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:
D.3.3 Design Method for the Factory of the Future[1]

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	February 2020	Month 4 -14	01/01/2019	36 Months

---

[1] "Any communication or publication related to the action, made by the beneficiaries jointly or individually in any form and using any means, shall indicate that it reflects only the author's view and that the Agency and the Commission are not responsible for any use that may be made of the information it contains."

Table of contents

1 Introduction

The complexity of manufacturing processes is steadily increasing. This complexity makes it impossible for manufacturing process supervision to be mastered by classical methods. Computerization is thus involved in the supervision and management of all organizational activities of companies. Computerization involves choosing and using an available tool for modelling, simulation and process analysis.

The variety of modeling processes, often, makes it necessary to implement specific modelling tools. The first phase of this process is the specification and implementation of the Modelling Method over a metamodeling platform.

The language specific to a Modelling Method [Karagiannis2002] relies heavily on graphic elements. In order to represent the properties of the models in the form of specifications, it is important to build the most appropriate language to enable these specifications to be written in a very clear and simple form.

A category [Barr2012] as well as a model is a mixture of graphical information and algebraic operations. Therefore, category language seems to be the most general to describe the models [Milner2009]. It can provide us with the features that must characterize both the Domain Specific Language (DSL) and the Modelling Method concept.

The theory of categories works with patterns or forms in which each of these forms describe different aspects of the real world. Category theory offers both, a language, and a lot of conceptual tools to efficiently handle models.

An important aspect of manufacturing process models is building complex functions from a given set of simple functions, using different operations on functions such as composition and repeat composition. Category theory is exactly the right algebra for such constructions.

Since category theory is an abstract algebra of functions, we can consider categories that are purely formal and do not necessarily consist of functions in the sense of the mathematical definition. These are the syntactic part of the models. Programs and languages are formal constructions that are intended to describe or specify formal functions. The category theory is well adapted to deal with the relationship between syntax and semantics.

Many system (process) properties can be unified and simplified through an arrow diagram presentation (graphs). The category theory provides the ideal framework for such constructions.

What characterizes a modelling language is primarily expressivity, but the language of categories is one of the most expressive languages of mathematics.

We will specify both the concept of modelling method and that of multi-level modelling in the language of the category. This approach provides us with information about the facilities that a metamodeling language must provide in order to be able to specify such models.

To specify the syntax of a graphical metamodel we will use a categorical sketch, which in turn is represented in a graphical language.

2 Formalizing the concept of modelling method

We understand a manufacturing process as a behavior model of a dynamic system at a certain level of abstraction. While a sequential system performs a single step at a time and can therefore be characterized by a single current state, the different components of a concurrent system may be in different local states at a time, which together make up the global state of the system at a time.

The behavior of the system is given by several processes that are executed simultaneously (parallel and distributed), where these processes exchange data to influence each other's evolution.

A process is a sequence of steps that define behavior. There are several approaches to the notion of step, which leads to as many different types of behavior.

Labeled transition systems are the most commonly used models for competing process semantics, and are essential for process algebras, where processes typically receive a semantic as labeled transition systems.

In this sense, one step is a triplet (s_1, α, s_2) where α is a label of an action, which is an element of a certain set of actions, and s_1 and s_2 are states from a set of states.

Thus, a manufacturing process can be defined as a graph P, whose nodes are the states in which the system can be at a given moment of operation, and the arcs of the graph P have labels representing actions that can be executed by the simulated system. An execution of process P is described by a movement along the arcs of graph P from one state to the other. Execution starts from a distinct state called initial state.

The theoretical mechanism most used for modeling processes is the transition system. Transition systems are mechanisms with a well-defined syntax and semantics, but become impossible to use in competing systems. For this reason, many other higher-level languages such as Petri Nets, Business Process Model and Notation (BPMN), Event-driven Process Chain (EPC), Unified Modeling Language (UML), etc. are used in practice. These models describe the processes in terms of activities ordered through casual dependencies [Craciunean2018] [Craciunean2019]. A mechanism that can be the base of such a metamodel is the concept of modeling method.

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

Essentially, a modeling method relies heavily on the graphical notation. The sketch L^1 that we will define in this section is generating the basic concepts for describing the models and the appropriate graphical notation. The modeling procedure describes the steps to be followed in applying the modeling language to create results, i.e. patterns.

Algorithms and mechanisms provide functionality for the use and evaluation of models described by a modeling language. This functionality is given by the processes that act on models to change their state. Combining these functionalities allows structural analysis as well as the simulation of the models.

The essential part of a Modeling Method is the modeling language. In our approach, this language is a Domain-Specific Language (DSL).

A DSL is a programming language that offers increased facilities for application development in a domain [Fowler2010] [Karagiannis2016]. The point is that the concepts and notation of such language are as close as possible to what we have in mind when we are thinking about a solution in this area. A Domain-Specific Modeling Language (DSML) is a DSL adapted to specify models.

In the Model Driven Engineering (MDE) conception, the development of DSMLs is an integral part of the software modeling process. These DSMLs are, in general, graphic languages adapted for specifying the models in a domain. Designing a new DSML involves first interpreting the nodes and edges of a graph and then imposing domain-specific constraints and assigning suggestive visual symbols to represent the concepts involved in the models specific to the language in question.

3 Defining the Modeling Language for Manufacturing Processes

The following sections describes the concepts and the design consideration for the Categorical Modeling Method proposed namely the Modeling Language for Manufacturing Processes (MLMP). They are a generalization of those presented in [Mironescu2019].
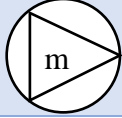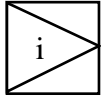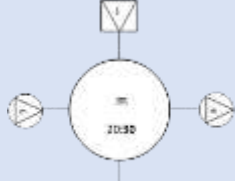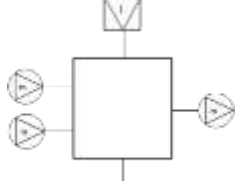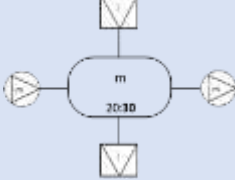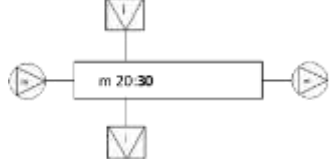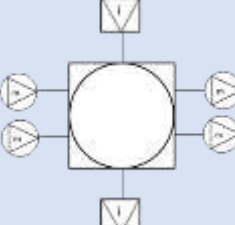
3.1 Static aspects

This part describes the main language concepts. The symbols used are given in Table 1. The graphical representations of the language concepts are presented in Table 2.

Table 1. Symbols used for the mathematical description of the modeling language concepts

Symbol	Meaning
AGV	autonomous guided vehicles
AGVs	vehicle-type transport equipment set
ALG	the control algorithm for which several representation options can be considered
BF	buffer
BFs	buffers set
c	capacity (maximal content)
cc	current content
ct	current quantity transported
com	command from the accepted set COM
COM	set of orders accepted by machines (symbols accepted on the information input port from the superior level)
CBP	conveyors, belts, pipes
CBPs	conveyor, belt, pipe type transport equipment set
ec	command and control element
EC	control elements set
fin	function that associates the input of each element with the corresponding buffer port
fout	function that associates the output port of each element with the corresponding buffer
FMS	flexible manufacturing system
i	information
I	set of all possible types of information
IP	information port
idin	start buffer
idout	stop buffer
lop	list of technological operations that the machine can perform
lopt	list of transport operations
lpii	list of information in-port
lpio	list of information in-port
lpmi	list of material in-port
lpmo	list of material out-port

m	type of material
M	set of all possible types of materials
MA	machine
MAN	manipulator
MANs	manipulator type transport equipment set
MI	list of input material-quantity pairs
MO	list of output material-quantity pairs
MP	material port
OP	set of all possible technological operations
opt	transport operation
OPT	set of all possible transport operations
p	port
pi	input port from which the manipulator takes the material
po	output port where the manipulator put the material
P	set of ports
PIN	reunion of all input ports sets of all elements
POUT	reunion of all output ports sets of all elements
PMI	set of materials input ports
PMO	set of materials output ports
PII	set of information input ports
PIO	set of information output ports
s	sense
S	set of the two possible types of senses (*in* or *out*)
t	type of port
ta	action duration
tt	transport time
T	set of all possible types of ports
TC	set of material-quantity pair
TS	transport system
WS	workstation
WSs	workstation set
X	set of objects (concepts) representing the graph nodes
X$_{AGV}$	set of Autonomous Guided Vehicles
X$_{CBP}$	set of conveyors, belts, pipes
X$_{MAN}$	set of manipulators
X$_{BF}$	set of collection buffers for material components
X$_{IP}$	set of information ports
X$_{MP}$	set of material ports
X$_{TS}$	set of transport systems for material components
X$_{WS}$	set of workstations for the primary components

Table 2. Graphical representation of the modeling language concepts

Graphical representation	Meaning
	Material type port
	Information type port
	Buffer
	Workstation
	Autonomous guided vehicles
	Conveyors, belts, pipes
	Manipulator
	Control element

3.1.1 Ports

Every port p has a type and a direction. There are general classes, depending on what is transmitted through the port. In this approach, we will use *M* and *I* as such classes. It would be straightforward to extend the system to other types, too (e.g. Energy). Within each class, labels can be declared that differentiate ports types:

- The material and information type label: $m \in M$, $i \in I$;
- The labels: $T = M \cup I$;
- The directions: $S = \{in, out\}$;
- The ports $P = \{(t, s) \mid t \in T, s \in S\}$, $P \subset T \times S$.

Every port has an *in* or *out* direction.
Graphical representation of the modeling language concepts are given in Figure 1. The connection between the entities is represented as in Figure 2 and has the port type and the sense on it.

Figure 1. Material type port (a) and information type port (b). The triangle shows the direction – the edge is pointed to the exterior (*out* port) or interior (*in* port) of the entity that contains it. The label is written inside.
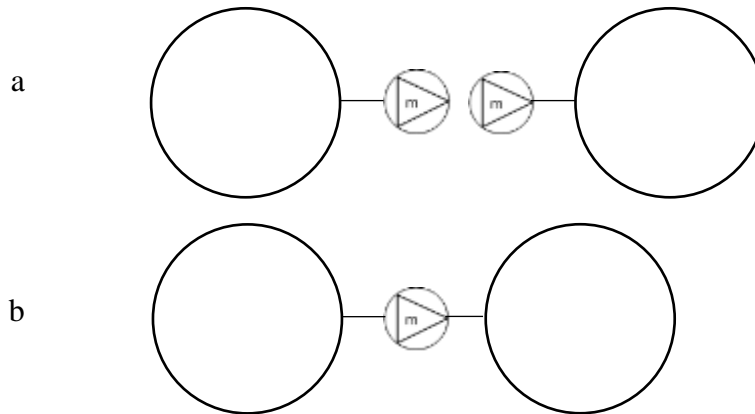

Figure 2. Connection representation. a) disconnected; b) connected.

Important sets:

- PMI is a set of materials input ports:

$$PMI = \{(m,"in") \mid m \in M\} \ \ PMI \subset M \times \{„in"\}$$

- PMO is a set of materials output ports:

$$PMO = \{(m,"out") \mid m \in M\} \ \ PMO \subset M \times \{„out"\}$$

- PII is a set of information input ports:

$$PII = \{(i,"in") \mid i \in I\} \ \ PII \subset I \times \{„in"\}$$

- PIO is a set of information output ports:

$$PIO = \{(i, "out") \mid i \in I\} \quad PIO \subset I \text{ x } \{„out"\}$$

3.1.2 Buffers

The buffers have: a type; a current content, cc = number of units (variable attribute); a capacity, c (constant attribute, representing the maximal content).
The buffers have an input port and an output port for each material.
The mathematical description of a buffer is:

$$B = \{(m, cc, c, (m, "in"), (m, "out"), (i, "in"), (i, "out")) \mid t \in T, cc \in \mathbb{N}, c \in \mathbb{N}\}$$

A buffer is an element of the Cartesian product:

$$B \subset T \text{ x } \mathbb{N} \text{ x } \mathbb{N} \text{ x } PMI \text{ x } PMO \text{ xPII x } PIO$$

Figure 3 presents a buffer as circle with input and output ports for material and information. In the center the type, current quantity and maximum quantity of material contained are displayed.
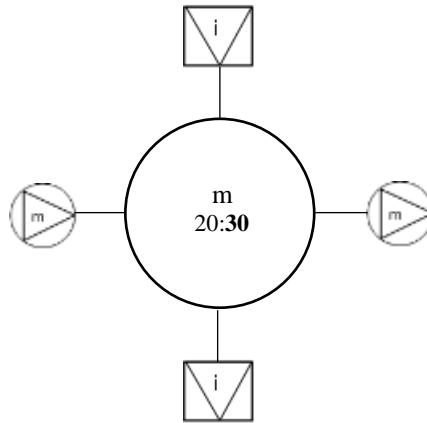

Figure 3. Buffer representation. In this particular case, the type is m, current content is 20 and capacity is 30.

3.1.3 Machines

Every machine has associated:
- at least one in-port for information on which commands or signals are transferred;
- at least one out-port for information on which the machine sends responses or commands;
- at least one in-port for materials;
- at least one out-port for materials.

Therefore, a definition of the abstract type from which all machines are derived, is:

$$MA = (lpii, lpio, lpmi, lpmo)$$

where:

- lpii is the list of information in-port: $lpii \subset P(PII)\backslash\varnothing$;
- lpio is the list of information in-port: $lpio \subset P(PIO)\backslash\varnothing$;
- lpmi is the list of material in-port: $lpmi \subset P(PMI)\backslash\varnothing$;
- lpmo is the list of material out-port: $lpmo \subset P(PMO)\backslash\varnothing$;
  In these expressions, $P(A)$ is the set of sub-sets of set A (where A is one of the port sets: PII, PIO, PMI, PMO). $P(A)\backslash\varnothing$ expresses the condition that the lists should have at least one element.

3.1.3.1 Workstations

Every workstation (WS) has a list of technological operations (product capabilities). Every technological operation contains: the type and the number of materials taken from the feed buffers (MI); the number and type of materials that will be deposit in the output buffers (MO); the duration of the operation (ta). This can be expressed mathematically as following:

- ▪ The tuple (com, MI, MO, ta) is an tehnological operation, where:
- - com is a command from the accepted set COM: com∈COM ;
- - COM is the set of orders accepted by machines (symbols accepted on the information input port from the superior level);
- - MI is a list of input material-quantity pairs: MI ⊂ P(TC);
- - MO is a list of output material-quantity pairs: MO ⊂ P(TC);
- - TC is the set of material-quantity pair:

$$TC = M \times N$$

  where N is the natural number set;

- - ta is the action duration: ta∈N.

OP is the set of all possible technological operations.
lop is a list of technological operations that the machine can perform: lop⊂ P(OP).
The workstation WS = (lpii, lpio, lpmi, lpmo, lop)
The types of materials defined in each operation, in MI and MO must correspond to the types of ports in lpii, lpio, lpmi, lpmo.
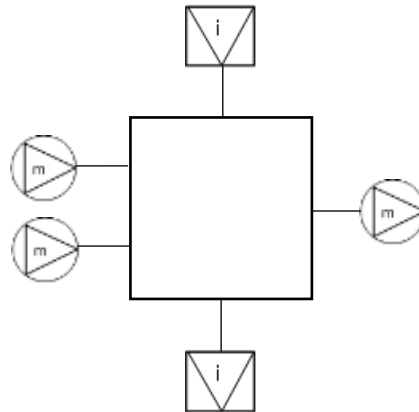The graphical representation (Figure 4) is a square.

Figure 4. Workstation representation

3.1.3.2 Transport machines

Every transport system (TS) has the same *in/out* port type and has a transport capacity (possibly variable in certain limits).

3.1.3.2.1 Autonomous Guided Vehicles (AGV)

Vehicles can move and are not permanently attached to a specific buffer. They have a single input port and an output port with which they can be attached to the corresponding ports of the same type buffers. They have associated: a material type; a current content, cc; a capacity, c; a list of transport operations, lopt.

As a result, a transport machine of this category is:

AGV=((i1, *in*),(i2, *out*),(t, *in*),(t, *out*), t, c, cc, lopt)

A transport operation (abbreviated opt) is defined by: the start and stop buffers indexes (idin, idout); current quantity transported, ct; transport time between the buffers, tt. The mathematical formula is:

opt =(idin, idout, ct, tt), with ct≤c

OPT represents all possible transport operations and lopt is a list of transport operations. Their relation is:

lopt⊂ *P*(OPT)

The route can be defined as a series of segments with nodes between them. This allows investigating situations in which AGVs may intersect.

The graphical representation of AGV (Figure 5) is a rectangle with round edges, together with the corresponding ports.
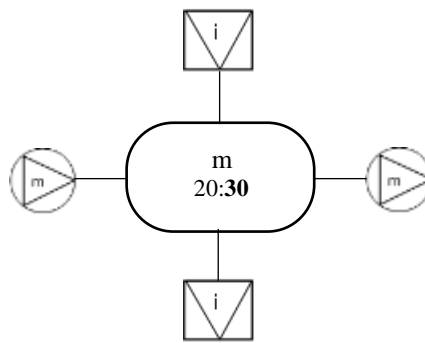

Figure 5. AGV representation

3.1.3.2.2 Conveyors, Belts, Pipes (CBP)

Conveyors, belts, pipes have associated: a material type; a capacity, c; a transport time, tt:

CBP = ((i1, *in*), (i2, *out*), lpmi, lpmo, c, tt)

The graphical representation (Figure 6) is a long and narrow rectangle with the corresponding ports.
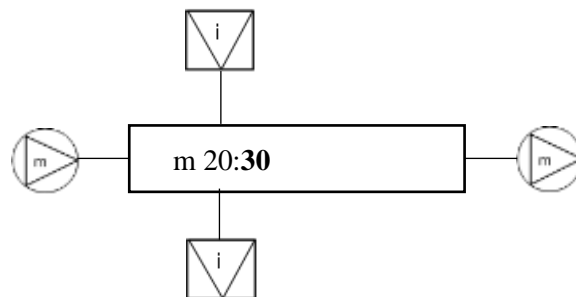

Figure 6. Conveyors, belts, pipes representation

3.1.3.2.3 Manipulators

MAN = ((i1, *in*), (i2,*out*), lpmi, lpmo, lopt)

The transport operation opt = (pi, po, ct, tt).

The input port, pi, from which the manipulator takes the material m is:

pi = (m, *in*)

The output port, po, where the manipulator put the material m is:

po=(m, *out*)

ct is the current quantity transported, tt is the transport time between two ports.

OPT represents all possible transport operations.

lopt$\subset P$(OPT)

The graphical representation (Figure 7) is a circle inscribed in a square (to suggest a turntable transfer machine), with the corresponding ports.
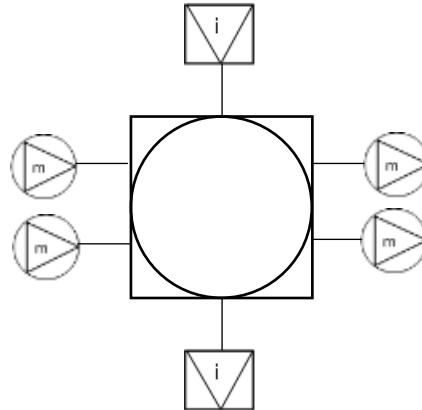

Figure 7. Manipulator representation

3.1.4 Command and control elements

The command and control elements, ec, have only *in/out* information ports.

$$ec = (lpii, lpio, ALG)$$

ALG is the control algorithm for which several representation options can be considered (state diagram, Petri net)
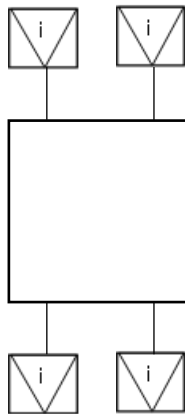
Figure 8. Control element representation

3.1.5 Flexible Manufacturing System

3.1.5.1 Model

A model of a flexible manufacturing system (FMS) is a structure:

$$M = (WSs, AVGs, CBPs, MANs, EC, BF, fin, fout)$$

where:

- – WSs is the workstation set;
- – AGVs is the vehicle-type transport equipment set (Automated Guided Vehicles);
- – CBPs is the conveyor, belt, pipe type transport equipment set;
- – MANs is the manipulator type transport equipment set;

- – EC is the control elements set;
- – BFs is the buffers set;
- – fin:PIN→BF is the function that associates the input of each element with the corresponding buffer port;
- – fout:POUT→BF is the function that associates the output port of each element with the corresponding buffer;
- – PIN is the reunion of all input ports sets of all elements;
- – POUT is the reunion of all output ports sets of all elements.

The type of port on the machine and the buffer must match.
Also, the machine operation sequences will have to correspond to the current configuration (the buffers to which they are connected to).

3.1.5.2 Example

For example, we present the model of an installation that assembles the subassemblies produced by two production lines. For simplicity, the command elements were not represented. The graphical representation of the system is depicted in figure 9.

Line 1
The raw material is temporarily stored in the B1 bunker. From here, it is transported with AGV1. In figure 9, it is docked to B1. As long as it is docked and not full, AGV1 is loaded by itself from the B1 bunker through the m1 port.
AGV1 runs between the bunker B1 and the bunker B7. Bunker B3 supplies the WS1 workstation through the m1 port.
The station is automatically loaded from the B1 hopper as long as there are parts/material m1 through the m1 type port. The processed parts / m3 intermediate material are unloaded through the m3 port into the B4 bunker.
From here, the m3 material type is taken through the m3 type port by the BT1 transporting belt and placed in the B5 bunker through the m3 type port. The B5 bunker supplies the WS2 workstation through the m3 port. Station WS2 unloads m4 material through m4 port in bunker B6.

Line 2
The raw material is temporarily stored in the bunker B2. From here, it is transported with AGV2. AGV2 runs between the bunker B1 and the bunker B3. In the figure he is docked at B3. As long as it is docked and not empty, and the bunker B3 is not full AGV2 is unloaded by itself in the bunker B3 through the m2 port.
Bunker B3 supplies the WS2 workstation through the m2 port. The WS2 station produces m5 type material which it unloads in the B8 bunker.

Assembly
The M1 manipulator alternatively takes material through the m4 and m5 type ports from the 2 production lines from the B6 and B8 bunkers. M1 transfers materials to B9 and B10 bunkers according to their types. Bunkers B9 and B10 feed the WS4 station. The WS4 station combines the two material types and produces the finished m6 material that it transfers to the B11 bunker.
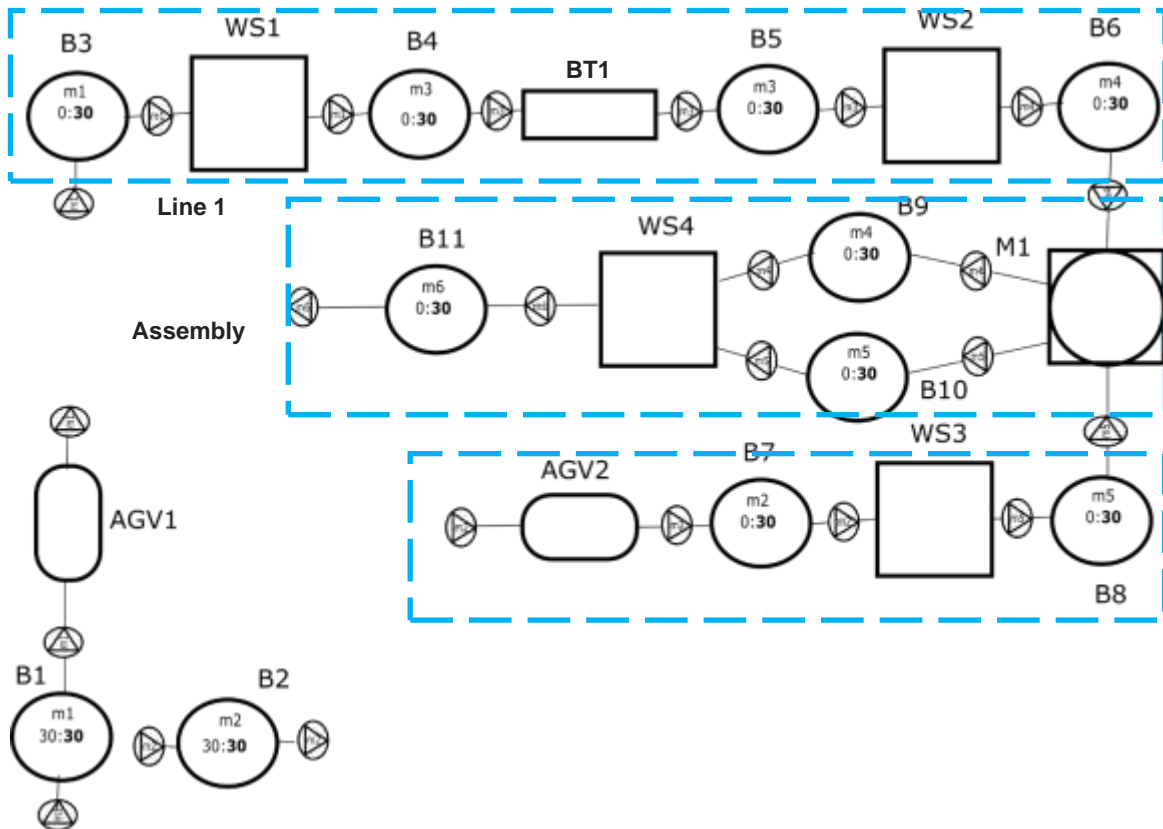
Figure 9. Example of FMS representation

3.2 Dynamic Aspect

This part describes the behavior associated with the concepts. The symbols used here are the same as those used in chapter 3.1, summarized in Table 1.

3.2.1 Buffers

The buffers (Figure 10) have three states: empty ($cc=0$), partially loaded ($0<cc<c$), full ($cc=c$). The buffer responds to 2 commands on the information *in* port: „load" and „unload".

If the buffer is partially loaded, „load" increments cc, „unload" decrements cc. On the out port is issued „cc". If the buffer is empty and receives "unload", it does not increment anything, just send "empty" on the information out port. If the buffer is full and receives "load", nothing is loaded, just "full" is send on the information out port.
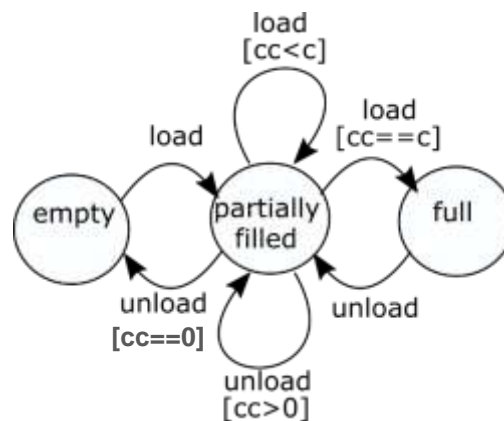

Figure 10. Behavior of the buffer

Each load/unload operation has a duration. This can be the default and the same for all items of this type for simplicity, or it can be entered as an additional attribute.

3.2.2 Machines

Machines take materials through ports from buffers to which they are connected to ports and deposit materials through the output ports in 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 example below (Figure 11), a general machine with a single input buffer and an output buffer is shown. Operation op1 requires n units in the input buffer and produces m units in the output buffer. When the op1 command appears on the input port, the machine tries to load from the input field n units – going into standby state when the buffer is completely emptied and resuming the load cycle when it is filled. Every step of the load cycles the machine sends the "unload" command to the buffer and reads the cc attribute of the buffer. After the machine is loaded with the units, the machine enters the processing state which has an associated duration. After the processing time is completed, the machine tries to unload in the output log m units – going into standby condition when the buffer fills completely and resuming the discharge cycle when emptying. Every step of the load cycles the machine sends the "load" command to the buffer and reads the cc attribute of the buffer. After the m units are loaded in the buffer, the machine enters the inactivity state (Idle) in which it can receive the following command.
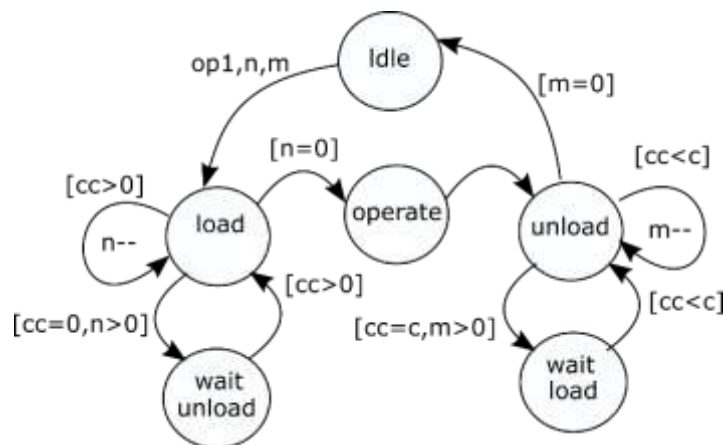

Figure 11. Behavior of a machine. n is the number of units to be unloaded from the input buffer; at each unload step, n is decremented (n--). m is the number of units to be loaded in the output buffer; at each load step, m is decremented (m--).

3.2.2.1 Workstations

The workstations act according to the current operation (op). The particularity is that a machine can have multiple input buffers as a result several load processes may occur, that are carried out in parallel. Only after all materials are loaded into the corresponding quantities shall be passed into the processing phase. In the bottom representation (Figure 12) we combined the notation from UML activity diagram (join fork on successive lanes) for a machine with two input buffers and one output. Also, flexible machines can be (in the static representation) linked to several input buffers but for a particular operation they take materials only from some of them.
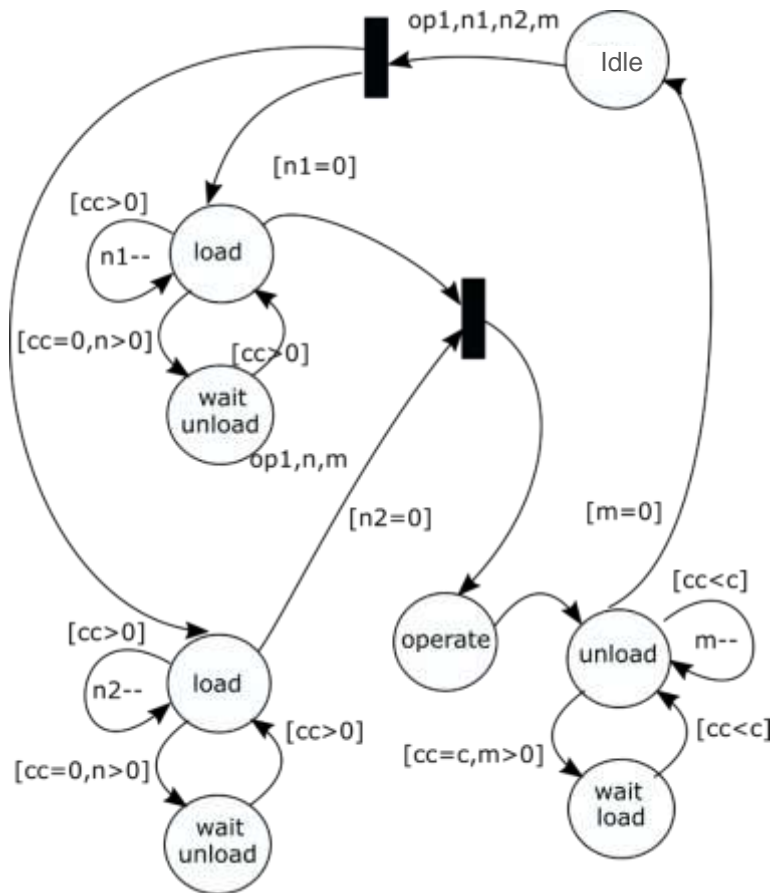
Figure 12. Behavioral state machine of the workstation

3.2.2.2 Transport machines

3.2.2.2.1 *Autonomous guided vehicles (AGV)*

The peculiarity of AGV is that they have:
- single input and output buffer
- the same amount taken from the departure buffer and unloaded in the arrival buffer.
As an attribute and position that means, the processing can be divided into at least two states (if only the starting and end positions) are considered **undocking** and **docking**. If more than one position is considered, each of these will be a state. The transition between them will be made on the basis of an order coming to the in-information port to allow control and synchronization of multiple AGPs. In the picture we have an AGV that pendas between buffers p1 and p2. It is docked for charging (dock l). After loading, it depends from p1 and when it receives the transfer command, it is docked for unload to p2 (dock u). After it is emptied it depends from p2 and when it receives the return command it retransfers to p1 where it is docked for loading.
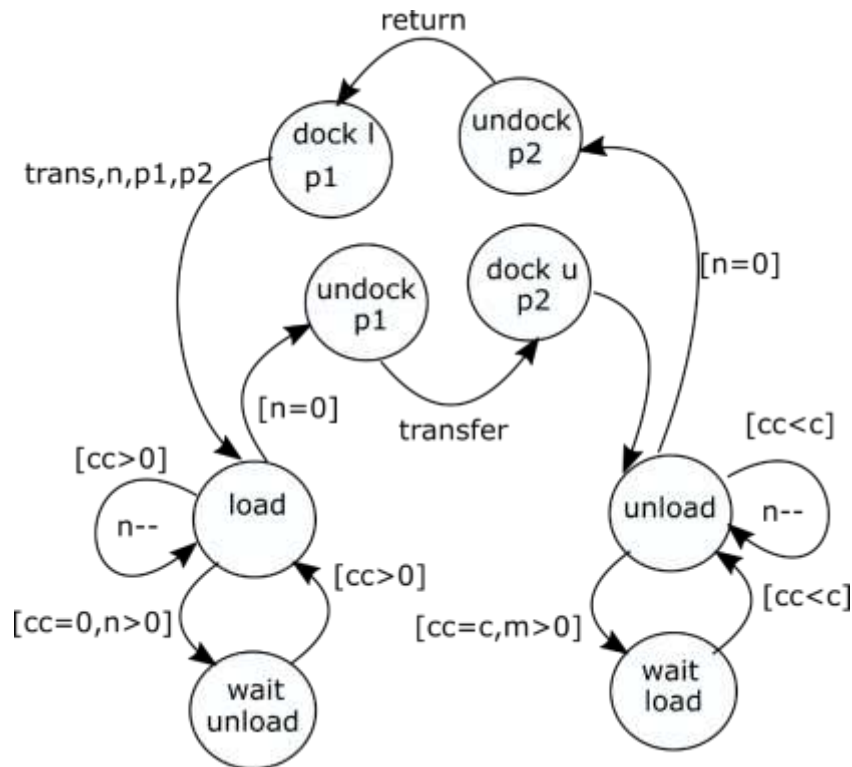
Figure 13. Behavior of an AGV

3.2.2.2.2 Conveyers, Belts, Pipes

In this case, the specific on/off command and the capacity of the transport system determine the total number of transported units. From the input buffer/buffers the quantities of material will be taken and be deposited with the delay corresponding to the selected speed in the input log. If they have multiple input ports, the streams will be aggregated. If there are multiple output ports, the streams will be split.

In Figure 14, a **conveyor** with an input port and an output port is shown. The system loads and unloads at the same time. Any stop loading or unloading stops the conveyor. Only if both processes unlock, the travel process resumes. ccb stands for the belt's current capacity (current capacity of belt). The system can be further refined by considering independent loading and unloading, but this complicates the general scheme.
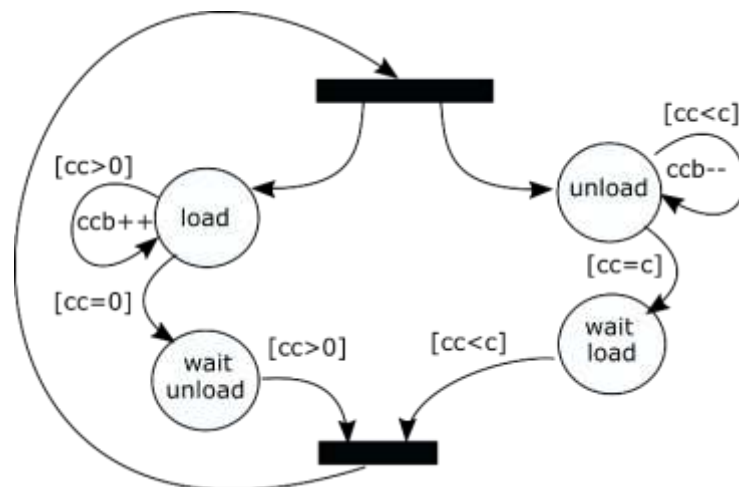

Figure 14. Behavior of a CBP

3.2.2.2.3 Manipulators

Each command specifies the quantity, port/buffer from which the port/buffer in which the materials are placed. The handler only functions if the source and destination are available. A handler that can transfer materials from buffer b1 to buffer b2 and alternately from b3 to b4 is shown in Figure 15.
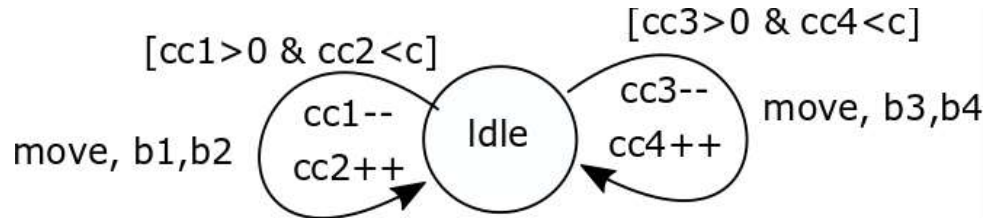

Figure 15. Behavior of a manipulator

3.2.3 Command and control elements

The control element executes the control algorithm stored in ALG. The control algorithm is described through a state machine or a Petri net. 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.

3.2.4 Basis for the graph grammar

The behavior is expressed in the categorical MLMP trough graphs transformations
We have 2 types of transformations:
- Only the transformation of the displayed attributes is required. The interface must only display the new values of the attributes (Figure 16). With the exception of AGV all other types have only change of the attributes
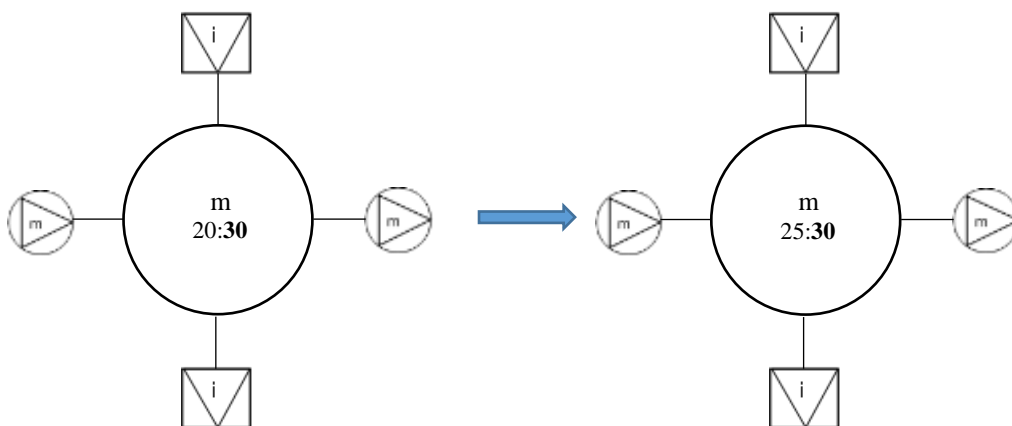
Figure 16. Graphical transformation of the attributes of a buffer (current quantity of material m). The arrow symbolizes the graphical transformation.

- The transformation of the representation graph is required (Figure 17). The AGV module must be displayed first in the current position, that is the subgraph representing AGV is attached to the connection port of the B2 buffer, and then

moved to the new position – i.e. the subgraph must be removed and attached to the connection port of buffer B7 (Figure 17). Possibly these positions could be marked graphically as the parking spaces of the AGPs.

-

Figure 17. Graphical transformation of the structure

Example of how all the different models play together → one big example scenario that links the different models.

4 Categorical specification of Modeling Language for Manufacturing Processes (MLMP)

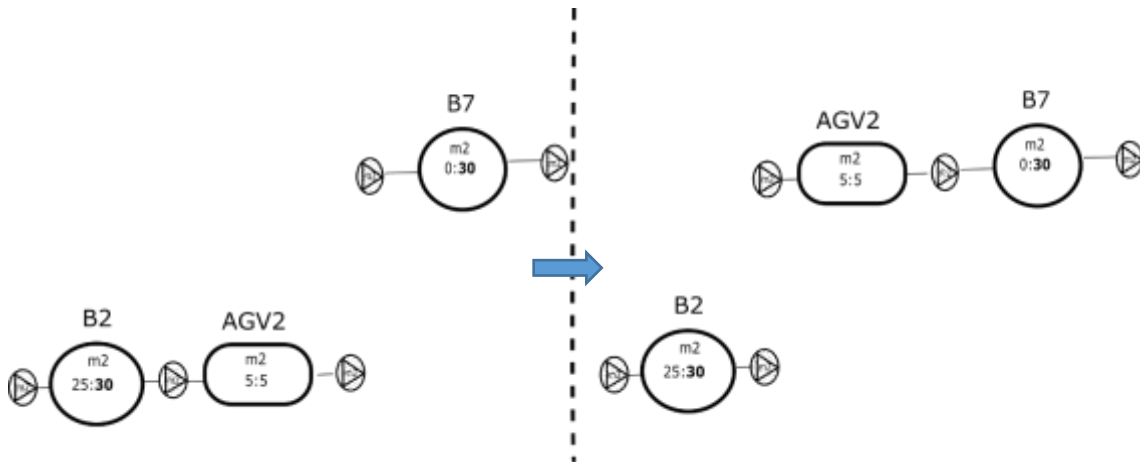Specifying a language involves establishing a syntax, which is a possible set of syntactic elements accepted in linguistic constructions, establishing a semantic domain that gives meaning to those constructions, and mapping syntactic constructions to this semantic domain [Bork2020] [Karagiannis2016].

Therefore, specifying a language must contain a syntax, a semantic domain, and a mapping of syntactic constructions to the semantic domain.

Although, at the level of language implementation, the first component specified is the syntax, in the designer's mind the semantics is what first appears, i.e. real concepts that underlie the constructions of the language and what these constructions mean. This is, in fact, the mechanism for the development of natural language, the significance of a concept first appears and then a syntactic notation is found for it, a notation needed in the communication process.

The semantics of a language is essential, because semantics describe the meaning of a language, but computers do not offer any possibility of manipulating semantics directly.

A modeling language must allow both the structure of a model and the behavior of the model to be specified. Therefore, such a language should allow both the syntax and semantics of the structure of a model to be specified, as well as the syntax and semantics of the behavior of that model.

In this section we will use category theory to formally specify syntactic constructions, both structural and behavioral, with their syntax and semantics.

4.1 Static model syntax

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.

The categorical sketch that we will use to specify the abstract syntax of the modeling language is a tuple $\mathcal{S}=(\mathcal{G}, \mathcal{C}(\mathcal{G}))$ where $\mathcal{G}$ is a graph and $\mathcal{C}(\mathcal{G})$ is a set of constraints on the classes of objects represented by the graphs nodes [Barr2012] [Diskin2012] [Wolter2015]. The graph components will be mapped to the Set category by a functor. The Set category is a category that has as objects sets and as arcs functions between these sets. Thus, each node of the graph will be transformed, in the Set category, into a set of objects of the same type and each arc of the graph will be transformed into a function. The constraints defined by a categorical sketch will be imposed on the corresponding sets of objects and functions in the Set category. Therefore, when defining the graph of a sketch and the corresponding constraints we must bear in mind that they will be mapped into the Set category. In the first phase we will define the graph of the sketch in which each atomic concept is represented by a node. The arcs of the sketch are called the sketch operators and allow the conditions to be imposed on the graph structure of the models.

We will define an MLMP model as a graph with a set of syntactic restrictions. These restrictions will then be introduced into the sketch of a modeling method metamodel based on mechanisms specific to the category theory such as commutative diagrams, limits and colimits and graph predicate signatures.

Definition 4.1.1 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:
   $X_{TS} = X_{AVG} \sqcup X_{CBP} \sqcup X_{MAN}$ where

X_{AVG} is a set of Autonomous 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, subpartitions on the set of transport systems. In addition, the definition includes connection constraints and number of arcs between different types of nodes.

4.1.1 Categorical sketch of MLMP

Categorical sketches are not designed as a notation, but as a mathematical structure that incorporates an exact formal syntax and semantics. We will use the same notations for the arcs of the graph of the sketch and the functions from Set, and the nodes from the graph of the sketch we will denote with lowercase letters and the objects from Set we will denote with uppercase letters.

We could therefore consider the starting point in defining a sketch corresponding to the meta-model a graph with two nodes x, γ and two parallel arcs σ and θ. However, this sketch is too general and does not in any way account for the specifics and restrictions of each metamodel.

Therefore, we need to introduce a series of helper objects and functions in the Set category to impose the constraints specific to each metamodel [Craciunean2018]. These helper

objects will be reflected in the sketch components (the graph of the sketch, commutative diagrams, cones and cocones).

The language offered by the classical sketch allows a precise, compact and elegant specification of the properties of the graphical models based on nodes and arcs and offers a strong mathematical context for the verification and analysis of the models.

However, the approach based on the classical sketch sometimes becomes too laborious especially because it requires the introduction of auxiliary elements and implies the definition from scratch of all the concepts involved in the model.

These deficiencies were solved by introducing Generalized Sketches which are based on the observation that a labeled diagram is an analogous construction of a logical formula i.e. mapped to the components of a graph, i.e., to the nodes and arcs of a graph [Wolter2015] [Diskin2012] . This approach preserves the benefits offered by the classic sketch and adds the facilities offered by the first order logic (FOL).

A predicate signature diagram is a tuple $\Theta=(\Pi,ar)$ where Π is a set of predicates and ar is a function $ar:\Pi \rightarrow Grf_0$ which maps each $P \in \Pi$ to an object (graph) in the Grf category. The Grf category is the category that has as its object the set Grf_0 of the graphs and as arcs the set Grf_1 of the homomorphisms between these graphs. The graph $ar(P)$ is called the shape graph arity of P. Shape graph arity becomes Shape graph for the diagrams in the classical sketch. This definition of the predicate signature allows the convenient specification of the logical constraints on the models at the level of the symbols involved in the signature of the predicates, constraints that will be reflected later on the models.

We build the corresponding MLMP model sketch. We go from the general sketch corresponding to a directed multigraph with loops (Figure 18) and introduce the restrictions in the MLMP model defined above (Definition 4.1.1). We must have in mind the idea that this graph will be mapped into Set. We will denote the sets of objects corresponding to each node with the same letters but in uppercase and for the functions corresponding to the arcs we will use the same notation as with the arcs of the graph of the sketch. Therefore, σ, θ will sometimes symbolize arcs of the graph of the sketch and sometimes the functions of the corresponding Set category.

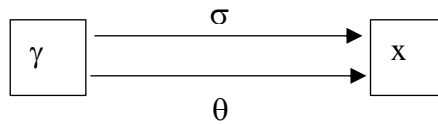
Figure 18.

We introduce the restrictions from the MLMP definition from above.

   i) $\mathcal{G}$ is a connected graph. The pushout of σ with θ introduces an equivalence class that defines the set of connected components of the graph . For the graph to be connected we must have only one equivalence class, i.e. the set of equivalence classes has the cardinal one in Set.

If we denote the pushout of σ with θ through $x \sqcup_\gamma x$ then $x \sqcup_\gamma y = (x \sqcup x)/\rho$, where ρ XX is the reflexive, symmetrical and transitive closure of the relation ρ^0 defined as follows:

$x_1 \rho^0 x_2 \Leftrightarrow \exists \gamma_{12} \in \gamma$ i.e. $x_1=\sigma(\gamma_{12})$ and $x_2=\theta(\gamma_{12})$

In the Set category the pushout of σ with θ is the colimit of the diagram from Figure 19. So, the colimit of the diagram in Figure 19 must to be a set with one element in Set category.
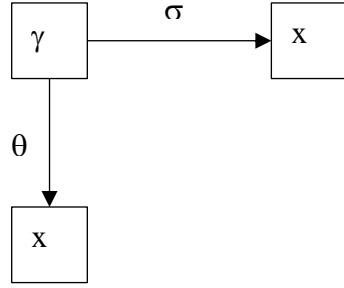
Figure 19

The graph of the sketch must contain the subframe of Figure 19.

We will use the following notation: $\text{Span}(x,y,z,r_{zx},r_{zy}) = (x\xleftarrow{r_{zx}} z \xrightarrow{r_{zy}} y)$.

We will therefore consider a predicate signature diagram $\Theta=(\Pi,ar)$ where:

$\Pi=\{$ $P_1(n_1,n_2,n_3,r_{31},r_{32}$ $)\}$ and shape graph arity of P_1, $ar(P_1(n_1,n_2,n_3,a_{31},a_{32}$))= $\text{Span}(1,2,3,r_{31},r_{32})$ defined as: $ar(n_1)=1$, $ar(n_2)=2$, $ar(n_3)=3$, $ar(a_{31})= r_{31}$, $ar(a_{32})= r_{32}$.

We then consider the diagram D_1 defined by the functor:

d_1 : $\text{Span}(1,2,3,r_{31},r_{32})\to \text{Span}(x,\gamma,z,\sigma,\theta)$ where $d_1(1)=x$, $d_1(2)=y$, $d_1(3)=z$, $d_1(r_{31})= \sigma$, $d_1(a_{32})= \theta$. In these conditions the predicate $P_1(n_1,n_2,n_3,a_{31},a_{32}$) is defined as follows: $P_1(n_1,n_2,n_3,a_{31},a_{32}$)= $|\text{CoLim}(D_1)| = 1$.

where $\text{CoLim}(D_1)$ is colimit of the diagram D_1 in Set.

We will mark these predicate signatures with a suggestive label [Conex], which will then be used to mark the graph of the sketch.

  ii) Between any two nodes there is only one arc. This constraint can be defined by a predicate with the same shape graph arity, $\text{Span}(x,y,z,r_{zx},r_{zy}) = (x\xleftarrow{r_{zx}} z \xrightarrow{r_{zy}} y)$. This predicate can be defined as follows: $P_2(n_1,n_2,n_3,a_{31},a_{32})=(\forall i_1,i_2\in n_3\Rightarrow((a_{31}(i_1)=a_{31}(i_2)\wedge(a_{32}(i_1)=a_{32}(i_2))\Rightarrow i_1=i_2))$ where the shape graph arity is $ar{:}P_2(n_1,n_2,n_3,a_{31},a_{32})\to\text{Span}(1,2,3,r_{31},r_{32})$ defined thus $ar(n_1)=1$, $ar(n_2)=2$, $ar(n_3)=3$, $ar(a_{31})=r_{31}$, $ar(a_{32})=r_{32}$.

Therefore the diagrams D_1 which maps shape graph arity to the graph of the sketch defined by the functor: $d_1{:}\text{Span}(1,2,3,r_{31},r_{32})\to\text{Span}(x,x,\gamma,\sigma,\theta)$ where $d_2(1)=x$, $d_2(2)=x$, $d_2(3)=y$, $d_2(r_{31})= \sigma$ and $d_2(r_{32})= \theta$, remains valid, and in this case. We will mark these predicate signatures with a suggestive label. In our case, such a label could be [NotMultiGraph]. These labels are then used to mark the graph of the sketch.

If we add the P_2 predicate to the signature $\Theta=(\Pi,ar)$, it becomes:

$\Pi=\{P_1,P_2\}$ and $ar(P_1)= ar(P_2)= \text{Span}(1,2,3,r_{31},r_{32})$.

  iii) On the set of nodes X we have a partition:

On the set of nodes X we have a partition: $X=X_{WS} \sqcup X_{TS}\sqcup X_{BF}\sqcup X_{MP}\sqcup X_{IP}$ where:

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

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

X_{BS} 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.

In other words, the set of objects X is the disjunctive union of three subsets of objects. This means that X is the coproduct of a discrete diagram consisting of five nodes. This discrete diagram is reflected in the graph of the sketch as in Figure 21. In the sketch of

the model the disjoint union Γ is the colimit of a discrete diagram (cocone), Discrete(x_1, …, x_n)=(x_1 … x_n).

This constraint are imposed by the predicate P_3(n, n_1, n_2, n_3 n_4, n_5)=|CoLim(Discrete(n_1, n_2, n_3 n_4, n_5)|=|n| with the graph signature ar:P_3(n, n_1, n_2, n_3 n_4, n_5)→Inclusion(6,Discrete(1,2,3,4,5)) where ar(n)=6, ar(n_1)=1, ar(n_2)=2, ar(n_3)=3, ar(n_4)=4, ar(n_5)=5. We have denoted with Inclusion(x,Discrete(x,x_1,x_2,x_3,x_4,x_5)) the graph in Figure 20, where ι_1, ι_2 , ι_3, ι_4, ι_5 are inclusion functions. We will mark this condition with the label [DisjointUnion5]. To map this graph signature to the graph of the sketch we will construct a diagram D_2 defined by the functor:

d_2: Inclusion(6,Discrete(1,2,3,4,5))→Inclusion(x,Discrete(x_{ws},x_{ts},x_{bf},x_{mp},x_{ip})) where:

ar(6)=x, ar(1)=x_1, ar(2)=x_2, ar(3)=x_3, ar(4)=x_4, ar(5)=x_5.

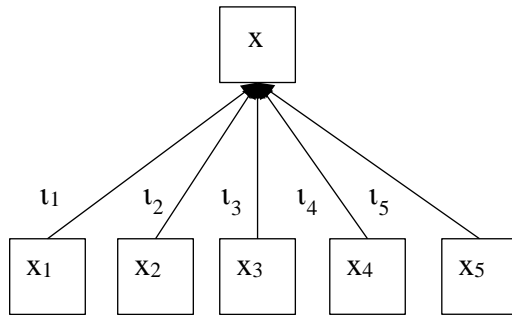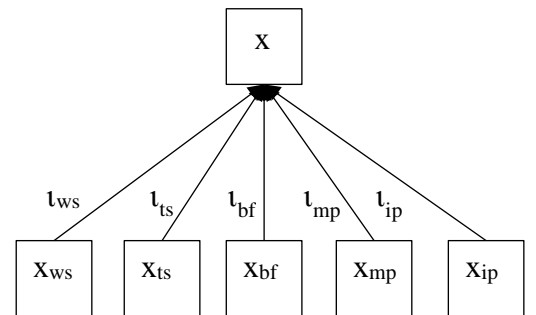
Figure 20

Figure 21

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

In the graph of the MLMP sketch we will have to include the following elements (Figure 22.) in order to be able to condition Γ to be the coproduct of the Γ_{WSMP}, Γ_{MPWS}, Γ_{BFMP}, Γ_{MPBF}, Γ_{TSMP}, Γ_{MPTS}, Γ_{IPWS}, Γ_{IPTS}, Γ_{IPBF} sets.

This constraint is similar to the previous one and are imposed by the predicate P_4(n, n_1, n_2, n_3 , n_4, n_5, n_6, n_7 , n_8, n_9)=|CoLim(Discrete(n_1, n_2, n_3 , n_4, n_5, n_6, n_7 , n_8, n_9)|=|n| with the graph signature ar:P_4(n, n_1, n_2, n_3 , n_4, n_5, n_6, n_7 , n_8, n_9)→Inclusion(10,Discrete(1,2,3,4,5,6,7,8,9)) where ar(n)=10, ar(n_1)=1, ar(n_2)=2, ar(n_3)=3, ar(n_4)=4, ar(n_5)=5, ar(n_6)=6, ar(n_7)=7, ar(n_8)=8, ar(n_9)=9. We will mark this condition with the label [DisjointUnion9]. To map this graph signature to the graph of the sketch we will construct a diagram D_3 defined by the functor:

d_3:Inclusion(10,Discrete(1,2,3,4,5,6,7,8,9))→Inclusion(γ,Discrete(γ_{wsmp}, γ_{mpws}, γ_{tsmp}, γ_{mpts}, γ_{bfmp}, γ_{mpbf}, γ_{ipws}, γ_{ipts} ,γ_{ipbf})) where:

ar(10)= γ, ar(1)= γ_{wsmp}, ar(2)= γ_{mpws}, ar(3)= γ_{tsmp}, ar(4)= γ_{mpts}, ar(5)= γ_{bfmp}, ar(6)=γ_{mpbf}, ar(7)=γ_{ipws}, ar(8)=γ_{ipts} , ar(9)=γ_{ipbf} .
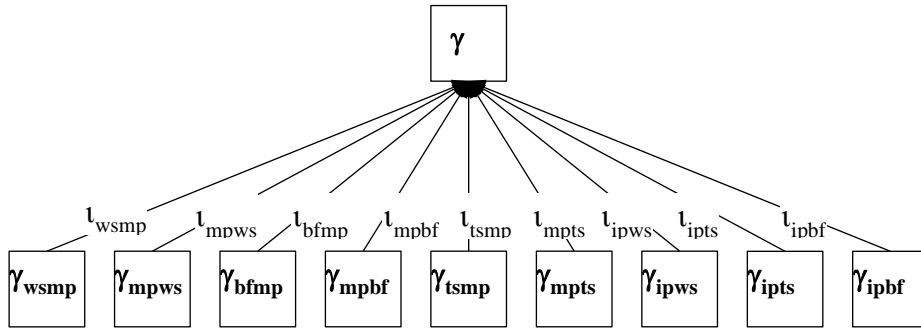
Figure 22

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

X_{AVG} is a set of Autonomous Guided Vehicles
X_{CBP} is a set of conveyors, belts, pipelines
X_{MAN} is a set of manipulators
This constraint is similar to the previous one and are imposed by the predicate
$P_5(n, n_1, n_2, n_3) = |CoLim(Discrete(n_1, n_2, n_3)| = |n|$ with the graph signature $ar(P_5(n, n_1, n_2, n_3)) = Inclusion(4, Discrete(1,2,3))$ where $ar(n)=4$, $ar(n_1)=1$, $ar(n_2)=2$, $ar(n_3)=3$. We will mark this condition with the label [DisjointUnion3]. To map this graph signature to the graph of the sketch we will construct a diagram D_4 defined by the functor:
d_4: $Inclusion(4, Discrete(1,2,3)) \rightarrow Inclusion(x_{ts}, Discrete(x_{avg}, x_{cbp}, x_{man}))$ where:

$ar(4)= x_{ts}$, $ar(1)= x_{avg}$, $ar(2)= x_{cbp}$, $ar(3)= x_{man}$.

The graph of the sketch will have to include the subgraph from Figure 23.
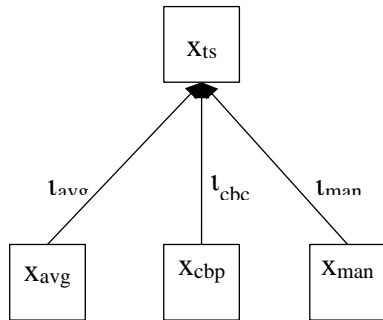

Figure 23

We have presented the graph in a subgraph sequence, each subgraph having its role in the graph of the sketch. This graph can be presented in a single image as in Figure 24.
Therefore the constraints imposed on the MLMP model determine a predicate signature diagram
$\Theta = (\Pi, ar)$ where $\Pi = \{P_1, P_2, P_3, P_4, P_5\}$ and ar is a function $ar: \Pi \rightarrow Graph_0$ which maps each predicate $P \in \Pi$ to an object in the Graph category, an object that is called shape graph arity $ar(P)$.
Then the Θ-sketch corresponding to the signature Θ is a tuple $\mathcal{S} = (\mathcal{G}, \mathcal{S}(\Pi))$ where $\mathcal{G}$ is the graph in Figure 24 and $\mathcal{S}(\Pi)$ is a family of diagram sets indexed by the set of predicates

marked Π, $\mathcal{S}(\Pi)=\{\mathcal{S}(P_1),\mathcal{S}(P_2),\mathcal{S}(P_3),\mathcal{S}(P_4),\mathcal{S}(P_5)\}$ where: $\mathcal{S}(P_1)=\{(P_1, d_1:ar(P_1)\rightarrow\mathcal{G})\}$,
$\mathcal{S}(P_2)=\{(P_2, d_1:ar(P_2)\rightarrow\mathcal{G})\}$,
$\mathcal{S}(P_3)=\{(P_3, d_2:ar(P_3)\rightarrow\mathcal{G})\}$, $\mathcal{S}(P_4)=\{(P_4, d_3:ar(P_4)\rightarrow\mathcal{G})\}$,$(P_5)=\{(P_5, d_4:ar(P_5)\rightarrow\mathcal{G})\}$.
A model of the sketch $\mathcal{S}=(\mathcal{G},\mathcal{S}(\Pi))$ is the image of a functor M:$\mathcal{G}\rightarrow$Set which validates the set of predicates Set(Π)={ Set(P_1), Set(P_2), Set(P_3), Set(P_4), Set(P_5)} where Set(Π) is obtained from $\mathcal{S}^g(\Pi)$ as follows:

Set(P_1)={(P_1,M∘d_1:ar(P_1)$\rightarrow\mathcal{G}$)},   Set(P_2)={(P_2,M∘d_1:ar(P_2)$\rightarrow\mathcal{G}$)},
Set(P_3)={(P_3,M∘d_2:ar(P_3)$\rightarrow\mathcal{G}$)},
Set(P_4)={(P_4,M∘d_3:ar(P_4)$\rightarrow\mathcal{G}$)}, Set(P_5)={(P_5,M∘d_4:ar(P_5)$\rightarrow\mathcal{G}$)}.
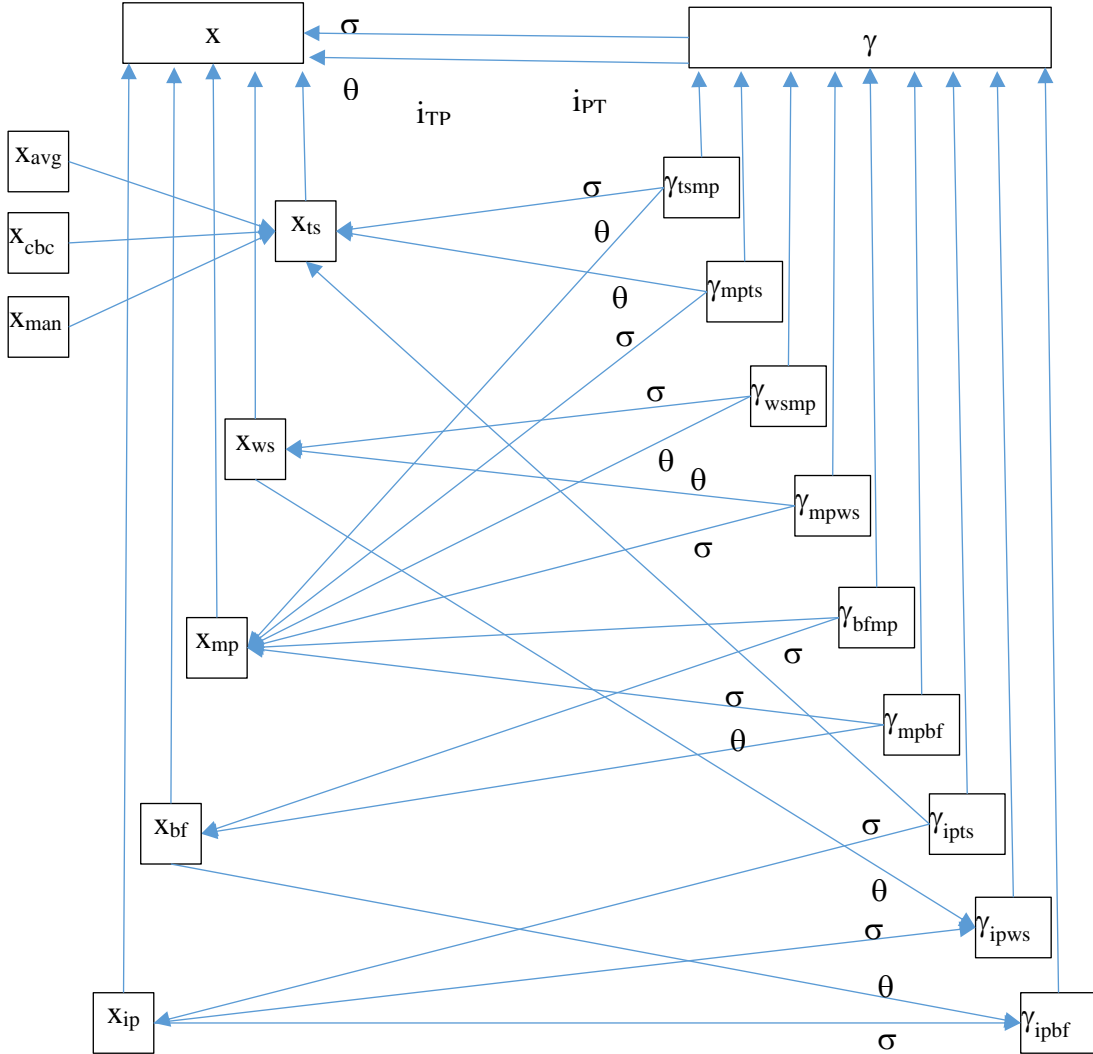
Figure 24

4.1.2 Behavioral syntax of MLMP

One of the key techniques in MDE for modeling the behavior of a system is the transformation of the model. This technique is also successfully used for the automation of other model management operations, such as code generation, model optimization, translation from one DSML to another, simulation, etc. In the case of diagrammatic models, the transformation of the models is based on the transformation of graphs, which

is a formal approach to structural changes of graphs by applying transformation rules [Plump2019] [Plump2010]. A graph rule, also called the production p=(L,R), is composed of two graphs; a left graph L, a right graph R and a mechanism that specifies the conditions and how to replace L with R.

Although, in our case, the behavioral model is not based on structural transformations of the graph, but on changes of attribute values, we will use graph transformations because they provide the necessary context to locate the components involved in a transformation and to locate the critical regions that will be defined for parallel behavioral transformations.

The double pushout (DPO) or single pushout (SPO) approaches, are transformations in successive steps of the left graph to the right graph [Plump2010] [Plump2019] [Ehrig2015].

We will specify the behavior of the MLMP model, with DPO graph transformations.

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 behavior is based. A graph transformation rule is a formal concept that precisely defines the model's behavior 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.

In the double-pushout (DPO) variant, a graphical production is denoted p=(L←K→R) and contains three graphs: a left graph L, a right graph R and an interface graph K contained both in R, and in L, where the arrows represent two total monomorphisms p_L:K→L and p_R:K→R. In this variant, a production p contains besides graphs L and R and a bonding graph K, also two total graphical monomorphisms.

The application of a production p=(L←K→R) to a graph G begins with the localization of an occurrence of L in G, given by a total match monomorphism m:L→G. Then we must construct on a graph D by deleting from G the difference between L and K, that is D=G\(L\K). The final graph H is obtained by joining to D the difference between R and K, that is H=D+(R\K). In order for D=G\(L\K) to become a graph in which all edges have a source and target, a certain bonding condition must be fulfilled, which leads to a well-defined graph D.

These graph transformations will be defined on the elements from the graph of the sketch in Figure 24. For the MLMP model, we have only two graph transformation rules p_1==(L$_1$$\overset{l_1}{\leftarrow}K_1$$\overset{r_1}{\rightarrow}R_1$) and p_2==(L$_2$$\overset{l_2}{\leftarrow}K_2$$\overset{r_2}{\rightarrow}R_2$) as we see in Figure 25 and Figure 26.

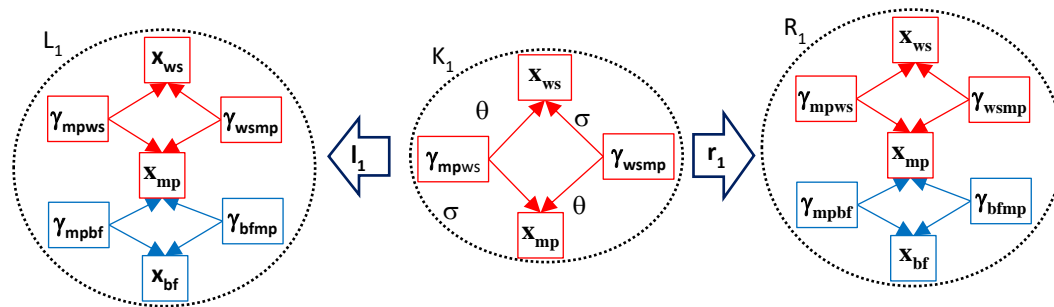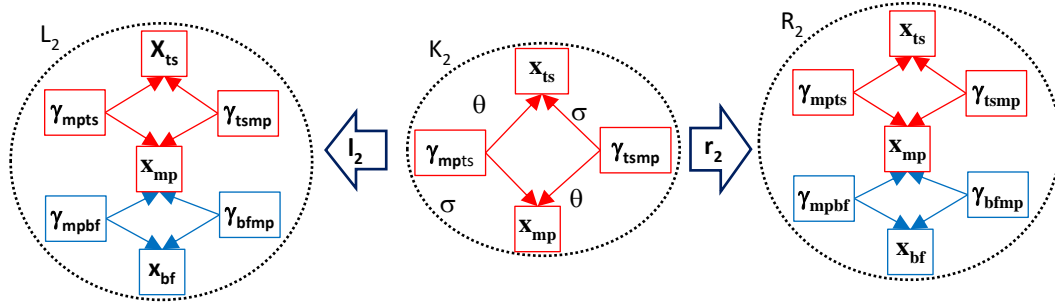Figure 25. Graph transformation p_1

Figure 26. Graph transformation p_2

Suppose we have a class of models Mod($\mathcal{S}$,Set) specified declaratively by a categorical sketch $\mathcal{S}$.

We consider that on the set of models Mod($\mathcal{S}$,Set) we have defined the set of endogenous transformation rules GTS={p_1,p_2}. The transformation rules GTS, graphically illustrates a possible evolution of the model states. Obviously, to specify the behavior of a model by such transformations, after each step the obtained model must conform to the sketch $\mathcal{S}$.

4.1.3 Semantics of MLMP Language

As we have seen, in principle, a static visual model is the image of a sketch $\mathcal{S}=(\mathcal{G},\mathcal{C}(\mathcal{G}))$ through a functor. In order to also define the behavior of a model it is necessary that the graph of the sketch be enriched besides the constraints ($\mathcal{G}$) also with types, attributes and behavioral rules.

An important extension of the graph $\mathcal{G}$ is the introduction of a type alphabet for nodes and a type alphabet for arcs, and assignment of types to each element of it [Ehrig2015] [Campbell2018] [Campbell2019]. Thus it becomes a type graph. We will consider in the following that the name of each type is identified with the name of the corresponding element of the sketch.

Then the typing of a model M: $\mathcal{S}\rightarrow$Set is made by a tuple $M^T=(M;type_M)$ where $type_M$ is a morphism from model M to the type graph $\mathcal{G}$ thus defined $type_M(X)=x$ where $X \in M(x)$ and x is a node or arc of the graph $\mathcal{G}$ of the sketch. Thus each element of a model will have a name and a type. We observe that the metamodel types can be similarly defined by a meta-metamodel.

The states of a model will be defined by the values of some attributes associated with the nodes and edges of the graph of the structure of the model as well as the structure of the model. The evolution of the model is based on the modification of the structure of the model within the limits allowed by the constraints ($\mathcal{G}$) and on the modification of the values of the attributes within the limits allowed by their type.

An attributed graph is a graph extended by attaching attributes to the nodes and edges of a graph, so that the nodes and edges can also be characterized by the attribute values. These attributes are represented by edges that link the nodes and arcs of the sketch graph to the corresponding data domain [Campbell2019].

In order to be able to define the behavior of a model at the metamodel level, we will now introduce the notions of signing a behavioral rule and signing a system of behavioral rules.

The signature of a behavioral rule is a tuple $\sigma=(L\overset{l_s}{\leftarrow}K\overset{r_s}{\rightarrow}R,C_L,Act,C_R)$ where:

L, K and R are attributed graphs L,K,R$\in$AGraph$_0$, l_s and r_s are graph monomorphisms l_s,$r_s\in$AGraph$_1$,

$C_L=(\Pi_L,ar_L)$ is a diagram predicate signature such that $ar_L:\Pi_L\rightarrow AGraph_0$, which we call the precondition signature.

$C_R=(\Pi_R,ar_R)$ is a diagram predicate signature such that $ar_R:\Pi_R\rightarrow AGraph_0$, which we call the postcondition signature.

Act is an action signature that specifies how to transform the elements of graph L which is the domain of action into the components of graph R which composes the codomain of the action.

Act has the shape graph arity a tuple $ar=(ar_L,ar_R)$, where $ar_L(Act)=L$ and $ar_R(Act)=R$. To simplify the exposure we will sometimes write an action in the form of Act(L;R). If we consider that the elements of graph L, the nodes and arcs, are $(x_1,...,x_m)$ and the elements of graph R are $(y_1,...,y_m)$ then $(y_1,...,y_m):=Act(x_1,...,x_m)$ and therefore we will denote the graph L with $L(x_1,...,x_m)$, the graph R with $R(y_1,...,y_m)$, the graph K with $K(z_1,...,z_l)$ and an action also with $Act(x_1,...,x_m; y_1,...,y_m)$. Most of the times in applications Act is a set of operations $\omega_i:x_1,...,x_m\rightarrow y_i$, $i=1,...,m$.

The behavioral signature Σ is a set of behavioral rules signatures.

To define the behavioral signatures of the MLMP language we consider the shape graphs $G_1(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9)$ from Figure 27 and $G_2(x_2, x_3, x_6, x_7, x_8)$ from Figure 28. These shape graphs are essential for mapping the behavioral transformations to the component elements of a model, and they must be defined to represent the local graph structure of the model. In constructing a model we can have variable structures such as joints and forks with a variable number of branches. In our case we have a join structure with a variable number of inputs in the X_{ws} component and therefore we will have to define a variable behavioral signature. We will represent these variable graph shapes as follows:

$G_1(<***x_7,x_2,x_6,x_1>, <*x_7,x_2>, x_3, x_4, <****x_7,x_2,x_6,x_1,x_5>, <**x_7,x_2,x_6>, x_7, x_8,x_9)$ and $G_2(<*x_7,x_2>, x_3, <**x_7,x_2,x_6>, x_7, x_8)$.

with significance $<*x_7,x_2>=\{ x_2|\theta(x_2)= x_7\}$; $<**x_7,x_2,x_6>=\{ x_6|\exists\ x\in<*x_7,x_2>$ i.e. $\sigma(x)= x_6\}$;

$<***x_7,x_2,x_6,x_1>=\{ x_1|\exists\ x\in<**x_7,x_2,x_6>$ i.e. $\theta (x_1)= x\}$ and

$<****x_7,x_2,x_6,x_1,x_5>=\{ x_5|\exists\ x\in<***x_7,x_2,x_6,x_1>$ i.e. $\sigma(x)= x_5\}$;

The components specified with this notation will not be addressed by their name in the implementation of the transformation, but by indirectness relative to the basic component x_7.

With this notation we can define joins with a variable number of branches and therefore all the shape graphs required in the case of MLMP language (Figure 29 and Figure 30).

In the case of the MLMP language, the behavioral signature is $\Sigma=\{\sigma_1,\sigma_2\}$ where:

$\sigma_1=(L^1\overset{lg}{\leftarrow}K^1\overset{rg}{\rightarrow}R^1,C_L^1 ,Act^1, C_R^1)$; $\sigma_2=(L^2\overset{lg}{\leftarrow}K^2\overset{rg}{\rightarrow}R^2,C_L^2,Act^2, C_R^2)$;

$L_1= R_1= L_2= R_2= G_1(<***7,2,6,1>,<*7,2>,3 ,4, <****7,2,6,1,5>,<**7,2,6>,7,8,9)$;

$K_1 = K_2 = G_2(<*7,2>,3, <**7,2,6>,7,8)$;

$C_L^1=(\Pi_L^1\ ar_L^1);\Pi_L^1=\{P_L^1(1,2,<*7,3>,<***7,3,8,4>,5,6,7,<**7,3,8>,<****7,3,8,4,9>)\};ar_L^1(x_i)=i,i=1,9$;

$C_R^1=(\Pi_R^1,ar_R^1);\Pi_R^1=\{P_R^1(1,2,<*7,3>,<***7,3,8,4>,5,6,7,<**7,3,8>,<****7,3,8,4,9>)\};ar_R^1(x_i)=i,i=1,9$;

$L_2= R_2=G_1(1,2,3,4,5,6,7,8,9)$; $K_2 = G_2(2,3,6,7,8)$;

$C_L^2=(\Pi_L^2 , ar_L^2); \Pi_L^2=\{P_L^2(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8,x_9)\}$; $ar_L^2(x_i)=i, i=1,9$;

$C_R^2=(\Pi_R^2, ar_R^2); \Pi_R^2=\{ P_R^2(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8,x_9)\}; ar_R^2(x_i)=i, i=1,9$;

Act^1: $(<***x_7,x_2,x_6,x_1>, <*x_7,x_2>, x_3, x_4, <****x_7,x_2,x_6,x_1,x_5>, <**x_7,x_2,x_6>, x_7, x_8,x_9)= Act^1(<***x_7,x_2,x_6,x_1>, <*x_7,x_2>, x_3, x_4, <****x_7,x_2,x_6,x_1,x_5>, <**x_7,x_2,x_6>, x_7, x_8,x_9)$;

Act^2: $(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8,x_9)= Act^2(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8,x_9)$;

In order to map the signature of a behavioral rule to the graph $\mathcal{G}$ of the sketch $\mathcal{S}$, we need a set of three diagrams D_L, D_K and D_R defined by three functors d_l, d_k si d_r, where d_k is the restriction of the functors d_l and d_r at domain K; $d_k=d_l/K=d_r/K$.

In the case of the MLMP language for the signature of the rule p_1 we have the diagrams (Figure 31):

d_l^1 : $G_1(<\!{*}{*}{*}7,2,6,1\!>,<\!{*}7,2\!>,3,4,<\!{*}{*}{*}{*}7,2,6,1,5\!>,<\!{*}{*}7,2,6\!>,7,8,9)\rightarrow G_1(\gamma_{bfmp}, \gamma_{mpws},$ $\gamma_{wsmp}, \gamma_{mpbf}, x_{bf}, x_{mp}, x_{ws}, x_{mp}, x_{bf})$ defined as $d_l^1(1)=\gamma_{bfmp}$; $d_l^1(2)=\gamma_{mpws}$; $d_l^1(3)=\gamma_{wsmp}$; $d_l^1(4)=\gamma_{mpbf}$; $d_l^1(5)=x_{bf}$; $d_l^1(6)=x_{mp}$; $d_l^1(7)=x_{ws}$; $d_l^1(8)=x_{mp}$; $d_l^1(9)=x_{bf}$;

$d_r^1=d_l^1$; and d_k^1 : $G_2(<\!{*}7,2\!>,3,<\!{*}{*}7,2,6\!>,7,8)\rightarrow G_2(\gamma_{mpws}, \gamma_{wsmp}, x_{mp}, x_{ws}, x_{mp})$ defined as restriction $d_k^1=d_l^1/K^1$;

And for the signature of rule p_2 we have the diagrams:

d_l^2 : $G_1(1,2,3,4,5,6,7,8,9)\rightarrow G_1(\gamma_{bfmp}, \gamma_{mpts}, \gamma_{tsmp}, \gamma_{mpbf}, x_{bf}, x_{mp}, x_{ts}, x_{mp}, x_{bf})$ defined as $d_l^2(1)=\gamma_{bfmp}$; $d_l^2(2)=\gamma_{mpts}$; $d_l^2(3)=\gamma_{tsmp}$; $d_l^2(4)=\gamma_{mpbf}$; $d_l^2(5)=x_{bf}$; $d_l^2(6)=x_{mp}$; $d_l^2(7)=x_{ts}$; $d_l^2(8)=x_{mp}$; $d_l^2(9)=x_{bf}$;

$d_r^2=d_l^2$; and d_k^2 : $G_2(2,3,6,7,8)\rightarrow G_2(\gamma_{mpts}, \gamma_{tsmp}, x_{mp}, x_{ts}, x_{mp})$ defined as restriction $d_k^2=d_l^2/K^2$;

The diagrams are a mechanism for associating formal graph components, represented by shape graphs to actual components in the graph of the categorical sketch.

In general, we can have more sets of diagrams and therefore we can have more behavioral rules at the metamodel level with the same behavioral signature. In our case, each signature of a behavioral rule generates a single behavioral rule at the sketch level. We will denote the set of behavioral rules induced by the behavioral signature Σ at the sketch level $\mathcal{S}$ with (Σ). In the case of MLMP language (Figure 30), $(\Sigma)=\{\mathcal{S}(\sigma_1), \mathcal{S}(\sigma_2)\}$ where:

$\mathcal{S}(\sigma_1)=(d_l^1(L^1)\overset{l_s}{\leftarrow} d_k^1(K^1)\overset{r_s}{\rightarrow} d_r^1(R^1),\{P_L^1(d_l^1(L^1))\},Act^1(d_l^1(L^1); d_r^1(R^1)), P_R^1(d_r^1(R^1)));$

$\mathcal{S}(\sigma_2)=(d_l^2(L^2)\overset{l_s}{\leftarrow} d_k^2(K^2)\overset{r_s}{\rightarrow} d_r^2(R^2),\{P_L^2(d_l^2(L^2))\},Act^2(d_l^2(L^2); d_r^2(R^2)), P_R^2(d_r^2(R^2)));$

The total morphisms l_s and r_s are defined as follows:

$l_s(d_k^1(n))= d_l^1(l_g(n)) \ \forall \ n\in K^1$; $l_r(d_k^1(n))= d_r^1(r_g(n)) \ \forall \ n\in K^1$ and respectively $l_s(d_k^2(n))= d_l^2(l_g(n)) \ \forall \ n\in K^2$; $l_r(d_k^2(n))= d_r^2(r_g(n)) \ \forall \ n\in K^2$.

To introduce the concept of behavioral model we need to first define the notion of matching a shape graph in its image through a functor in Set.

If we have a graph $\mathcal{G}=(N, A, s, t)$ and a functor $\phi:\mathcal{G}\rightarrow$Set that associates to each node $x_i\in N$, of the graph a set of objects $\phi(x_i)$ and to each arc $r\in A$ $r:x_i\rightarrow x_j$, $x_j\in N$, a function $\phi(x_k):\phi(x_i)\rightarrow\phi(x_j)$ then a matching of the graph $\mathcal{G}$ in $\phi(\mathcal{G})$ is a total monomorphism of graphs m:$\mathcal{G}\rightarrow\phi(\mathcal{G})$.

Therefore, the image of a matching m of the graph $\mathcal{G}$ in $\phi(\mathcal{G})$ is a graph $\mathcal{G}_m=(m(N), m(A), m(s), m(t))$ so that $\forall y_i\in m(N)\Rightarrow\exists \ x_i\in N$ with $y_i\in\phi(x_i)$ and $\forall a_i\in m(A)\Rightarrow\exists \ r_i\in N$ with $a_i\in\phi(r_i)$ respecting the conditions of homomorphism $m(s(r_i))=m(s)(m(ri))$ and $m(t(r_i))=m(t)(m(ri))$ for all $r_i\in A$.

We will denote the set of graph matches $\mathcal{G}$ in $\phi(\mathcal{G})$ with m(ϕ,$\mathcal{G}$).

We can now introduce the behavioral model of the sketch $\mathcal{S}=(\mathcal{G},\mathcal{C}(\mathcal{G}))$ that we call a behavioral model. A behavioral model consists of the set of all the behavioral transformations induced by the signatures of behavioral rules defined as above, that is, a behavioral signature Σ.

A behavioral model of the sketch, $\mathcal{S}$ is an application M^C: $\mathcal{G}\rightarrow$Set defined by two functors M_L: $\mathcal{G}\rightarrow$Set and M_R: $\mathcal{G}\rightarrow$Set, which maps each node of the graph $\mathcal{G}$ in sets of classes of the type of the corresponding node and each arc of the graph $\mathcal{G}$ in an appropriate function, and the set $\mathcal{S}(\Sigma)$ of behavioral rules becomes a set of behavioral rules in Set so

Set(Σ)={τ(t) | for each t∈$\mathcal{S}$(Σ) and each match m_L∈m(M_L∘d_l,L), m_K∈m(M_K∘d_K,K) and m_R∈m(M_R∘d_R,R) so that, m_L∘d_l, m_K∘d_K, m_R∘d_R are diagrams}.

More precisely, according to this definition in a behavioral model, for any behavioral transformation t∈$\mathcal{S}$(Σ) and for any pair of matches m_L∈m(M_L^C∘d_L,L) and m_R∈m(M_R^C∘d_R,R) 　　　　　we　　　　　have　　　　　rule (L$\overset{l}{\leftarrow}$K$\overset{r}{\rightarrow}$R,{P(m_L∘d_L∘ar(P))|P∈C_L},Act(m_L(d_L(L));m_R(d_R(R)), {(P(m_R∘d_R∘ar(P))|P∈C_R}) ∈Set(Σ)

where l(m_K (o))=m_K(l_s(o)), ∀ n∈K^1 ; r(m_K (o))=m_K(r_s(o)), ∀ n∈K^1 and m_L∘d_l, m_K∘d_K, m_R∘d_R are diagrams. Thus, the Set category endowed with these behavioral rules becomes the semantic universe of the behavioral model. We will call the application M^C:→Set thus defined, behavioral model.

Thus the set of behavioral rules Set(Σ) can be built automatically for each model specified by the sketch $\mathcal{S}$. For MLMP the set of behavioral rules Set(Σ) is:

Set(Σ)={τ_{11}, τ_{12}, τ_{13}, τ_{14}, τ_{21}, τ_{22}, τ_{23}, τ_{24}, τ_{25},} where

τ_{11}=((L$_{11}\overset{l}{\leftarrow}K_{11}\overset{r}{\rightarrow}R_{11}$,$P_L^1$($A_L^{11}$),Act($A_L^{11}$; A_R^{11}),P_R^1(A_R^{11})) where the following notations have been used:

L$_{11}$= R$_{11}$=G$_1$({Γ_{24}}, {Γ_{23}}, Γ_{22}, Γ_{21},{B$_3$},{P$_{12}$},WS$_1$,P$_{11}$,B$_4$); K$_{11}$=G$_2$({Γ_{23}}, Γ_{22}, {P$_{12}$},WS$_1$,P$_{11}$);

A_L^{11}=A_R^{11}={Γ_{24}}, {Γ_{23}}, Γ_{22}, Γ_{21},{B$_3$},{P$_{12}$},WS$_1$,P$_{11}$,B$_4$; A_K^{11}={Γ_{23}}, Γ_{22}, {P$_{12}$},WS$_1$,P$_{11}$;

τ_{12}=((L$_{12}\overset{l}{\leftarrow}K_{12}\overset{r}{\rightarrow}R_{12}$,$P_L^1$($A_L^{12}$),Act($A_L^{12}$; A_R^{12}),P_R^1(A_R^{12})) where the following notations have been used:

L$_{12}$= R$_{12}$=G$_1$({Γ_{16}}, {Γ_{15}}, Γ_{14}, Γ_{13},{B$_5$},{P$_8$},WS$_2$,P$_7$,B$_6$); K$_{12}$=G$_2$({Γ_{15}}, Γ_{14}, {P$_8$},WS$_2$,P$_7$);

A_L^{12}=A_R^{12}={Γ_{16}}, {Γ_{15}}, Γ_{14}, Γ_{13},{B$_5$},{P$_8$},WS$_2$,P$_7$,B$_6$; A_K^{12}={Γ_{15}}, Γ_{14}, {P$_8$},WS$_2$,P$_7$);

τ_{13}=((L$_{13}\overset{l}{\leftarrow}K_{13}\overset{r}{\rightarrow}R_{13}$,$P_L^1$($A_L^{13}$),Act($A_L^{13}$; A_R^{13}),P_R^1(A_R^{13})) where the following notations have been used:

L$_{13}$= R$_{13}$=G$_1$({Γ_5},{Γ_6},Γ_7,Γ_8,{B$_7$},{P$_3$},WS$_3$,P$_4$,B$_8$); K$_{13}$=G$_2$({Γ_6},Γ_7,{P$_3$},WS$_3$,P$_4$);

A_L^{13}=A_R^{13}={Γ_5},{Γ_6},Γ_7,Γ_8,{B$_7$},{P$_3$},WS$_3$,P$_4$,B$_8$; A_K^{13}={Γ_6},Γ_7,{P$_3$},WS$_3$,P$_4$;

In the case of the behavioral transformations τ_{11}, τ_{12} and τ_{13}, the matching of the shape graph in the model from Figure 9 imposes on positions 1,2,5 and 6 sets with one element (Figure 31) because the shape graphs look like those in Figure 27 and Figure 28.

τ_{14}=((L$_{14}\overset{l}{\leftarrow}K_{14}\overset{r}{\rightarrow}R_{14}$,$P_L^1$($A_L^{14}$),Act($A_L^{14}$; A_R^{14}),P_R^1(A_R^{14})) where the following notations have been used:

L$_{14}$=R$_{14}$=G$_1$({Γ_{33},Γ_{34}},{Γ_{35},Γ_{36}},Γ_{37},Γ_{38},{B$_9$,B$_{10}$},{P$_{17}$,P$_{18}$},WS$_4$,P$_{19}$,B$_{11}$);

K$_{14}$=G$_2$({Γ_{35},Γ_{36}},Γ_{37}, {P$_{17}$,P$_{18}$},WS$_4$,P$_{19}$);

A_L^{14}=A_R^{14}={Γ_{33},Γ_{34}},{Γ_{35},Γ_{36}},Γ_{37},Γ_{38},{B$_9$,B$_{10}$},{P$_{17}$,P$_{18}$},WS$_4$,P$_{19}$,B$_{11}$;

 A_K^{14}={Γ_{35},Γ_{36}},Γ_{37}, {P$_{17}$,P$_{18}$},WS$_4$,P$_{19}$;

In the case of the behavioral transformation τ_{14}, the matching of the shape graph in the model from Figure 9 impose on positions 1, 2, 5 and 6 sets with two elements so that the shape graphs look like those in Figure 29 and Figure 30.

τ_{21}=((L$_{21}\overset{l}{\leftarrow}K_{21}\overset{r}{\rightarrow}R_{21}$,$P_L^2$($A_L^{21}$),Act($A_L^{21}$; A_R^{21}),P_R^2(A_R^{21})) where the following notations have been used:

L$_{21}$= R$_{21}$=G$_1$(Γ_{28}, Γ_{27}, Γ_{26}, Γ_{25},B$_1$,P$_{14}$,AVG$_1$,P$_{13}$,B$_3$); K$_{21}$=G$_2$(Γ_{27}, Γ_{26}, P$_{14}$,AVG$_1$,P$_{13}$);

A_L^{21}=A_R^{21}=Γ_{28}, Γ_{27}, Γ_{26}, Γ_{25},B$_1$,P$_{14}$,AVG$_1$,P$_{13}$,B$_3$; A_K^{21}=Γ_{27}, Γ_{26}, P$_{14}$,AVG$_1$,P$_{13}$;

τ_{22}=((L$_{22}\overset{l}{\leftarrow}K_{22}\overset{r}{\rightarrow}R_{22}$,$P_L^2$($A_L^{22}$),Act($A_L^{22}$; A_R^{22}),P_R^2(A_R^{22})) where the following notations have been used:

$L_{22}= R_{22}=G_1(\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4,B_2,P_1,AVG_2,P_2,B_7)$; $K_{22}=G_2(\Gamma_2, \Gamma_3, P_1,AVG_2,P_2)$;

$A_L^{22}=A_R^{22}=\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4,B_2,P_1,AVG_2,P_2,B_7$; $A_K^{22}=\Gamma_2, \Gamma_3, P_1,AVG_2,P_2$;

$\tau_{23}=((L_{23}\overset{l}{\leftarrow}K_{23}\overset{r}{\rightarrow}R_{23},P_L^2(A_L^{23}),Act(A_L^{23}; A_R^{22}),P_R^2(A_R^{23}))$ where the following notations have been used:

$L_{23}= R_{23}=G_1(\Gamma_{20}, \Gamma_{19}, \Gamma_{18}, \Gamma_{17},B_4,P_{10},AVG_3,P_9,B_5)$; $K_{23}=G_2(\Gamma_{19}, \Gamma_{18}, P_{10},AVG_3,P_9)$;

$A_L^{23}=A_R^{23}=\Gamma_{20}, \Gamma_{19}, \Gamma_{18}, \Gamma_{17},B_4,P_{10},AVG_3,P_9,B_5$; $A_K^{23}=\Gamma_{19}, \Gamma_{18}, P_{10},AVG_3,P_9$;

$\tau_{24}=((L_{24}\overset{l}{\leftarrow}K_{24}\overset{r}{\rightarrow}R_{24},P_L^2(A_L^{24}),Act(A_L^{24}; A_R^{24}),P_R^2(A_R^{24}))$ where the following notations have been used:

$L_{24}= R_{24}=G_1(\Gamma_9, \Gamma_{10}, \Gamma_{30}, \Gamma_{32},B_8,P_5,M_1,P_{16},B_{10})$; $K_{24}=G_2(\Gamma_{10}, \Gamma_{30}, P_5,M_1,P_{16})$;

$A_L^{24}=A_R^{24}=\Gamma_9, \Gamma_{10}, \Gamma_{30}, \Gamma_{32},B_8,P_5,M_1,P_{16},B_{10}$; $A_K^{24}=\Gamma_{10}, \Gamma_{30}, P_5,M_1,P_{16}$;

$\tau_{25}=((L_{25}\overset{l}{\leftarrow}K_{25}\overset{r}{\rightarrow}R_{25},P_L^2(A_L^{25}),Act(A_L^{25}; A_R^{25}),P_R^2(A_R^{25}))$ where the following notations have been used:

$L_{25}= R_{25}=G_1(\Gamma_{12}, \Gamma_{11}, \Gamma_{29}, \Gamma_{31},B_6,P_6,M_1,P_{15},B_9)$; $K_{25}=G_2(\Gamma_{11}, \Gamma_{29}, P_6,M_1,P_{15})$;

$A_L^{25}=A_R^{25}=\Gamma_{12}, \Gamma_{11}, \Gamma_{29}, \Gamma_{31},B_6,P_6,M_1,P_{15},B_9$; $A_K^{25}=\Gamma_{11}, \Gamma_{29}, P_6,M_1,P_{15}$;
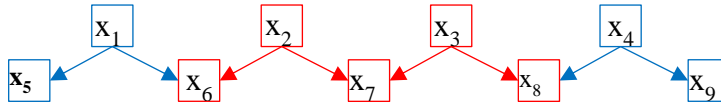

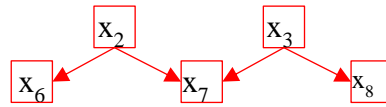Figure 27. Graph $G_1(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9)$
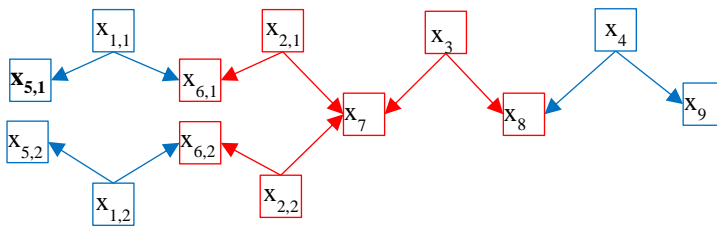

Figure 28. Graph $G_2(x_2, x_3, x_6, x_7, x_8)$
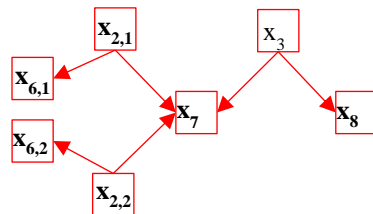

Figure 29. Graph $G_2(x_2, x_3, x_6, x_7, x_8)$

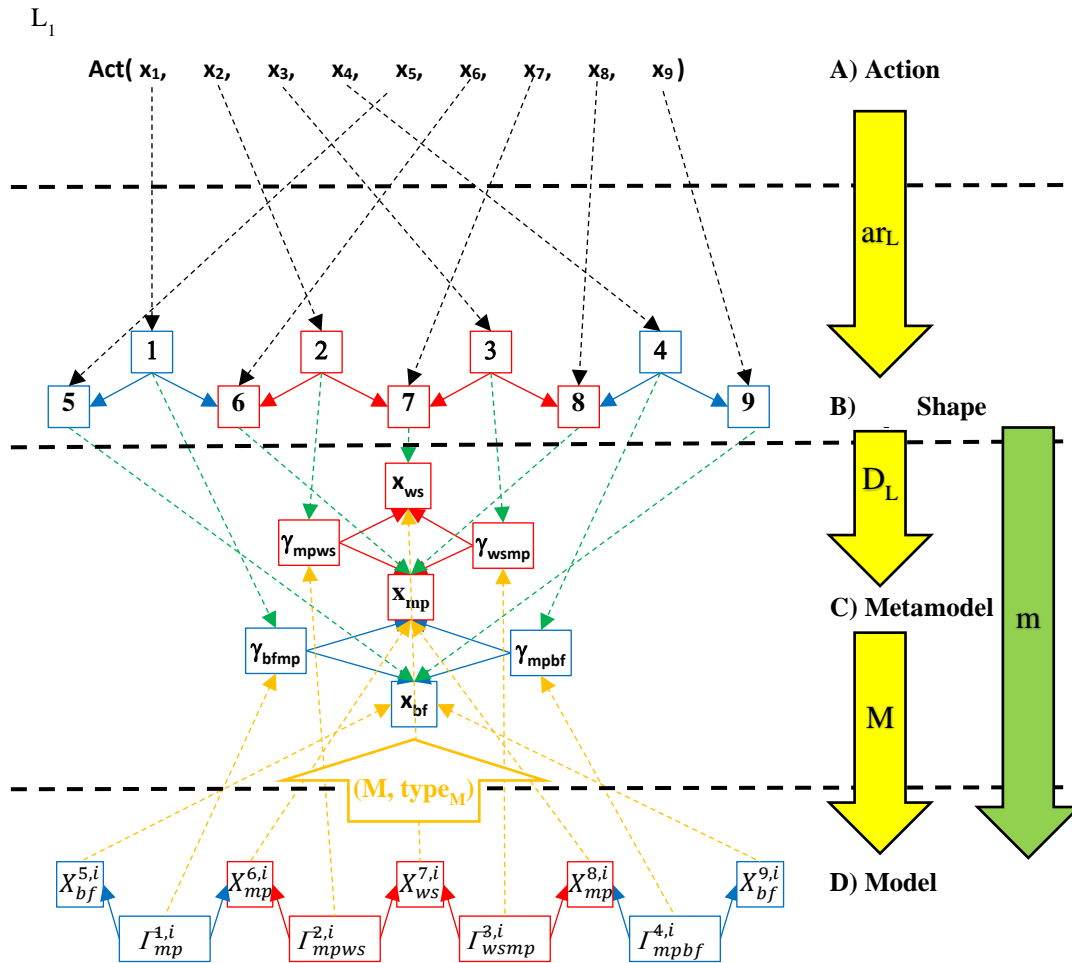Figure 30. Graph $G_2(x_2, x_3, x_6, x_7, x_8)$

Figure 31. Graph

Model transformation involves addressing two important aspects: defining transformations and applying these transformations. In MDE, the transformation of the models has as main objective the automatic generation of some models written in a target language based on models written in a source language according to the transformation rules.

Applying a transformation rule specified by a behavioral signature $\sigma=(L\overset{l_s}{\leftarrow}K\overset{r_s}{\rightarrow}R,C_L,Act,C_R)$ is done as follows:

- We find a total matching morphism m:L→G (Figure 31).
- The preconditions are verified, that is, the fulfillment of the predicates defined by the C_L signatures, among which is the gluing condition.
- The graph transformation defined by the cospan $L\overset{l_s}{\leftarrow}K\overset{r_s}{\rightarrow}R$ is executed.
- The Act action is executed.
- The postcondition is verified, that is, the fulfillment of the predicates defined by the C_R signatures. If the postconditions are not fulfilled the rule cannot be applied and rollback is performed.

If a transformation rule appears several times in a model then it can be applied sequentially or in parallel several times. Also, if two different transformation rules are independent, they can be applied simultaneously. Graphic transformation systems can be non-deterministic, i.e. in a certain step there are several transformation rules that respect

the conditions of application and one of them must be chosen or, there are several matches and one of them must be chosen. There are several techniques for controlling these situations.

The dynamic behavior of an MLMP model over time is accomplished by generic algorithms that implement the behavioral 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 behavioral transformations executed. The semantics of an MLMP defines how process tokens are propagated through the arcs and objects of a model.

In the modeling method concept the simulation of a model is based on mechanisms and algorithms that are written in a programming language. The behavior 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.

5 Conclusions

The document describes the development of a modeling method and the associated DSML - MLMP to be used in the design of a digital factory of the future.

The first section summarizes the mathematical fundamentals of developing the modeling language and methods. Arguments are presented, sustaining the adequacy of the chosen mathematical instrument - category theory - for this task.

These fundamentals are used in Section 2 to formalize the description of the static and dynamic aspects of a manufacturing process. On this basis, the development process for a modeling method described in [Karagiannis&Kühn2002] [Bork2019] was chosen.

The elements of the MLMP - which is an essential aspect of the modelling method - are presented together with their graphical representation in Section 3. The elements are depicting conceptual building blocks used by professionals in the field of manufacturing processes to design manufacturing floor plans. An example of a model built from this elements is also provided.

Section 4 describes the development of the metamodel of the MLMP

The category theory is used to formally specify syntactic constructions of the language, both structural and behavioral.

The developed metamodel will be used for the implementation of the design tool for the factory of the future. Although the presented case models only the base level of the Reference Archetecture Model for Industry 4.0 – RAMI 4.0 (Reference Architecture Industry 4.0) [Anderl2016]. The final design tool will support multiple levels of the architecture specification.

They are at least two ways of supporting multiple views/ abstraction levels in the modeling and design tools. One is to use a unique model including all necessary elements and all their attributes of the modeled universe. Each view is then filtering the specific elements.

The other way is to have distinct models for each view and consequently distinct modelling languages and modelling tools for each view. The aggregating tool should allow the coupling of the models trough defined interfaces and the simulation of the whole universe.

The decision will be taken in the tool design phase. The present document will be then supplemented with the description of additional elements and attributes or of the additional languages and model interconnecting interfaces.

6 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. [Mironescu2019] Mironescu I.D. (2019) An ADOxx Based Environment for Problem Based Learning in Manufacturing Systems Design, 9th International Conference on Manufacturing Science and Education – MSE 2019 "Trends in New Industrial Revolution", MATEC Web Conf., Vol. 290, 2019, DOI:
   https://doi.org/10.1051/matecconf/201929014003

5. [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)

6. [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).

7. [Craciunean2019]D.C. Crăciunean, *Categorical Grammars for Processes Modeling*, International Journal of Advanced Computer Science and Applications(IJACSA), 10(1), (2019)

8. [Karagiannis&Kühn2002] Karagiannis D., Kühn H. (2002) Metamodelling Platforms. In: Bauknecht K., Tjoa A.M., Quirchmayr G. (eds) E-Commerce and Web Technologies. EC-Web 2002. Lecture Notes in Computer Science, vol 2455. Springer, Berlin, Heidelberg

9. [Barr2012] Michael Barr And Charles Wells, Category Theory For Computing Science- Reprints in Theory and Applications of Categories, No. 22, 2012.

10. [Diskin2012] Zinovy Diskin, Tom Maibaum- Category Theory and Model-Driven Engineering: From Formal Semantics to Design Patterns and Beyond, ACCAT 2012

11. [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,

12. [Plump2019] D. Plump, 'Computing by graph transformation: 2018/19', Department of Computer Science, University of York, UK, Lecture Slides, 2019.

13. [Campbell2019] G. Campbell, B. Courtehoute 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.

14. [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. [Hristakiev2018] I. Hristakiev, 'Confluence analysis for a graph programming language', PhD thesis, Department of Computer Science, University of York, UK, 2018. [Online]. Available: https://etheses.whiterose.ac.uk/20255/.

15. [Plump2010] D. Plump, 'Checking graph-transformation systems for confluence', ECEASST, vol. 26, 2010. DOI: 10.14279/tuj.eceasst.26.367.

16. [Ehrig2015] Hartmut Ehrig, Claudia Ermel, Ulrike Golas, Frank Hermann, Graph and Model Transformation General Framework and Applications, Springer-Verlag Berlin Heidelberg 2015

17. [Milner2009] R. Milner, The Space and Motion of Communicating Agents, Cambridge University Press, (2009)

18. [Anderl2016] Anderl R. (2016). Industrie 4.0 - Digital Transformation in Product Engineering and Production. Conference: 21st International Seminar on High Technology - Smart Products and Smart Production, At Piracicaba (SP), Brazil

7 Annex A. List of Abbreviations

DSL	Domain Specific Language
DSML	Domain Specific Modeling Language
FOL	First Order Logic
MDE	Model-Driven Engineering
MLMP	Modeling Language for Manufacturing Processes
DPO	Double PushOut
SPO	Single PushOut
BPMN	Business Process Model and Notation
EPC	Event-driven Process Chain
UML	Unified Modeling Language