Interoperable data model for simulation-in-the-loop

Michele Ciavotta
Department of Informatics, Systems and Communication
University of Milano Bicocca
Milan, Italy
michele.ciavotta@unimib.it

Andrea Bettoni, Gabriele Izzo
Department of Innovative Technologies
University of Applied Sciences of Southern Switzerland
Manno, Switzerland
{name.surname}@supsi.ch

***Abstract*—In the context of industry 4.0, where factory components become more and more intelligent, the role of virtualization and simulation becomes central. This paper presents a flexible, modular, scalable, extensible and interoperable modelling language expressly designed to support multi-disciplinary simulations. In particular, the language is based on a meta-model that provides patterns of both hierarchical and graphical aggregations and imposes a modelling process based on the principle of separation of concerns using a layered model. An example of applying the model to a business use case is also reported.**

***Index Terms*—Digital Continuity, Interoperability, Virtualization, Collaborative Simulation, Digital Twin**

I. Introduction

In the manufacturing context, simulation refers to a broad cluster of methods, techniques and resulting software applications aimed at modeling and analyzing, through simulated experiments, the behavior of real production systems [1], [2]. Simulation enables to test the effects of design choices by exploring *what-if* scenarios without having to involve the real shop floor and, potentially, to evaluate its performance even before it is actually deployed.

Historically designers and managers have used these tools to perform specific analysis targeting separated factory domains [3] such as process validation, optimization of production layouts, scheduling and purchasing forecast, to name a few. Today, the rising complexity of distributed value networks calls for systems that extend beyond the single company borders and disciplinary domains, envisioning the application of simulation techniques to be integrated into cooperative digital environments. It is only by abating those barriers that industrial companies will be able to leverage on the availability of Big Data, to enable collaborative decision-making processes and to align intra- and inter-company operations. To fully achieve these goals, simulation needs to grasp the benefit coming from the rising of the disruptive changes experienced by its technological enablers. As more and more factory smart objects, namely Cyber Physical Systems (CPS), become capable of establishing a capillary sensing of the shop floor and make it available in real-time, the distance between the simulation run-time and the real factory situation results shortened; this allows moving from what-if analysis of possible scenarios to the support in *now-what* decisions by applying simulation to current production issues. Moreover, since the computational power and reasoning capabilities are distributed as distributed are CPSs, simulation processes can now be distributed and located closer to shop floor by exploiting emerging paradigms such as Edge Computing.

However, as [4] points out, there are still relevant gaps affecting the maturity level in the adoption of these enablers in simulation and forecasting. First, although high-performing computing services are available in the Cloud, simulation sitll under-exploit them, being almost always stuck in a centralized and localized vision that failed to take advantage of the opportunities offered by Cloud first and, more recently by the Edge computing. Even the game-changing possibility to reliably collect and use in real-time data from the shop floor to mirror the factory is not currently exploited within simulation tools still struggling in the definition of virtual factory models and needing responsive big data management infrastructures [5]. Lack of flexible and extensible data models are hindering the emergence of multi-disciplinary simulation technologies to support the virtual investigation in a holistic perspective that promotes decision-making processes enriched by the analysis of the multiple domains interacting within the factory [6]. Ultimately, digital models need to evolve coherently with the growth of production systems along their life-cycles to leap over both punctual and paradigmatic transformations taking place in the shop floor and in the information systems that support them [7].

II. Industrie 4.0: Filling the Gaps

In the current context, dominated by the advent of the Industrie 4.0, production systems are characterized by a higher complexity of decision-making processes that are loosely distributed among the CPSs operating along the entire value network. It is time to envision interoperable simulation services capable to operate near the data sources to support first-time-right solutions to multi-disciplinary issues occurring in production, logistics and management processes.

This paper proposes four items that should lead the path of simulation technologies development to embrace and empower the digital transformation of industry. First, a virtual interoperable shop floor representation, relying on a shared modeling of core data that can be expanded to suit specific purposes, is required to sustain integration of different domains and provision of cloud-based simulation services. Especially SMEs, the least capable among industrial organizations to access the benefits of simulation, will profit from this democratizing approach that removes or reduces the adoption barriers while expanding the scope of simulation. Interoperability of data models will pave the grounds for achieving situation awareness, the second item, that embodies the digital doppelganger (a.k.a. Digital Twin) concept to create live digital copies of the simulated environments and processes thus making it possible to oversee and control them on a factual basis. The achievement of the first two items will allow simulation to take place in the loop of factory operations, dramatically changing its role from a predictive and testing techniques that is use ex-ante to make decisions on possible designs, to a responsive and holistic system to support decision-making in real-time scenarios. Finally, simulation models will need to be flexible and reconfigurable to leverage the smart nature of CPSs pursuing the plug-and-simulate condition, the last item, in which adding of new resources or modifying their behavior is automatically or semi-automatically mirrored in the corresponding factory digital twin. Supporting such ambitious objectives calls for a set of requirements that data models have to fulfill. To this end, our data model needs to be:

Expressive - to enable the description of possibly any resource or flow involved in production, logistics and management processes of different industrial domains.

Extensible - to build, upon a core representation of the factory environment, a growing model ecosystem capable to maintain the digital information available all along the factory life-cycle.

Interoperable - to provide users with a personalized view and proper simulation tools that present the virtual environment from their own point of view thus supporting the decision-making processes, activity planning and operation controlling.

Scalable - that is the ability to function efficiently when the context is changed in size or volume featuring multi-level access features and suitable aggregation patterns.

Modular - so that it provides representation building blocks that can be rearranged and reconfigured to follow the continuous evolution of the real factory.

III. Interoperable Virtualization and Simulation

For Industrie 4.0, the shop floor ceases to be a rigid environment, firmly regulated by time-based automation systems; the whole approach to manufacturing has to be entirely redesigned to be composed of intelligent elements (from simple sensors to production machines and robots) that are often identified with the term CPS [8]. These elements are capable of interacting with each other, with the environment, and with products by sharing information on their state, making the entire production environment highly flexible. In order for this vision to be achievable, the elements of the digital factory need to be *smart* [9], [10], that is, programmable and with sufficient computational capacity to react to events in the appropriate time frame; moreover, they must also be connected through communication channels that guarantee a reduced latency.

The role of communication in smart manufacturing environments is twofold: on the one hand it guarantees that the actors at shop floor level can interact, by exchanging valuable information, and orchestrate production in a distributed way; on the other hand it enables the almost real-time monitoring, paving the way for the creation of CPS digital twins. In fact, the information collected in large quantities from CPSs can serve both to refine/learn their behavior in order to obtain always-accurate simulation models (*Real-to-Digital Synchronization*), and to feed the simulation with events and data coming from the factory in real time (*Mirroring*). Both features prelude the integration of simulation in the heart of production process to exploit its potential not only in the factory design and planning phase but also in the operative one with a multitude of possible applications (*Simulation-in-the-loop*).

Achieving such an ambitious objective confronts us with a plethora of exiting challenges, including the definition of a flexible, modular, extensible yet interoperable language to describe the digital double of the factory to come. In this paper we present a data model expressly designed to support simulation in the framework of Industrie 4.0. To provide the modeler with a powerful and flexible tool, we have built the data model atop a core language (a meta-model) based on a mixed white-box/black-box paradigm, generic enough to shape the traits of any CPS using plain terms like *attributes* and *artifacts*.

Simulation in Industry is fundamentally multidisciplinary, covering different application domains. Accordingly, depending on the domain, the same digital twins can be typified differently as distinct are the objectives of the simulations. In order to better manage the situation in which different simulators are involved, but also help modularize the work of people involved in the modeling, we have brought the principle of *separation of concerns* as well as the concept of artifact to the language. Both these features will be presented in full detail in the next section, while here we aim at briefly presenting the intended objectives. As for the former, we have structured the language so that the digital representation of a production layout (from a single machine up to a whole factory) is represented by a set of layers, each devoted to the description of a particular domain. Consider, for example, the case where the model should be processed by a Kinematic and a Discrete Event Simulator (DES). The model in question will be structured in at least three levels, a first level containing kinematic and 3D models of the entities involved in the simulation, the second level will contain logical models describing the behavior of entities in reaction to events, and the third level will link the physical description to the related logic. As far as the latter is concerned, the language features black-box binary items as part of the overall description of the digital twin. This choice was forced by the fact that simulators often use very different behavioral models, and they export them using incompatible (often proprietary) formats. Thus, a white-box/black-box approach has been adopted to enable the modeler to define transparent items (e.g., Elements, Archetypes, and Attributes, see Section IV-A) as well as binary (and solution dependent) elements.

Figure 1 displays the general architecture of the simulation-

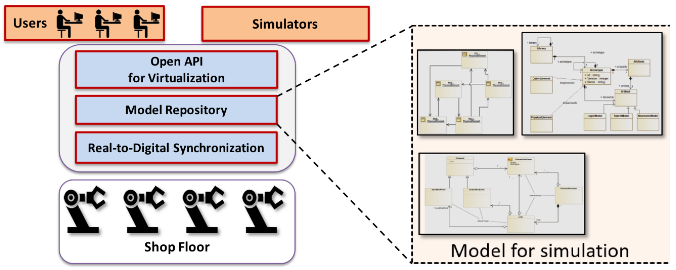

Figure 1. Architecture supporting the simulation-in-the-loop

in-the-loop system as conceived within the project motivating our work. The model is created and managed via an open API. This can be used both by modelers to create the digital representation of the factory at issue and by simulators. The model is saved in a repository and continuously updated through a module in charge of implementing Real-to-Digital synchronization. As regards the Open API for virtualization they are structured in two levels. The lower API level allows the management of core meta-model elements while the upper level features higher edpoints, specific for each simulation class. This second level maps the particular concepts of the domain using the underneath API. In this level all the logic that enforces a semantic coherence of the domain (e.g. integrity check, validation) is implemented. Simulation tools outside the platform will be able to access preferably only the level that concerns their domain.

The Real-to-Digital Synchronization can be defined as the process of continuously updating of CPS models stored in the Model Repository, tracking the evolution of the shop floor. Having up-to-date models is especially important in simulation since CPSs along their life cycle are subject to aging, straining, and reconfiguration processes, which can change their behavior and performance; in this situation the simulation outcomes may result substantially different from reality and, therefore, of very limited utility. In order to be able to make decisions based on reliable simulation models, it is of paramount importance to detect changes in the CPSs automatically and continuously, and to adapt parameters and scenarios accordingly. The core of the synchronization grounds in the processing of data gathered at shop floor level. Synchronization with shop floor, while massively involving the digital representation of the factory, goes beyond the objectives of this document. An extended discussion of this subject is deferred to future publications.

IV. A Model for Simulation

One of the most common features of manufacturing plants is that they are mostly designed out of standardized components (machine tools, robots, etc.) composed in a modular way. Through a good organization of modules, in fact, it is possible to speed up engineering as well as the simulation setups, maximizing the reuse of components. Similarly, simulation software tools often provides libraries of models that can be aggregated and extended to assemble a full plant layout. In this work we aspire to provide the same efficient re-use approach implementing classes to describe resources, called Archetypes and Elements, which are instances of such elements and the components of the plant model. The relationship that exists between Archetypes and Elements is similar to the one that exists in Object Oriented Programming (OOP) [11] between a Class and an Instance (Object) of that class.

Another important requirements is that the resulting data model shall support semantically meaningful collection of relations (henceforth referred to as *Layers*). We deem particularly important to provide the modelers with a tool to gather links of the same type. The idea here is to simplify the modeling process by dividing the overall models in several levels. By way of example, a first layer may contain the definition of the plant topology with its hierarchies of production resources; the other are graphs that can be used to express relations between resources. Logical, electrical, and pneumatic layers are just a few examples of layers. Figure 2 illustrates graphically this multi-layer modeling approach.

This remainder of section documents the resulting data model for simulation, which is organized into 9 areas shortly described below:

Core Model - documents the core classes used for the definition of entities and relations used in other sections.

Archetype Model - introduces the concepts of Archetypes as the basis of the model reuse paradigm.

Element Model - this model presents the concept of Element as the instantiation of an Archetype according to the OOP approach.

Connection Model - the connection model is devoted to the definition of the links between two model entities.

Layer Model - presents the concept of layer as a collection of items that can be further specified using attributes.

Attribute Model - specifies the concept of property to be used to annotate other elements of the model, namely Library, Archetype, Artifact, and Endpoint.

Role Model - presents a set of classes that defined the concept of Role as semantic description to further specify the constituent elements of the model.

Plant Model - introduces the Plant as an aggregation of layers representing among other aspects its physical and logical nature.

Project Model - documents all the classes that represent multi-plant simulation projects and that enable simulation tools to share plant models and results.

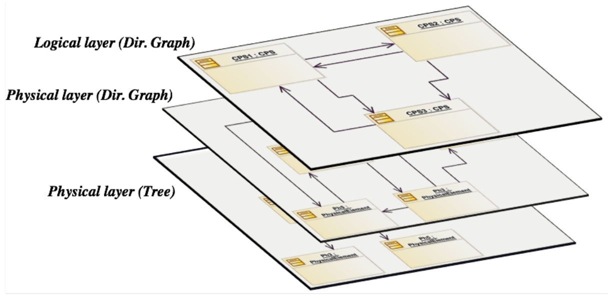

Figure 2. Modeling using superimposed layers

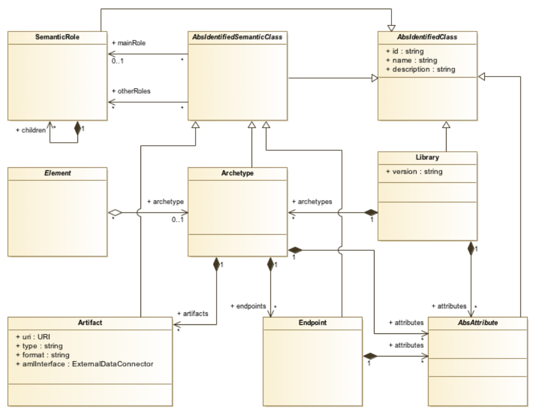

Figure 3. Class Diagram of the Core model - Part I

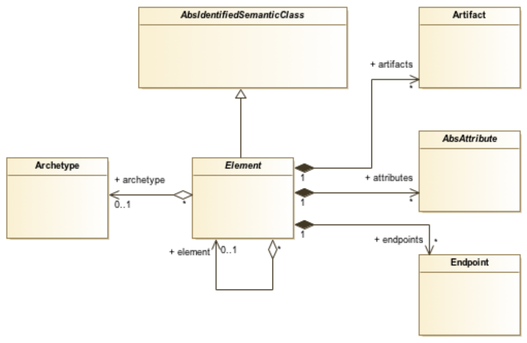

Figure 4. Class Diagram of the Core model - Part II

In this paper, nonetheless, due to the limited space available, only the main concepts and elements of the meta-model will be presented.

A. Core Model

Our modeling language is entirely upheld by a meta-model featuring the ground concepts and the sintax of the language. Nine entities are contained in the core model. Figures 3 and 4 illustrate graphically such entities along with their relations, using the UML class diagram notation. We have chosen to divide the core model into two separate diagrams to improve the fruition. The central components of the model are: the archetype and the element. These have a symmetrical structure: both are characterized by one or more roles that semantically identify them, by a unique identifier within the model, by attributes that can be complex at will, by endpoints and artifacts. An element can be defined from an archetype but unlike OOP this is not strictly necessary. This means that an element can be defined as a stand-alone component or that it is possible to add or modify the characteristics inherited from an archetype.

The other elements of the model are:

Semantics, identification, and grouping of language elements, namely *SemanticRole*, *AbsIdentifiedClass*, *AbsIdentifiedSemanticClass*, and *Library*.

Artifacts - references to external relevant models treated as third-party black-box components (e.g., Geometry Models).

Attributes - properties used by the modeler to specify Archetypes and Elements.

Endpoints - semantically defined extremes of links connecting Archetypes or Elements. Links, and the rules by which they can be created, are defined in the Connection Model.

B. Artifact Model

One of the cornerstone requirements that our data model has to fulfill is to provide support also for black-box attributes, referred to as *Artifacts*; it will allow managing elements that represent information that is processable only by a specific software solution. An artifact, in our model, can be defined as an informative fiche containing, among other pieces of information, a URI that points to a particular file. Four further specifications are provided off-the-shelf (see Figure 5):

KinematicMode - this class represents a reference to an external model carring information about the kinematic behavior of the artifact the model is referred. For instance a file in COLLADA [12] format also carries information about the kinematics and could be a candidate for this kind of model. Nonetheless, specific simulators might use their own binary format to express kinematic properties.

GeometryModel - this class embodies the concept of a black-box model holding geometry-related information about the particular plant resource. 3D models as COLLADA or JT [13] belong to this particular category.

LogicalModel - this class serves as a reference for models that define the behavior of plant resources. Such models, expressed for instance in PLCOpen XML [14], fully describe the actions of the actors involved in the plant as machines, CPS, sensors, personnel or software. While other models describe the static nature of a plant, Logical Models (or Behavioral models) describe the way the elements react to events (or clock ticks).

SynchModel - this class represents the synchronization model defined. In a nutshell, such models are binary files to be executed by an edge-computing platform to carry out the real-to-digital synchronization process, that is the mechanism in charge of updating the digital representation of a plant resource through the processing of data collected on the shop floor. A comprehensive description of the synchronization model, the abstractions on which it is based and its execution mode is beyond the scope of this document.

C. Connection Model

The connection model, within our model for simulation, is the part devoted to the definition of the connection mechanisms between two entities. To this end, the Endpoint class (see Section IV-A) is extended and the concepts of *Link* and *ConnectionLayer* are introduced. The concept of link is described by means of an abstract class (*AbsLink*) then extended into two concrete classes (*UndirectedLink* and *DirectedLink*). The UndirectedLink joins together two instances of the Endpoint

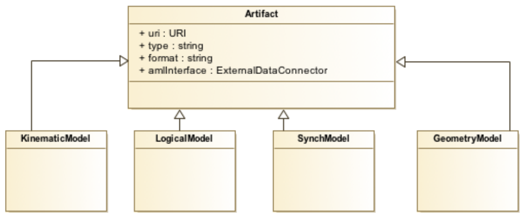

Figure 5. Class Diagram fo the Artifact Model

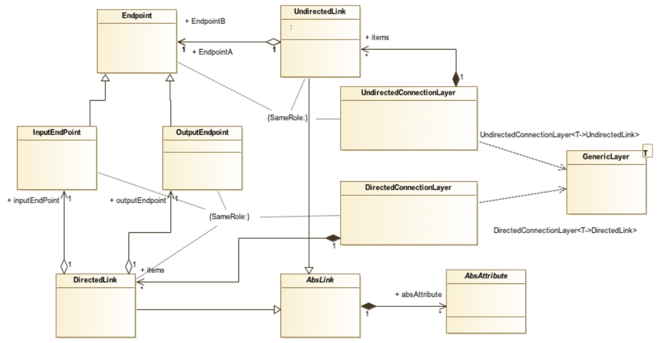

Figure 6. Connection Model

class and represents a connection between the elements to which the endpoints belong. This link does not provide any information about the direction of such a connection. To express the concept of direction, a DirectedLink has to be used instead. Figure 6 shows the components that shape the connection model along with the relations that exists among them. The reader should notice that links are collected within either a *DirectedConnectionLayer* or an *UndirectedConnectionLayer*, both instantiating the Template class *GenericLayer*$\langle T \rangle$. Finally, the concepts of Endpoint, Link and ConnectionLayer, in their directed or undirected form, are subject to the constraint that all the actors must share the same role in order to be semantically meaningful.

The purpose of this model is to provide the tools (links and layers) to define aggregations of elements in the form of direct or indirect graphs. Supporting this aggregation pattern is one of the model requirements that, together with the hierarchical aggregation pattern, allows to shape one or more plants by superimposing different layers. The advantage of this approach is that it enables distributed, flexible and independent creation and evolution of the model. Different experts, in fact, can work in the definition of different elements and connect them by means of a graph of semantically defined links.

D. Plant Model

The concepts presented so far define a meta-model containing core items and tools to extend them. Starting from those core classes, specific manufacturing and simulation concepts have been devised. The Plant component is one of those, it is a specification of the Element class featuring set of resources (physical, logical, and product) as well as their connections. Furthermore, since one of the goals of our meta-model is

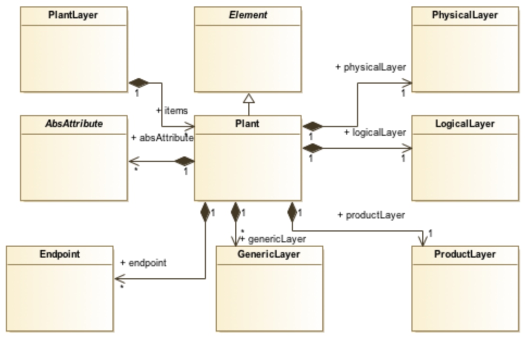

Figure 7. Class diagram of the Plant model

to enable a layer-based modeling approach wherein different actors (experts in different domains) can easily collaborate in defining a complete simulation model (see Figure 2), the fact that the structure proposed to represent a plant also includes several layers should not surprise. Some of them are mandatory and specific as the physical, logical and product layers, which model hierarchies of physical, logical and product entities, respectively. Other are optional like one or more GenericLayers, which present to model graph-shaped relations between the elements of the mandatory layers. Furthermore, Endpoints can be defined to model connections among plants (via Endpoints).

Plants in turn can be aggregated into their own level that, together with other pieces of information, defines a Simulation Project.

V. Use Case: White-goods Production

The rationale of this section is to present a use case scenario, highlighting the way it has been modeled using the proposed data model. The use case describes the hob handling and palletizing operations carried out within a real production plant in Italy. The plant produces over a hundred different configurations of cooking hobs, belonging to three categories: gas, electric or pyro-ceramic. The production is organized in batches; hobs of the same type are aggregated taking into account different orders. Each batch (and every hob in it) is identified by a code to ensure a high level of traceability. For the same reason, as soon as a single hob is released, an automated reading system records its associated event. The cooking hobs are realized by different production stations; each station produces one hob at a time and it is specialized in producing items of only one category. Seven palletizers are placed at the end of the stations; they have the task of aggregating in a pallet elements belonging to the same lot, assigning a unique 2D barcode to the pallet and placing them on a conveyor. The length of the conveyor, its average speed as well as the shelf life of the pallets on it are known. On the end of the conveyor there is a 2D barcode reader and an automated sorter system. The sorter has the task of dispatching the pallets out of the conveyor to different shipping lines. The plant features 18 shipping lines freely assignable to any type of product.

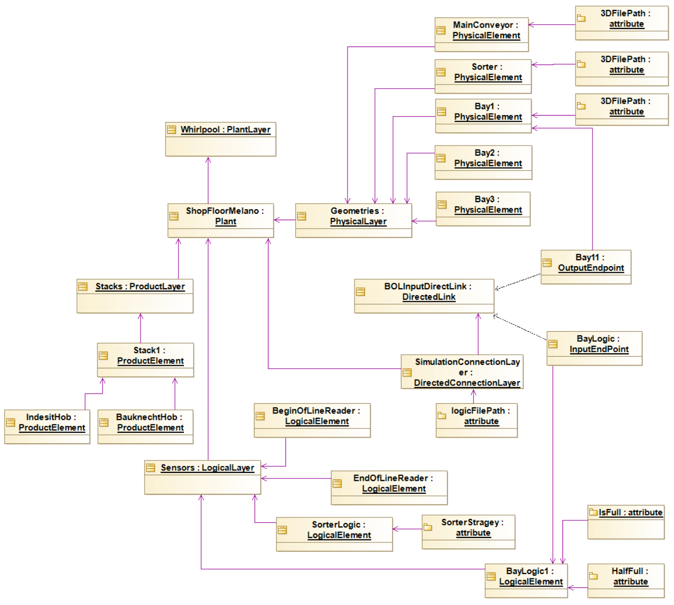
3DFilePath : attribute
MainConveyor : PhysicalElement
3DFilePath : attribute
Sorter : PhysicalElement
3DFilePath : attribute
Bay1 : PhysicalElement
Whirlpool : PlantLayer
Bay2 : PhysicalElement
Bay3 : PhysicalElement
ShopFloorMelano : Plant
Geometries : PhysicalLayer
Bay11 : OutputEndpoint
Stacks : ProductLayer
BOLInputDirectLink : DirectedLink
BayLogic : InputEndPoint
Stack1 : ProductElement
SimulationConnectionLayer : DirectedConnectionLayer
IndesitHob : ProductElement
BauknechtHob : ProductElement
BeginOfLineReader : LogicalElement
logicFilePath : attribute
Sensors : LogicalLayer
EndOfLineReader : LogicalElement
IsFull : attribute
SorterLogic : LogicalElement
SorterStragey : attribute
BayLogic1 : LogicalElement
HalfFull : attribute

Figure 8. Plant diagram of the *Sorter* use case

An example of data model instantiation for this use case is presented in Figure 8; it represents the plant according to the Plant Model presented in Section IV-C. Following the layered structure at the basis of the modeling approach enforced by the data model, the plant is defined as a collection of layers, and specifically:

- a Physical layer, defining the main elements of the plant along with their artifacts as geometries, representing 3D models of the components (conveyors, sorter and bays) of the graphical simulation model;
- a Logical layer, defining the internal automation logic of all the components involved in the simulation;
- a Product layer, describing the product stacks and the single product units composing them.
- a DirectedConnectionLayer named SimulationConnectionLayer, which connects the physical elements and the logical ones.

VI. Conclusions

In this paper a flexible and innovative data model explicitly designed to support multi-disciplinary virtualization and simulation within the frame of Industrie 4.0 is presented. At the same time, we have explained the needs the model addresses and motivated the decisions made. In what follows some highlights are reported:

1) In order to grant a high degree of reusability the Object Oriented Paradigm has been taken as reference to develop the Archetype-Element duality. In this way the model enables the creation of libraries of components to be reused in different simulation/virtualization scenarios.
2) To achieve a high level of abstraction, flexibility, and compatibility among different simulators a meta-model featuring a white-box/black-box approach has been proposed.
3) To streamline the process of modeling in a multidisciplinary scenario, a layered pattern is implemented and enforced.

Future work will concentrate on providing the language with concepts pertaining to the time, and CPS state. In addition, the language will provide specific constructs for the description of data flows generated by CPS with the aim of facilitating and automating the real-to-digital synchronization process.

Acknowledgment

This project has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 723094 (FAR-EDGE).

References

[1] C. A. Chung, *Simulation modeling handbook: a practical approach*. CRC press, 2003.
[2] S. Bangsow, *Manufacturing simulation with plant simulation and simtalk: usage and programming with examples and solutions*. Springer Science & Business Media, 2010.
[3] J. W. Fowler and O. Rose, "Grand challenges in modeling and simulation of complex manufacturing systems," *Simulation*, vol. 80:9, pp. 469–476, 2004.
[4] P. Pedrazzoli, D. Corti, M. Dal Lago, M. Cocco, D. Cerri, M. Taisch, and S. Terzi, "Simulation and forecasting technologies: Gaps and challenges for their future role in manufacturing," in *International ICE Conference on Engineering, Technology and Innovation (ICE)*, 2014, pp. 1–4.
[5] M. Ciavotta, M. Alge, S. Menato, D. Rovere, and P. Pedrazzoli, "A microservice-based middleware for the digital factory," *Procedia Manufacturing*, vol. 11, pp. 931–938, 2017.
[6] P. Hehenberger, B. Vogel-Heuser, D. Bradley, B. Eynard, T. Tomiyama, and S. Achiche, "Design, modelling, simulation and integration of cyber physical systems: Methods and applications," *Computers in Industry*, vol. 82, pp. 273–289, 2016.
[7] A. Azab, T. AlGeddawy *et al.*, "Simulation methods for changeable manufacturing," *Procedia CIRP*, vol. 3, pp. 179–184, 2012.
[8] N. Jazdi, "Cyber physical systems in the context of Industry 4.0," *2014 IEEE Automation, Quality and Testing, Robotics*, pp. 2–4, 2014.
[9] J. Lee, "Smart Factory Systems," *Informatik-Spektrum*, vol. 38, no. 3, pp. 230–235, 2015.
[10] E. Hozdić, "Smart factory for industry 4.0: A review," *International Journal of Modern Manufacturing Technologies*, vol. 7, no. 1, pp. 28–35, 2015.
[11] P. Wolfgang, "Design patterns for object-oriented software development," *Reading Mass*, p. 15, 1994.
[12] R. Diankov, R. Ueda, K. Okada, and H. Saito, "Collada: An open standard for robot file formats," in *Proceedings of the 29th Annual Conference of the Robotics Society of Japan, AC2Q1–5*, 2011.
[13] A. Katzenbach, S. Handschuh, and S. Vettermann, "Jt format (iso 14306) and ap 242 (iso 10303): The step to the next generation collaborative product creation," in *Digital Product and Process Development Systems*, G. L. Kovács and D. Kochan, Eds., 2013, pp. 41–52.
[14] E. van der Wal, "Plcopen," *IEEE Industrial Electronics Magazine*, vol. 3, no. 4, p. 25, 2009.