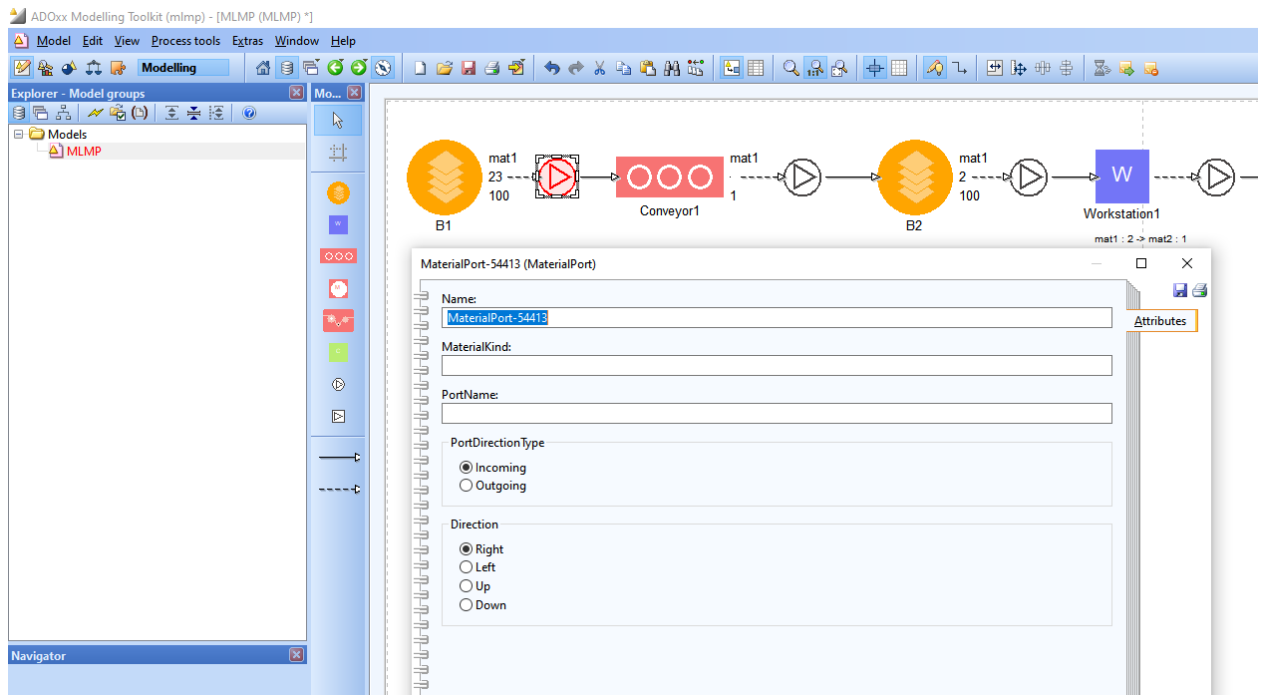


Fig.12. Port customization

The semantics of the behavioural dimension captures the behaviour of the active components of the model. In the case of the MLMP language, the semantics of the behavioural dimension was implemented at the metamodel level through two behavioural rules that simulate the behaviour of the two types of active components, workstation and transport machine.

Each workstation is fed from one or more input buffers and deposits the processing result in one or more output buffers that have limited capacities. A workstation works asynchronously if it has raw material in the input buffers and enough space in the output buffers. If one of these conditions is not met, the station stops and will start automatically when the conditions are met. The processing operation has a certain duration.

Each transport machine has a limited transport capacity and can transport several types of components in specified quantities. A transport machine works asynchronously if it has enough parts in the input buffer and also has enough space in the output buffer. If one of these conditions is not met, the conveyor stops and will start automatically when the conditions are met. The transport operation has a certain duration.

The transformation rules express local changes of the graphs and are therefore very suitable to describe the local transformations of the model states, on which the description of its behaviour is based. A graph transformation rule is a formal concept that precisely defines the model's behaviour through preconditions, postconditions and transformation steps ordered only by the causal dependence of the actions, which facilitates the application of independent rules in an arbitrary order.

The semantics of the behavioural dimension is given by the execution of the functions that implement the behavioural rules [D3.3]. Executing a behavioural rule starts with matching it in a model, continues with checking the execution conditions, then if the conditions are met the rule is applied and the operation is performed in the OperationCode field, otherwise the rollback operation is performed.

Fig. 13 highlights the step-by-step execution of an MLMP model.

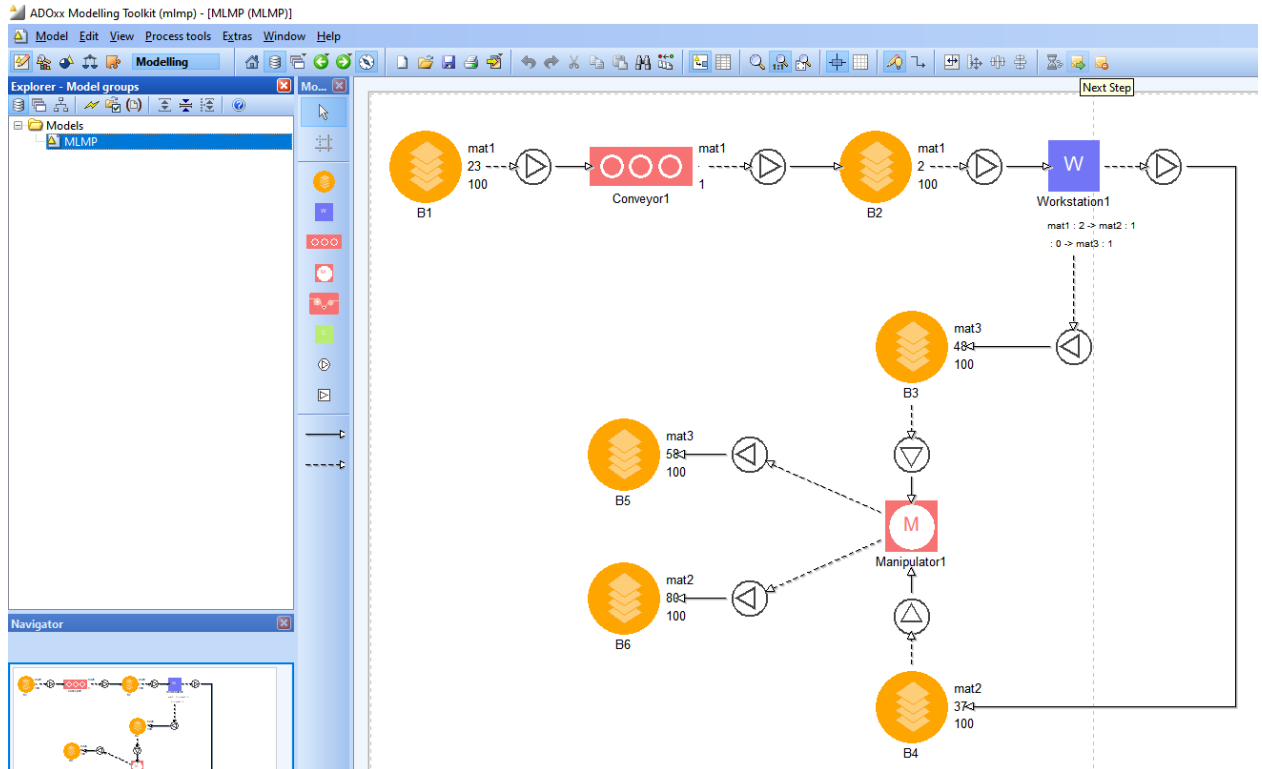


Fig.13. Step by step execution

Fig.14. highlights the execution of a step in the execution of an MLMP model. We notice that the values of the attributes change at each step according to the behavioural rules.

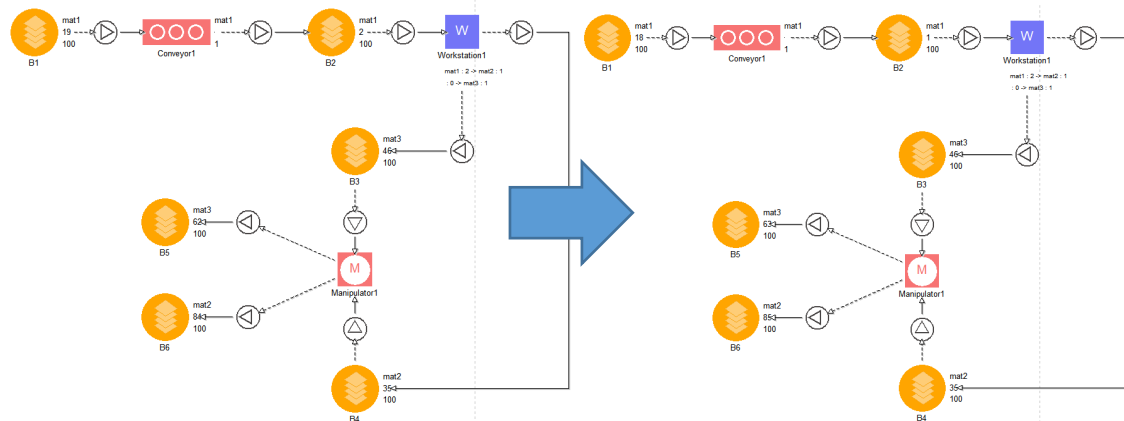


Fig.14. A step in applying behavioural rules

The dynamic behaviour of an MLMP model over time is accomplished by generic algorithms that implement the behavioural transformations. The simulation begins by initializing the system with data describing its initial state. The dynamics of the system are accomplished by the succession of the behavioural transformations executed. The semantics of an MLMP defines how process tokens are propagated through the arcs and objects of a model.

In the modelling method concept the simulation of a model is based on mechanisms and algorithms that are written in a programming language. The behaviour of the model is described by rules that specify how expressions are evaluated and

commands executed. These rules provide an operational semantic that provides a language implementation.

3 Conclusions

This paper demonstrates the feasibility of the theoretical concepts presented in [D3.3] and provides motivational arguments for the implementation of DSMLs as basic components of the modelling method concept. The concepts that represent the lexical atoms of language and the relationships between them determine the design principles of flexible manufacturing systems as structure, components and operations. The flexible manufacturing cell model becomes in this context a structure based on autonomous, distributed, cooperative and intelligent modules, able to fulfil the specific functions of the manufacturing process.

Although the developed DPPT is only an initial version with a minimal set of facilities, it highlights the advantages of such a diagrammatic DSML. A DSML, with a small set of well-chosen domain concepts, can have complex semantics to cover the modelling domain. This is due to the fact that the behavioural dimension of the models is embedded at the metamodel level. Also, this semantic load is due to the unlimited complexity of the relations between the lexical atoms of a diagrammatic language in contrast to the limited relations between the lexical atoms of the textual languages.

We could say that the transition from programming languages to domain-specific diagrammatic modelling languages is as important as the transition from programming in assembly languages to programming in high-level languages. As we can see such a language is also intuitive and easily accessible due to its visual character especially if the notations used for atomic components are well chosen. These features of a domain-specific diagrammatic modelling language make it usable in all domain-specific modelling phases due to the fact that it is accessible to all parties involved in the modelling process.

4 References

1. [Karagiannis2016] D. Karagiannis, H.C. Mayr, J. Mylopoulos, *Domain-Specific Conceptual Modeling Concepts, Methods and Tools*. Springer International Publishing Switzerland (2016)
2. [Bork2020] Dominik Bork *, Dimitris Karagiannis, Benedikt Pittl, *A survey of modeling language specification techniques*, Information Systems 87 (2020) 101425, journal homepage: www.elsevier.com/locate/is
3. [Fowler2010] M. Fowler, R. Parsons, *Domain Specific Languages*, 1st ed. Addison-Wesley Longman, Amsterdam, 2010.
4. [Bork2019] D. Bork, R.A. Buchman, D. Karagiannis, M. Lee, E.T. Miron, *An Open Platform for Modeling Method Conceptualization: The OMiLAB Digital Ecosystem*, Communications of the Association for Information Systems, forthcoming, <http://eprints.cs.univie.ac.at/5462/1/CAIS-OMiLAB-final-withFront.pdf> (2019)
5. [Craciunean2018] D.C. Crăciunean, D. Karagiannis, *Categorical Modeling Method of Intelligent Workflow*. In: Groza A., Prasath R. (eds) Mining Intelligence and Knowledge Exploration. MIKE Lecture Notes in Computer Science, vol 11308. Springer, Cham (2018).
6. [Craciunean2019] D.C. Crăciunean, *Categorical Grammars for Processes Modeling*, International Journal of Advanced Computer Science and Applications(IJACSA), 10(1), (2019)
7. [Wolter2015] Uwe Wolter, Zinovy Diskin, *The Next Hundred Diagrammatic Specification Techniques, A Gentle Introduction to Generalized Sketches*, 02 September 2015 : <https://www.researchgate.net/publication/253963677>,
8. [Plump2019] D. Plump, 'Computing by graph transformation: 2018/19', Department of Computer Science, University of York, UK, Lecture Slides, 2019.
9. [Campbell2019] G. Campbell, B. Courtehoue and D. Plump, 'Linear-time graph algorithms in GP2', Department of Computer Science, University of York, UK, Submitted for publication, 2019. [Online]. Available: <https://cdn.gjcampbell.co.uk/2019/Linear-Time-GP2-Preprint.pdf>.
10. [Campbell2018] G. Campbell, 'Algebraic graph transformation: A crash course', Department of Computer Science, University of York, UK, Tech. Rep., 2018. [Online]. Available: <https://cdn.gjcampbell.co.uk/2018/Graph-Transformation.pdf>.
11. [Plump2010] D. Plump, 'Checking graph-transformation systems for confluence', ECEASST, vol. 26, 2010. DOI: 10.14279/tuj.eceasst.26.367.
12. [Ehrig2015] Hartmut Ehrig, Claudia Ermel, Ulrike Golas, Frank Hermann, *Graph and Model Transformation General Framework and Applications*, Springer-Verlag Berlin Heidelberg 2015
13. [Milner2009] R. Milner, *The Space and Motion of Communicating Agents*, Cambridge University Press, (2009)
14. [D3.3] Design method for the Factory of the Future

5 Annex A. List of Abbreviations

| | |
|-------------|---|
| AGV | Automated Guided Vehicle |
| DSML | Domain Specific Modeling Language |
| MLMP | Modeling Language for Manufacturing Processes |
| DPP | Digital Production Planner |
| DPPT | Digital Production Planner Tool |

6 Annex B. List of Figures

| | |
|--|----|
| Fig.1. An example of an MLMP model | 5 |
| Fig.2. Buffer notation | 6 |
| Fig.3. Workstation notation | 6 |
| Fig.4. Transport machines notations | 7 |
| Fig.5. Material port notation | 7 |
| Fig.6. Control element and information port notation | 8 |
| Fig.7. Buffer customization | 9 |
| Fig.8. Workstation customization | 10 |
| Fig.9. Conveyor customization | 11 |
| Fig.10. Manipulator customization | 12 |
| Fig.11. AGV customization | 13 |
| Fig.12. Port customization | 14 |
| Fig.13. Step by step execution | 15 |
| Fig.14. A step in applying behavioural rules | 15 |