Production Recipe Validation through Formalization and Digital Twin Generation

[1]Stefano Spellini, [1]Roberta Chirico, [1]Marco Panato, [2]Michele Lora, [1]Franco Fummi
[1]Department of Computer Science - University of Verona - name.surname@univr.it
[2]Singapore University of Technology and Design - name_surname@sutd.edu.sg

***Abstract*—The advent of Industry 4.0 is making production processes every day more complicated. As such, early process validation is becoming crucial to avoid production errors thus decreasing costs. In this paper, we present an approach to validate production recipes. Initially, the recipe is specified according to the ISA-95 standard, while the production plant is described using AutomationML. These specifications are formalized into a hierarchy of assume-guarantee contracts. Each contract specifies a set of temporal behaviors, characterizing the different machines composing the production line, their actions and interaction. Then, the formal specifications provided by the contracts are systematically synthesized to automatically generate a digital twin for the production line. Finally, the digital twin is used to evaluate, and validate, both the functional and the extra-functional characteristics of the system.**

The methodology has been applied to validate the production of a product requiring additive manufacturing, robotic assembling and transportation.

***Index Terms*—Smart Manufacturing, Design Automation, Digital Twin, Simulation and Validation.**

I. Introduction

The *Industry 4.0* is a new industrial automation trend introduced at the Hanover Fair in 2011 [1]. It proposes to revolutionize the entire manufacturing world by integrating many new technologies, to massively improve production volumes and quality while creating and optimizing business models to obtain an always increasing cost efficiency. In this context, Cyber-Physical Systems (CPSs) provide a new degree of controllability to production machines, while supplying extremely precious data about the executed processes. This opens to promising scenarios, like the raise of *Digital Twins* [2], creating a virtual representation of a real production line to emulate its current and future behavior. A system stakeholder may be interested in exploiting such novel possibilities to modify, to optimize, and generally to improve the existing manufacturing systems. Also, digital twins are also useful during the planning phase, to avoid bottlenecks or to certify the feasibility of the desired production. Each scenario involves the contribution of simulation techniques, to aid the validation and optimization of manufacturing systems [3].

In such an innovative context, multiple tools, languages, and standards are involved in enabling the communication and the correct interpretation of messages between various production levels. From the business level, which has its expression in the Enterprise Resource Planning (ERP) systems, to the control aspects implemented by Programmable Logic Controllers (PLCs) installed on plant components, a central role is played by the Manufacturing Execution System (MES). It aggregates data from other production software modules and acts to improve the production and to solve incurring manufacturing problems. A plethora of information domains with different data models and semantics is involved in these tools, making communication a difficult task.

This work has been partially supported by the *MIUR "Dipartimenti di Eccellenza" 2018-2022* grant.

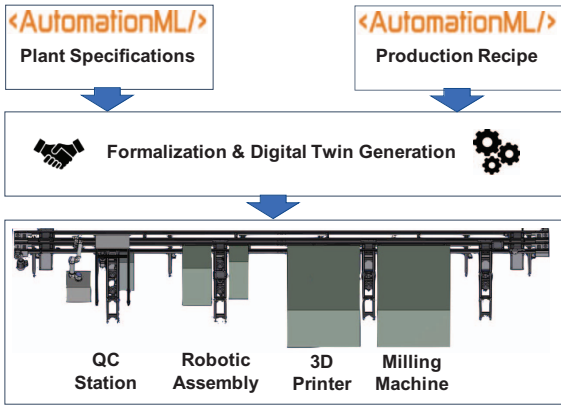

Figure 1: Conceptual overview of the proposed methodology: the AutomationML data format is used as an aggregator of information regarding the production line. The formalization of such an information set is carried out using assume-guarantee reasoning through contracts. The simulation of the contract's implementation ensemble is carried out to validate a specified production recipe.

To tackle such issues, different standards have been developed, such as AutomationML (AML) [4] for the architectural and plant topology view, and ANSI/ISA-95 [5] for the business level and the Manufacturing Operations Management (MOM). When (re-)designing, configuring, or optimizing a production system, the complexity induced by the presence of such heterogeneous information is hardly bearable if tackled manually. Thus, design automation becomes crucial to obtain a digital model of the production system. In recent decades, the field of Electronic Design Automation (EDA) provided a vast amount of methodologies, approaches, and formalisms tackling heterogeneous representations of requirements and systems [6]. The results achieved by fifty years of research in EDA may be a good source of ideas, approaches, and methodologies to tackle the design of machinery in the context of Industry 4.0. We are interested in how those methodologies can help to define and create a production system's digital twins. Such a powerful tool provides a robust simulation that can be programmed and, thus, can give answers on production recipes feasibility over the system model.

As depicted in Figure 1, this paper proposes a formalization

methodology over different aspects of the manufacturing system, from the topology of the production line to production recipes. In addition, the specific behavior of each manufacturing machine composing the production line can be characterized by existing industrial standards [7]. The exploited specification formalism is based on a hierarchy of Assume-Guarantee (A/G) contracts, where each level represents a machine's specification stage, from plain production considerations to functional models depicting generic machine's actions. Each contract is composed of a set of information and constraints gathered from different sources. As an example, the Manufacturing Execution System (MES) is capable of detailing production recipes and the AML standard provides plenty of data regarding machines and production line topology. The consistency of each contract certifies the soundness of different properties, but to fully verify the applicability of a certain production recipe on a production line, we exploit simulation procedures over digital twins. In fact, this approach generates a set of contract's implementations that can be easily imported into discrete-event plant simulators. The system's simulation is capable of representing plant behavior regarding a specific production process.

We apply the proposed methodology on a real-world case study, showing that a formalized production recipe can be validated on a systematically constructed digital plant model.

II. Background

A. State-of-the-art and Contribution

In order to simulate a manufacturing plant, it is necessary to produce its model first. Creating models of already existing machines may be an error-prone process that may lead to inaccurate models [8]. As such, it may be ideal to start from formal specifications of systems and components. In [9], a method to formally specify industrial component behaviors has been proposed: it focuses on control logic components, the sensor/actuator behaviors and it presents how to build a formal specification to verify their correctness. However, this approach does not rely on simulation. As such, employing only formal methods, it may lead to complexity issues. A combination of formal methods with simulation is described in [10], [11]: synthesis from Linear Temporal Logic (LTL) specifications and simulation are used in combination to design and validate a robotic system. However, the proposed formalization is specifically tailored for robotics applications.

In this work, we extend the state-of-the-art by proposing a systematic (*i.e.*, guided by industrial standards) formalization of the production line's components and processes. Then, we present a methodology producing the digital-twin of a manufacturing system from its formalization.

B. Languages for manufacturing process specification

1) The ANSI/ISA-95 standard: has been developed by the International Society of Automation (ISA) to define the interface between the enterprise structure and the control systems [12]. The standard defines three main categories of interest, each defining a set of information models interfacing the different parts of a manufacturing company: *information models between business and manufacturing operation systems*, *information models for activities defined in manufacturing operation systems*, and *information models within manufacturing operation systems*.

The automation pyramid is assumed to be composed of 5 different levels, each managed by different systems and different timeframes. We are interested specifically levels 3 and 4, operating respectively on Manufacturing Operations and Control Information and Business planning and logistics. Parts 1, 2 and 5 of the standard are dedicated to the proposition of a consistent terminology between automation Levels 4 and 3, to bridge the the information gap between two different manufacturing views. More precisely, a set of data models formally details the types of information that pass through both systems, such as the *Product definition* model, which describes processes and requirements to make a product, or the *Resource definition* model, that characterizes available resources such as pieces of equipment, materials and also personnel.

2) AutomationML: it is an XML-based data format, created to provide a formal exchange model for heterogeneous engineering tools [13]. In industrial contexts, it is capable of describing plant components under different points of view. AutomationML objects encapsulate information about plant topology and geometry, as well as kinematics and behavior logic. Different standards are strongly intertwined within AutomationML to compose a complete description of the system to be characterized. In particular, the Computer Aided Engineering Exchange (CAEX) (IEC 62424) provides the features to represent a topological view of the system. CAEX may be also used to store hierarchical object information, acting as a top-level format for each of the other standards. This standard is mainly based on the following elements [14]:

- *RoleClassLib* is used to detail the functionality associated to plant types of equipment.
- *SystemUnitClassLibrary* provides a collection of vendor-specific equipment instances.
- *InterfaceClassLibrary* defines all useful interfaces to connect each plant model and external XML standards.
- *InstanceHierarchy* depicts object instances modeling the plant topology and hierarchy.

AutomationML modular structure allows the integration with other XML data formats. As an example, [15] describes the linking of objects between AutomationML and ANSI/ISA-95 and its implementation, Business To Manufacturing Markup Language (B2MML) [16]. The purpose of B2MML is to foster the full integration between manufacturing and business views, and to provide a consistent data model between ERP and MES systems. As such, the procedures to make a manufacturing system consistent with the ANSI/ISA-95 standard can be greatly simplified, since most modern software support XML parsing and syntax checking.

C. Assume-Guarantee Contracts

A contract C for a component M is a triplet (V, A, G): V is the set of the component variables, while A and G are sets of *assertions* describing behaviors over V [17]. A is the set of the contract's *assumptions*, *i.e.*, the behavior of the environment of M assumed by the model. G is the set of *guarantees*, *i.e.*, the behaviors guaranteed by M whenever the assumptions hold.

A component M implements a contract C whenever M and C are defined over the same set of variables and all the behaviors of M satisfy the guarantees of C in the context of the assumptions. Moreover, a component E can also be associated with a contract C as an environment for the contract. E is said to be a legal environment of C whenever the behaviors specified by E are a subset of A. The A/G contract theory [17] defines a set of operations; this work uses the *composition* operation to build complex contracts from simpler ones, and the *consistency* operation to check the non-emptiness of the set of implementations satisfying a contract.

Behaviors described by assumptions or guarantees of a contract may be expressed using many different formalisms [18]. Using LTL formulas is pretty common when describing the behavior of a reactive system [19]. However, checking the realizability of a LTL contract is 2-EXPTIME-complete, which makes it practically infeasible [20]. Luckily, reducing the LTL expressiveness to its *General Reactivity (GR(1))* fragment it exists an algorithm able to decide the realizability in $O(N^3)$ [21]. Such an algorithm, implemented by different tools, performs reactive synthesis from a GR(1) specification, to obtain a Mealy Machine implementing a control strategy for the specified reactive system [22], [23].

D. Case study

The manufacturing case study used throughout the paper is based on the *Industrial Computer Engineering (ICE) laboratory*[1]: a research facility focused on advanced manufacturing and equipped with a full-size production line. The case study implements an additive manufacturing scenario. The manufacturing system is composed of a plastic Fused Deposition Modeling (FDM) 3D Printer, a Quality Check (QC) station and a collaborative robotic assembly cell. The transportation of final products or workpieces towards different stations is implemented through a complex system of conveyor belts. The specified production recipe, depicted in Figure 2, starts from the 3D printing of a small plastic LEGO®-like brick. Once the printing process is finished, the piece is checked for defects in the QC station. In the meantime, two other plastic bricks are gathered from the warehouse and are assembled together in the robotic assembly station. Once the printing and quality control processes are completed, the produced and verified piece is assembled to the ensemble of other pieces. The final product has to be verified in the QC cell for assembly defects and then it is transported to the warehouse.

III. Contract-based system formalization

We rely on a hierarchy of A/G contracts to formalize the different aspects of the production plant. Figure 3 summarizes these contracts and their relations. The hierarchy of contracts is made of four different levels, grouped into two categories: *Recipe contracts*, and *Machine contracts*.

The *Recipe contracts* are meant to formalize the production requirements by describing the characteristics of the item meant to be produced, and consequently the actions to be performed to transform the initial raw material. The recipe

[1]University of Verona, Department of Computer Science, Computer Engineering for Industry 4.0: *http://www.di.univr.it/?ent=progetto&id=4935*

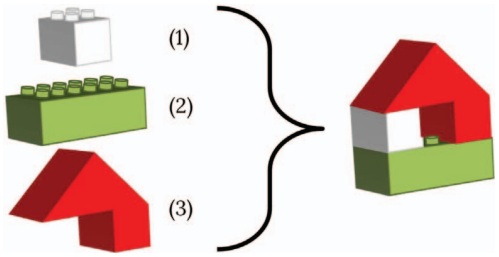

Figure 2: 3D representation of the parts that compose the final product of the case study. The (1) and (2) pieces are gathered from the warehouse, while (3) has to be 3D printed.

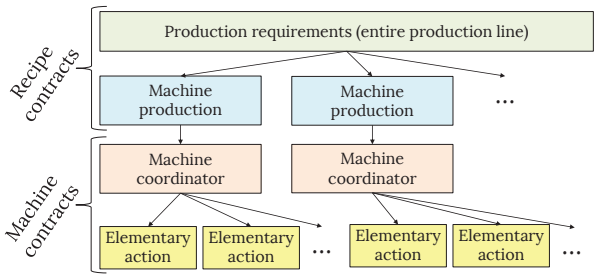

Figure 3: Summary of the A/G contracts used in the formalization.

contracts are organized on two levels. The *Production requirements contract* specifies the sequence of operations to be performed. The contract assumes that the entire ensemble of machines composing the production line provides the required operations. Then, it guarantees that the specified sequence is realizable using the machines ensemble. The *Production requirements contract* is decomposed in a set of *Machine production* contracts, one for each machine in the production line. Each machine contract guarantees the availability of the actions provided by the machine while assuming the physical details of such operations.

Each machine production contract is *refined* by a *Machine coordinator* contract. The latter belongs to the sub-hierarchy of the *Machine contracts*. For each machine, a *Machine coordinator* contract formalizes the operations provided by the machines, guaranteeing the availability of such operations and assuming the capacity of the machine of performing the set of elementary actions necessary to implement each operation. The behavior of each machine is formalized by a set of *Elementary action* contracts describing the "primitive" actions performed by the machine.

Both the Recipe contracts and Machine contracts can be automatically generated from AutomationML specifications. Recipe contacts are generated from ISA-95 descriptions, as described in Section IV. Meanwhile, the machine contracts are generated automatically from AutomationML, relying on the methodology proposed in [7]. Such contracts are used to generate a digital twin for the production plant by exploiting automatic synthesis from contracts and automatic code generation [24]. Section V show the formalization of the machines of the case study and the generation of the digital twin for the production line. Finally, Section VI will show how the digital twin can be used to validate the production line, and to gather production extra-functional data.

IV. Production recipe formalization

A production recipe is a set of particular manufacturing steps necessary to produce a specific product. Each step is implemented through different machines and it is characterized by a set of attributes, such as machine parameters, process duration or the Bill of Materials (BOM) that is required to fulfill the production. This information is typically detailed in the MOM model given by the ISA-95 standard. Assuming a MES is consistent with the ISA-95 standard, it is capable of providing persistent information about production recipes and production constraints, regarding a particular manufacturing process implemented by a production line. As such, it could be exploited to formalize a set of production requirements and to verify that such a set can be implemented onto a manufacturing system digital model, *e.g.*, the digital twin.

The design flow we are proposing is capable of handling MES data to specify a set of A/G contracts defining the required production. The design problem is specified also with regards to a set of production constraints, derived from production models or from machine's limited capabilities. Such constraints express, as an example, the availability of the required machine or the presence of an operator acting on the manufacturing component. In this sense, the B2MML structure details multiple XML elements expressing such an information set [25]. A number of constructs are offered by this standard to represent the set of resources necessary to implement a production process:

- *EquipmentInformation*, data on manufacturing machines and their availability.
- *MaterialInformation*, the needed material to implement the production process.
- *PersonnelInformation*, employees associated with a particular machine or manufacturing step.

In addition, production requirements are expressed as production recipes and implemented through the *OperationDefinition* structure. Each OperationDefinition is composed of a set of *OperationsSegments*, which represents the manufacturing step as a work-unit, characterized by the required materials, personnel, and pieces of equipment to be successfully accomplished. Multiple attributes further define the sub-process parameters, such as step duration and dependencies to other previous or next sub-processes.

Regarding the contracts hierarchy depicted in Figure 3. The production recipe is defined into the first two levels. The first level gives a complete overview of the manufacturing process, representing the entire production over the entire set of machines on the production floor. The complexity involved in the production recipe contract-based specification is tightly bound to the number of manufacturing steps, constraints and parameters. For this reason, we propose a specification problem decomposition, represented by the second level of the hierarchy: the "*Machine Production*" contracts. The decomposition individually considers the recipe's required production of each machine, splitting constraints and requirements in multiple contracts-based specifications. This decomposition approach is able to reduce the complexity involved in consistency checking and synthesis procedures of contracts. The top hierarchy level is still needed, to guide the execution flow of the recipe between different machines, and thus specifying LTL properties such as:

$$\Box(steps = 0) \rightarrow (segment_0 \wedge ConveyorActive)$$
$$\Box(steps = 1) \rightarrow (segment_1 \wedge 3DPrinterActive)$$
$$...$$

where each step variable assignment define an OperationsSegment and involves the usage of a certain equipment.

In the second level, each OperationsSegment of a manufacturing component is systematically collected, with its constraints and its recipe-specific dependencies. Constraints such as the material availability and the required employee are specified through a set of assumptions on the A/G contract. Regarding the case study described in Section II-D, it is assumed that the 3D printer status is not "maintenance", and thus the machine is available for the production. Moreover, the contract assumes that, to produce the specified piece in the OperationsSegment, the quantity of extruded plastic is 50gr, and that such a quantity is available. This assumed property can be specified in LTL as follows:

$$\Box((\neg maintenance \wedge material\_quantity > 50) \rightarrow \bigcirc(printPiece\_1))$$

The *printPiece_1* variable specifically refers directly to an OperationsSegment structure of the recipe, which describes the printing of the the plastic piece that will be assembled to the other brick present in the warehouse. The machine's status (*e.g.*, maintenance) evolution, as well as the fulfillment of the necessary production material (*e.g.* plastic filament), are modeled in lower-level contracts and are only assumed at this point. A guarantee specifies that the necessary quantity of material is consumed at the end of the manufacturing step:

$$\Box((printingCompleted) \rightarrow (material\_quantity = material\_quantity - 50))$$

Furthermore, the production contract has to guarantee that eventually the plastic brick is produced. This behavior is specified guaranteeing that when the "printingCompleted" signal is true, the OperationsSegment is over:

$$\Box((printingCompleted) \rightarrow \bigcirc(segmentOver_n))$$

n is an index identifying an exact OperationsSegment. Multiple segments could be associated with a single machine during the entire production. If that's the case, the "Machine Production" contract needs to model the dependencies of such OperationsSegment set, which can be different from global production dependencies. As an example, the QC station has to check for defects on the 3D printed piece first and then has to verify that the final product has been correctly assembled. As such, two different segments are implemented through the QC system and, thus, the QC "Machine Production" contract specifies two distinct manufacturing steps. For such reason, the contract has to guarantee the order of segments, *e.g.*, the first check must be on the printed piece ($segment_n$) and the second on the assembled product ($segment_{n+1}$):

$$\Box((segmentOver_n) \rightarrow \bigcirc(segmentOver_{n+1}))$$

The same structure is used to model dependencies between different machine's production contracts, assuming the segments existence but not considering any of their constraints.

V. Plant Specification and Digital Twin Generation

To specify the lower control levels over manufacturing components, additional data is needed. The information structure provided by B2MML can be integrated into AML, which contains each XML field necessary to represent a B2MML data model [26]. In particular, the *InstanceHierarchy* AML structure is the top-level view over the production line specification. In addition to production concepts, the B2MML *Equipment Information* can be extended to model the plant topology: each machine equipment XML block has an interface connector, depicting the set of closest and reachable components, due to transportation system's limits. As such, it expresses the guarantee that the production recipe is consistent with the actual plant structure. With regards to the case study considered in Section II-D, the material transportation system is implemented through a set of conveyor belts. Consequently, the set of topology constraints is negligible, since a work-material is able to reach a destination machine from every source machine. Nonetheless, a more constrained plant structure can be represented by applying the outlined principles.

Every functional behavior of a machine is described using the DIN 8580 standard. Specifically, the modeling (composition and detail) of the behaviors are selected from the standard, which describes the set of elementary production actions of a machine. This information is integrated into the AML structure, with details on the action's environment and attributes (*e.g.*, time constraints). In fact, each machine action is represented as a *RoleClass* element. Regarding the hierarchy depicted in Figure 3, levels labeled with "*Machine Contracts*" specifically define a set of contracts on elementary actions and on the execution flow of such actions' set. More specifically, the "*Machine Coordinator*" represents of constraints of the entire set of elementary actions. A group of variables is used to link multiple actions together (*e.g.*, the change of tool of the milling machine) and to model dependencies between actions (*e.g.*, pick before place for a manipulator arm). On the other hand, "*Elementary Action*" contracts specify the behavior of the action on the workpiece. The action's contract-based specification assumes a discretized grid representing the space and also a discretized shape of the work-material. The objective of the contract is to specify the effects of the manufacturing process on the material, that has to be guaranteed if assumptions hold.

For example, the "*"Robotic Assembly*" station, consisting in two collaborative manipulator arms, is specified by multiple elementary actions gathered from the Deutsches Institut für Normung (DIN) standard:

- *Compose*: both manipulators have to pick the two pieces and then synchronize at a fixed point in space, to assemble the final product
- *Decompose*: one arm picks the composed piece and then both hands are positioned in the cooperation point to decompose the product

Table I: Time required to consistency check and synthesize the set of A/G contract of the production line hierarchy: (1) 3D Printer, (2) Conveyor Belts, (3) Quality Check, (4) Robotic Assembly, (5) Milling Machine, (6) Production Requirements of the entire system.

	(1)	**(2)**	**(3)**	**(4)**	**(5)**	**(6)**
Consistency Checking & Synthesis (s)	0.25	0.67	2.58	25.51	0.48	0.04
Code generation (s)	0.07	0.20	0.76	3.69	0.20	0.06
Total Time (s)	0.32	0.87	3.34	29.21	0.68	0.10

- *Turn*: turn the piece to a specific side.
- *Move*: place the piece in a specific position.

The coordinator's contract guarantees that the composed and decomposed actions are not carried out at the same time. Moreover, the specification forces that the piece is taken by the nearest manipulator. It is also initially assumed that two pieces are needed for the composition action to be executed. Each action is detailed with a "*Elementary action*" dedicated contract-based specification, that describes the change of the shape of the work-piece. This contract abstracts the space and the material with multiple grids of different granularity, depending on the assumed environment for such action:

- *Side Grid*: one cell represents one side of the piece.
- *2D Grid*: one cell represents one 2D portion of the piece.
- *3D Grid*: the work-piece is described by a 3D grid where each cell contains a 3D sub-portion of the material.

As an example, to describe the elementary "Turn" action related to the manipulator arm, the piece is represented with a "Side grid" of only one dimension, since a higher granularity is not needed. In other actions, such as the printing action of the 3D printer, the piece is considered as a parallelepipedon, so it is modeled with a 3D Grid. Moreover, the 2D grid is mainly used to represent the conveyor belts set, in order to model the movement of the material throughout the production line. From realizable contracts, multiple tools such as *Slugs* [23] exploit reactive synthesis algorithms, able to construct implementations consistent to the initial contract-based specification. Each implementation is then translated to executable code and integrated into the plant simulator environment, to obtain the digital twin model.

VI. Experimental Results

The validation of the formalized production recipe passes through the generation of the production plant digital twin, that we use to verify whether the production recipe is consistent with the structure and functionality of the manufacturing system. Furthermore, the simulator on which the digital twin simulation is set up can provide insights about production data (*e.g.*, produced pieces per timeframe) and about the extra-functional properties of the system (*e.g.*, power consumption).

The methodology is tool-independent and can be used to generate implementations compatible with many state-of-the-practice plant simulation software. The only requirement is that the tool of choice is capable of importing pure C/C++ code. Figure 4 depicts the case study simulation environment that we use as a demonstrator: Technomatix Plant Simulation [27]. This software provides a production discrete-event environment and has been specifically developed to

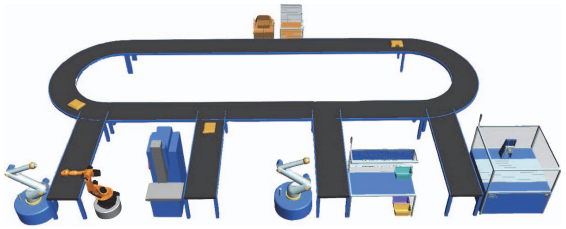

Figure 4: ICE case study representation into the Technomatix Plant Simulation environment.

carry out analysis about production plants. The elementary unit in Plant Simulation is a "Station", which represents a simple machine. Plant Simulation provides multiple interfaces to various languages, to ease the integration of external source code that defines each station's control logic.

Table I depicts the main results achieved. It reports the time required by consistency checking, synthesis and code generation procedures of each A/G contract-based specification associated with all machines, both at recipe and machine level. As an example, the total time referred to the 3D printer (column (**1**)) is composed by Machine Production contract (defining the piece that has to be printed), the Machine Coordinator (handling the elementary action's flow to produce the product) and a set of Elementary Action contracts (*i.e.*, Extrude Plastic and Change Layer). Column **(6)** refers to the first level of the hierarchy, that is independent of the specifications of machines.

A. Production Recipe Validation

Plant Simulation refers to a produced product as *MU*, which is an an abstraction of the shape and attributes over the workpiece. MU properties can be user-defined and their evolution can be modeled throughout the production. To validate the formalized production recipe, the simulation demonstrates that both the product can be produced and also that the product properties (*e.g.*, weight) are consistent with the specification.

B. Extra-functional Property Analysis

The simulation is also capable of providing interesting functional data on production throughput and extra-functional properties such as power consumption. Figure 5 depicts the power consumption of each machine in 4 hours and 30 mins of simulated production. In such time, the production line produced 8 pieces and consumed a total of 42.7kWh. The 3D printer is both the bottleneck of the production, requiring 30 minutes to produce a single plastic brick, and the manufacturing component that has consumed the most energy.

VII. Conclusions

This paper proposes a production recipe validation strategy based on the generation of digital twins and the simulation of such a system model. A central role is played by a hierarchy of A/G contracts, that specify multiple level behaviors, from plain production's actions to control strategies over the manufacturing machine. We applied this methodology on a real-world case study, showing its applicability on a concrete production line and obtaining also preliminary information about some extra-functional features of the system.

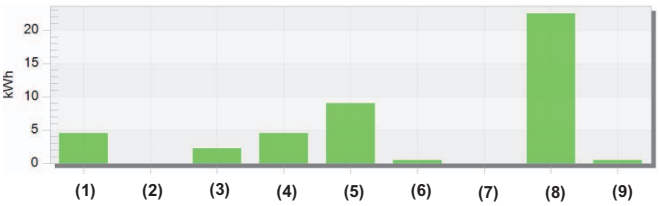

Figure 5: Power consumption of each machine regarding the production case study: (1) Conveyor to Robotic Assembly, (2) Conveyor to Milling Machine, (3) Conveyor to 3D Printer, (4) Conveyor to QC, (5) Main Conveyor Belt, (6) Robotic Assembly System, (7) Milling Machine, (8) 3D Printer, (9) QC System.

References

[1] R. Drath *et al.*, "Industrie 4.0: Hit or hype? [industry forum]," *IEEE Industrial Electronics Magazine*, vol. 8, no. 2, pp. 56–58, jun 2014.

[2] J. Vachalek *et al.*, "The Digital Twin of an Industrial Production Line within the Industry 4.0 Concept," in *Proc. of IEEE PC*, jun 2017.

[3] D. Mourtzis *et al.*, "Simulation in manufacturing: Review and challenges," *Procedia CIRP*, vol. 25, pp. 213–229, 2014.

[4] AutomationML Consortium, "AutomationML," 2006. [Online]. Available: https://www.automationml.org/o.red.c/home.html

[5] International Society of Automation, "ISA-95 Standard," 2000. [Online]. Available: https://www.isa.org/

[6] A. Sangiovanni-Vincentelli, "Corsi e Ricorsi: The EDA Story," *IEEE Solid-State Circuits Magazine*, vol. 2, no. 3, pp. 6–25, 2010.

[7] R. Chirico *et al.*, "A Contract-based Methodology for Production Lines Validation," in *Proc. of IEEE INDIN 2019*, pp. 1–4.

[8] F. Fummi *et al.*, "Moving from co-simulation to simulation for effective smart systems design," in *Proc. of IEEE/ACM DATE*. European Design and Automation Association, 2014, p. 286.

[9] O. Ljungkrantz *et al.*, "Formal Specification and Verification of Industrial Control Logic Components," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 3, pp. 538–548, July 2010.

[10] S. Spellini *et al.*, "Work-in-Progress: Introducing Assume-Guarantee Contracts for Verifying Robotic Applications," in *Proc. of CODES+ISSS*, 2018, pp. 1–2.

[11] S. Spellini *et al.*, "Compositional Design of Multi-Robot Systems Control Software on ROS," *ACM Transactions on Embededded Computing Systems*, vol. 18, no. 5s, pp. 71:1–71:24, 2019.

[12] D. Brandl, "What is ISA-95? Industrial Best Practices of Manufacturing Information Technologies with ISA-95 Models," 5 2008. [Online]. Available: http://www.apsom.org/docs/T061_isa95-04.pdf

[13] R. Drath, "Let's talk AutomationML what is the effort of AutomationML programming?" in *Proc. of IEEE ETFA*, Sep. 2012, pp. 1–8.

[14] O. Kovalenko *et al.*, "AutomationML Ontology: Modeling Cyber-Physical Systems for Industry 4.0," *IOS Press Journal*, 2018.

[15] B. Wally, "Application Recommendation Provisioning for MES and ERP – Support for IEC 62264 and B2MML," 2018.

[16] MESA International, "B2MML," 2010. [Online]. Available: http://www.mesa.org/en/B2MML.asp

[17] A. Benveniste *et al.*, "Contracts for System Design," *Foundations and Trends in Electronic Design Automation*, vol. 12, no. 2-3, pp. 124–400, 2018.

[18] P. Nuzzo *et al.*, "A platform-based design methodology with contracts and related tools for the design of cyber-physical systems," *Proceedings of the IEEE*, vol. 103, no. 11, pp. 2104–2132, 2015.

[19] P. Nuzzo *et al.*, "CHASE: Contract-based requirement engineering for cyber-physical system design," in *Proc. of IEEE/ACM DATE*, 2018, pp. 839–844.

[20] R. Rosner, "Modular Synthesis of Reactive Systems," *PhD thesis. The Weizmann Institute of Science*, 1991.

[21] N. Piterman *et al.*, "Synthesis of reactive (1) designs," in *Proc. of ACM VMCAI*. Springer, 2006, pp. 364–380.

[22] I. Filippidis *et al.*, "Control Design for Hybrid Systems with TuLiP: The Temporal Logic Planning Toolbox," in *Proc. of IEEE CCA*, 2016, pp. 1030–1041.

[23] R. Ehlers *et al.*, "Slugs: Extensible GR(1) Synthesis," in *Proc. of CAV 2016*. Springer, 2016, pp. 333–339.

[24] M. Lora *et al.*, "Translation, Abstraction and Integration for Effective Smart System Design," *IEEE Transactions on Computers*, 2019.

[25] I. Harjunkoski *et al.*, "Sharing Data for Production Scheduling Using the ISA-95 Standard," *Frontiers in Energy Research*, vol. 2, 10 2014.

[26] B. Wally, "Application Recommendation Provisioning for MES and ERP - Support for IEC 62264 and B2MML," 2018.

[27] Siemens, "Tecnomatix Plant Simulation," 2017.