From Multi-Level to Abstract-Based Simulation of a Production Line

Stefano Centomo, Enrico Fraccaroli, Marco Panato
Department of Computer Science – University of Verona (Italy)
name.surname@univr.it

***Abstract*—This paper proposes two approaches for the integration of cyber-physical systems in a production line in order to obtain predictions concerning the actual production, core operation in the context of Industry 4.0. The first approach relies on the Multi-Level paradigm where multiple descriptions of the same CPS are modeled with different levels of details. Then, the models are switched at runtime. The second approach relies on abstraction techniques of CPS maintaining a certain levels of details. The two approaches are validated and compared with a real use case scenario to identify the most effective simulation strategy.**

I. Introduction

The concept of Industry 4.0 [1] represents an innovative vision of what will be the factory of the future. The principles of this new paradigm are based on interoperability and data exchange between different industrial equipment. In this context, Cyber-Physical Systems (CPSs) cover one of the main roles in this revolution. The entire factory can be seen as a set of CPSs and the resulting system is also called Cyber-Physical Production System (CPPS). This CPPS represents the Digital Twin of the Factory with which it would be possible to make analysis regarding the Real Factory [2]. The interoperability between the real industrial equipment and the Digital Twin [3] allows to make predictions concerning the quality of the products. Several tools [4] allow to model a production line, considering different aspects of the factory (*i.e.* geometrical properties, the information flows, *etc.*). However, these simulators do not provide natively any solution for the design integration of CPSs, making impossible to have precise analysis concerning the real factory.

This paper proposes two different approaches for the integration of CPS in a production line simulator (see Fig. 1). The approach on the left side of the figure relies on the *Multi-Level* simulation where multiple descriptions of the same CPS are managed. These descriptions have different levels of detail and are switched at runtime in order to optimize the overall simulation time. The second approach is based on *Abstraction* techniques where a set of manipulations are made on the whole system in order to obtain semantically equivalent descriptions but reducing the complexity of the starting models. Different coordination strategies are presented in order to integrate and simulate correctly the abstracted models. The two approaches are then integrated with *Siemens Plant Simulation* with a real use case scenario. The obtained results show the benefits of the CPS integration in both of the approaches.

II. Background

In these years several providers proposed different tools to model manufacturing processes.Report [4] summarizes periodically all tools by proposing comparisons on their main characteristics (usability, costs, features, *etc.*). This report is a useful guideline for selecting most suited plant simulators with respect to the parameters to evaluate. A production line

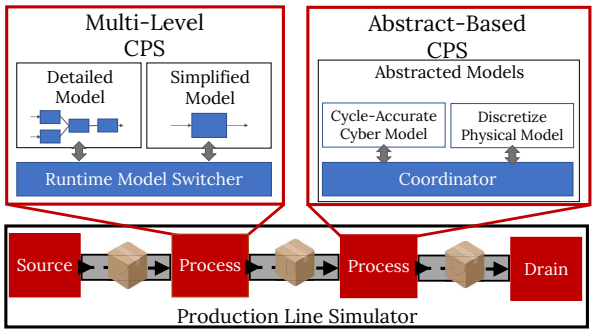

Figure 1: *Multi-level* and *Abstract-Based* approaches for the CPS integration in a production line simulator.

is the composition of processes, organized into a chain, which has the main purpose of handling information. Despite some differences, all the simulations share the following principles:

- *Layout Planning*: Represents the geometrical structure of the production line. A library of components allows to model the factory, by considering physical constraints.
- *Material Flow/Fluid Simulation*: Represents the movements of products from a process to the others. This is made possible with components like line transporters or pipe, depending on the material state of matter, *i.e.*, solid or fluid.
- *Process Simulation*: Represents the physical transformation made by the equipment of the factory to the products.

From the simulation perspective, most of the available simulators rely on the *discrete-event* model of computation. This paper makes use of Siemens *Plant Simulation*, a simulator which over the years has become a standard de facto in the designing of production lines. It is a Model-Based tool that provides a library of customizable components that represent the basic building blocks of a factory. Combination of these blocks allow to model different aspects of a production chain. The products are called Mobile Unit (MU) and they represent the entities moving among the blocks of the production line.

SingleProcess is the fundamental block provided by the tool used to represents the physical process of an equipment on a MU. Plant Simulation offers the possibility to customize the behaviour of every block with an internal programming language called *SimTalk*. It also provides a functionality called *C-interface* to import dynamic libraries written in C/C++, enabling to customize the behaviour of the simulation and also connect other tools.

III. Multi-Level Modeling and Simulation

Multi-level approach [5] allows managing multiple descriptions of the same system with different level of detail. Mixing the execution of these descriptions at runtime allows to speedup the simulation instead of using only the high

978-3-9819263-2-3/DATE19/©2019 EDAA

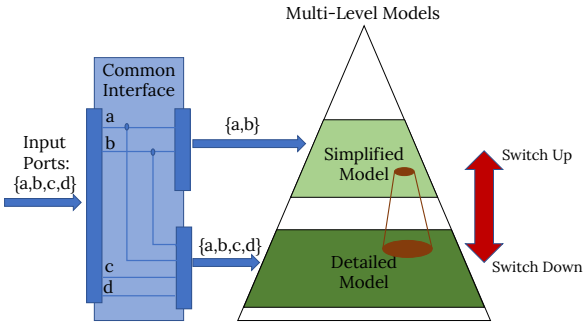

Figure 2: Multi-level Common Interface and switching actions

resolution model, but maintaining a certain level of precision compared to the low accuracy of the low resolution model. Multi-level approach requires to face some issues that can be summaries as follows:

- Interfaces of the multi-level models;
- Models switching actions;
- State mapping of the models.

Let consider two different behavioural descriptions of a CPS: a *Detailed Model* and a *Simplified Model*. These two models could have different interfaces (see Fig. 2). The *Simplified Model* has fewer ports than the *Detailed Model*, but all the ports of this model are a subset of the *Detailed Model* interface. The standardization of a `Common Interface` is needed to easily switch between the two models (see Fig. 2).

The exposed ports of this block are represented by the interface of the *Detailed Model*, in order to capture all the properties without losing any information. When the *Detailed* model is in use, the `Common Interface` block receives inputs `a,b,c,d` and redirects all the four inputs to the model. When *Simplified Model* is running, the `Common Interface` block redirects only `a,b`. The second issue to face with Multi-Level approach regards the switching actions used to switch from a model to another. *Switching-up* action allows switching from a high-resolution model to another with a lower resolution. *Switching-down* action switch from a low-resolution model to a high-resolution model.

Every switching action requires to exchange the internal state of the current model to the new switch one, in order to be consistent between the models. This is called *state mapping* [6]. For instance, let consider the *Detailed Model* simulated for a certain amount from $t=0$ to $t=h$. At time $t=h$ a *switching-up* action is triggered. Without *state mapping*, the simulated time of the *Simplified Model* is zero but the global time of the simulation is h. The internal status of the *Simplified Model* refers to its initial conditions and not to the time h, making an inaccurate simulation. In this paper all the considered models expose internal storing variables as ports to the `Common Interface` blocks, in order to be able to perform a state mapping.

A. Application of Multi-Level Simulation to a Production Line

The application of Multi-level approach with *Plant Simulation* requires to define synchronization rules in order to handle correctly the data from the models to the production Line simulator and vice-versa.

The `Runtime Model Switcher` is the actor in charge to exchange the data and manage the simulation of the two models (see Fig. 3). The *Common Interface* block is part of the `Runtime Model Switcher`. The `Runtime Model Switcher` receives input data from *Plant Simulator* and then redirect it to the `Common Interface`. After the data exchange, the `Runtime Model Switcher` select the model to simulate performing a switching action. The `Runtime Model Switcher` also stores the shared properties of the models, needed to perform the state mapping. In *Plant Simulation* we assume that the switching actions, can be performed when a `MU` enters in a `Single Process` block that represents an equipment of the factory. With this assumption, we say that the resolution of the switching points is Mobile Unit Accurate (MU-ACCURATE).

IV. ABSTRACT-BASED MODELING AND SIMULATION

A CPS can be modelled with a low level of details in a unique language but the resulting system will be not accurate. Refine both Cyber and Physical systems requires specific languages of different tools tailored to specific domains that use different Models of Computation (MOC). The simulation of the Cyber and the Physical systems together is time-consuming, because of the synchronization of the different MOC and because of Co-Simulation mechanisms between different tools. The goal of the *Abstract-Based* approach is to abstract the descriptions of the two subsystems in order to reduce the global complexity of the whole Systems. The abstract-Based approach starts from specific-domain languages like Hardware Description Languages (HDLs) for the Cyber part and Verilog-AMS to model the Physical System, that uses a Continuous Time MoC. This paper relies on abstraction technologies provided by a HIFSuite framework [7]. Such a framework allows to manipulate models described using HDLs (Verilog and VHDL) or Analog Mixed Signal (Verilog-AMS). HIF provides a set of methods to import model descriptions, manipulate them and generate semantic equivalent C++ code. The front-end methods translate the model descriptions in a proprietary language (HIF format). The manipulation tools make transformation on the obtained HIF description. Finally, the back-end methods allow generating the C++ code. There are three different abstractions available: *Analog Abstraction*, *Datatype Abstraction* and *Protocol Abstraction*.

The *Analog Abstraction* is a process which aims at simplifying continuous time components for an easier integration and efficient simulation [8]. The challenge is related to their behaviour, which is usually described through a system of differential equations. State of the art simulators represent the behaviour as sparse matrices and solve the system at each simulation step, with a consequent loss of performances. With the abstraction, the complexity of solving such systems is anticipated during models generation. It enriches the initial set of equations using Kirchhoff's laws. Then, it solves the resulting enriched system with a symbolic solver and produces a signal-flow representation of the model. The final description contains the simplified equations that describe the relation between inputs and the outputs of the model.

The *Datatype Abstraction* is a process which transforms HDL datatypes, like logics and bit vectors with a many-valued logic, into efficient C++ native ones [9].

The *Protocol Abstraction* is performed on the simulation protocol of the model [10]. This manipulation reduces the internal processes of the module comparing them sensitivity lists and merging them together. Furthermore, it encapsulates an optimized Discrete-Event scheduler inside the C++ final code. The resulting C++ code contains a special structure that represents the interface of the starting model and a set of methods to simulate it.

A. Application of Abstract-Based Simulation to Production Line

The Cyber and Physical Abstracted models need to be integrated into *Plant Simulation*. First, a simulation coordinator

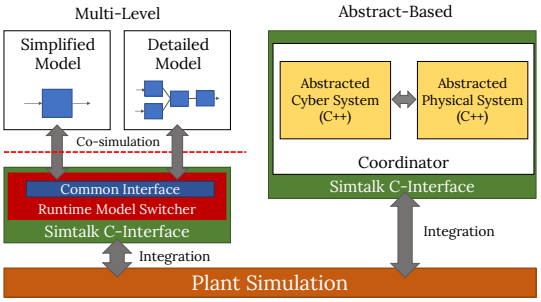

Figure 3: Overview of the Experiment Setup.

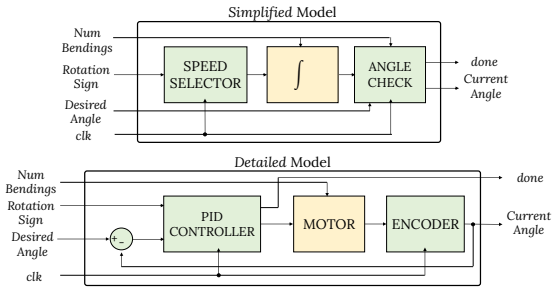

Figure 4: Simplifed and Detailed Physical Models.

is needed to synchronize the two abstracted systems of the CPS. This paper presents two different coordination strategies. The first strategy is the *Cycle-Accurate* where both of the Systems are simulated with a timestep that is equal to the clock period of the Cyber System. This coordination requires a lot of synchronization points depending on the clock period.

The second is the *Transaction-Level* strategy, and it is event-oriented. For instance, if the Cyber model is blocked, waiting for a certain event from the Physical system, the coordinator will not execute it. This solution is still correct, but requiring fewer synchronization points than the *Cycle-Accurate*. Moreover, it requires to know in advance the synchronization mechanisms between the two parts of the CPS in order to identify which synchronization points can be avoided. The resulting CPS is then integrated in *Plant Simulation* using *SimTalk C-Interface* APIs.

V. Experimental Results

The two presented approaches are tested with a real use case scenario of a simple production line. The production line is composed of three processes and represents a bending operation of metal sheets. The first process applies to the metal sheet a barcode containing the information of the desired bend angle. The second represents the bending machine that reads the angle to bend from the barcode and executes the bending operation. The real bending process of the equipment requires at most 3 seconds to bend a metal sheet. The last process checks the quality of the bending operation comparing the desired angle and the real bent metal sheet. The two proposed approaches presented in this paper integrate a CPS in the bending process in order to make a more accurate estimation of the production in terms of productivity and quality(see Fig. 3). The *Physical System* represents the behaviour of a bending machine, described using Verilog-AMS, that is controlled by the *Cyber System*. The *Cyber System* of the CPS is a Hardware platform composed of a CPU, a Memory, a Bus, a Barcode sensor and an Actuator to the bender physical system. All the components of the HW platform are described by using HDLs.

The bending software reads the angle to bend from the metal sheet using the Barcode Sensor, and then redirects it to the *Physical Systems* with a set of commands.

A. Multi-Level Experiment

The Multi-Level experiment relies on two different versions of the CPS called *Simplified Model* and *Detailed Model* (see Fig. 4). In both of the CPS models, the *Cyber System* is represented with the digital platform explained in the previous section. In the *Simplified Model* the *Physical System* consists of a behavioural description of the bender equipment, with a low level of detail. In particular, the *Physical System* is modelled with an integrator that describes the bending operation. The integrator receives as input a value that represents the constant bending speed to apply. This value is calculated by a *Speed Selector* node, which adopts two different values according to the bending rotation versus. The value provided by the integrator is then compared with the desired angle by the *Angle Controller* node in order to drive the *done* signal. The *numBendings* is used to model the machinery wearing.

In the *Detailed Model* the *Physical System* allows to make a precise estimation of the execution time and the quality of operation. A *PID Controller* is used to control a DC Motor in order to bring every sheet to the desired angle, which is read by an encoder that translates the motor rotational position in a digital value. The DC Motor model uses the *numBendings* value to modify its internal parameters: this allows to simulate its wearing since those values represent the mechanical and electrical characteristics of the Motor.

The `Runtime Model Switcher` is connected to the CPS simulator using Co-simulation techniques. In the other side, the `Runtime Model Switcher` is integrated in *Plant Simulation* using *C-Interface* proprietary interface. The `Runtime Model Switcher` is then linked to the Bending `SingleProcess` and executed only when a new MU enters in that node of the production line. The simulation starts using the *Simplified Model* of the CPS. Every 20 bending operations, the `Runtime Model Switcher` enables both the models and provides to them the current metal sheet to bend. After the bending process, the quality deviation between the two models is evaluated: if it crosses a certain threshold, the `Runtime Model Switcher` deactivates the *Simplified Model* and switches to the *Detailed Model*. The *Detailed Model* is kept enabled until the quality deviation remains over the threshold. The simulation resolution is MU-Accurate, meaning that the switching actions can be performed only at the entrance of a MU in the `SingleProcess`. When the execution of the bending operation is completed, the `Runtime Model Switcher` retrieves the executed time to bend the metal sheet and returns it to the `SingleProcess` of Plant Simulation.

B. Abstract-based Experiment

To better evaluate the *Abstract-based* approach performances, the two CPS versions (*Simplified Model* and *Detailed Model*) are considered. As explained in IV-A the entire abstraction toolchain is based on the HIFSuite framework [7]. The models of the Cyber and Physical systems, are first translated in HIF format, using front-end methods. After the first translation, the *Physical System* is abstracted using the *Analog Abstraction*, defining the time step to use for the discretization. Then, *Datatype* and *Protocol Abstraction* are performed on both the obtained HIF descriptions. Finally, the HIF back-end tools are invoked to generate the C++ code. For the *Cyber System* the *Protocol Abstraction* is performed with Cycle-Accurate (CA) resolution. More in details, when the

Table I: Simulation times of bending operations for Mobile Units in the two different approaches.

Simulated Bending Operations	Multi-Level			Abstract-based				Simulated Time	
	Simplified model	Detailed model	Hybrid	Abstracted Simplified Model		Abstracted Detailed Model		Simplified Model	Detailed Model
				Cycle-Accurate Coordinator	Transaction-Level Coordinator	Cycle-Accurate Coordinator	Transaction-Level Coordinator		
1	0.22	0.63	0.22	0.208	0.016	0.075	0.008	1.22	0.66
20	5.28	15.12	9.74	4.992	0.384	1.801	0.192	16.70	11.46
50	12.76	36.54	27.69	12.064	0.928	4.351	0.464	39.28	27.67

Table II: Times needed to simulate one second in the two different approaches.

Average Time to Simulate 1 Second (s)	Multi-Level			Abstract-based			
	Simplified model	Detailed model	Hybrid	Abstracted Simplified Model		Abstracted Detailed Model	
				Cycle-Accurate Coordinator	Transaction-Level Coordinator	Cycle-Accurate Coordinator	Transaction-Level Coordinator
	0.180	0.955	0.437	0.170	0.013	0.114	0.012

simulation method of the *Cyber System* is called, it performs a simulation of an entire clock cycle. A *Coordinator* is needed to simulate together the obtained abstracted Systems that compose the CPS. This paper proposes two different coordinators, *Cycle Accurate Coordinator* and *Transaction-Level Coordinator*. The *Cycle Accurate Coordinator* simulates and exchanges the data between the Cyber and the Physical System at every clock cycle period. The *Transaction-Level Coordinator* reduces the number of communication points based on the communication protocol between the two systems. When the *Physical System* is performing a bending operation, the *Cyber System* has to wait until the finishing operation event is triggered. Thus, the *Transaction-Level Coordinator* can pause the *Cyber System* and simulate only the *Physical System* until this event. Finally, the resulting C/C++ code of the CPS is wrapped with the *C-interface* avoiding completely Co-Simulation mechanisms.

C. Simulation Results Comparison

In Table I and II are reported the results of the experiments between the two approaches.

Table I reports the required time to simulate the metal sheet bending using the different models. The last two columns report the simulated times. The simulated time of the *Detailed Model* is more accurate than the *Simplified Model*. In *Multi-Level* approach, the *Hybrid* column reports the simulation results mixing the *Simplified* and *Detailed Models*. At the beginning of the simulation, *Simplified Model* is activated, in fact, it requires similar simulation times. When the number of bending operations increases, the required time moves towards the *Detailed Model* one, which is enabled to keep the quality deviation estimation under a certain threshold. This approach requires an effort by the CPS simulator that has to simulate every single clock cycle of the HW platform, in order to obtain the most precise simulation. It is also slowed down by Co-Simulation mechanisms. The *Abstracted Simplified Model* with *Cycle-Accurate* coordinator, requires almost the same time as the *Simplified Model* with *Multi-Level* approach. This is due to the very small complexity of the *Simplified Model* that cannot be reduced from the abstraction techniques. The *Abstracted Detailed Model* with the same coordinator achieves a speed-up of 10x compared to the *Detailed Model* of the *Multi-Level* approach. However, the *Cycle-Accurate* coordinator simulates the busy waiting of the *Cyber System* during the physical bending introducing unnecessary communication points. The *Transaction-Level Coordinator* still maintains the same timestep precision but avoiding all the unnecessary communication points. The *Abstracted Simplified Model* with *Transaction-Level Coordinator* achieves a speed-up of $\sim 10x$ than the *Cycle-Accurate* coordinator simulation. The *Abstracted Detailed Model* obtains a $\sim 100x$ speed-up than the *Multi-Level Detailed Model* and $\sim 10x$ as against its *Cycle-Accurate* version. In general, using the *Transaction-Level* coordination algorithm allows reducing simulation times of one degree of magnitude. From Table I it could seem that the *Abstracted Simplified Model* is slower than the *Abstracted Detailed Model*.

In Table II is reported the average time needed to simulate one second for the different approaches. In The *Multi-Level* approach, the *Detailed Model* is 5x slower than the *Simplified Model*. The *Abstracted Simplified* and *Abstracted Detailed Model* with the *Transaction-Level coordinator* requires almost the same average time to simulate one second.

VI. Concluding Remarks

In this work, we proposed two different approaches to integrate CPSs in a production line simulator. The *Multi-Level* approach has shown the benefits of switching between different models at runtime has been proved with the obtained results, but with an approximate simulation. The *Abstract-based* represents a promising alternative solution that abstract models through a set of abstraction steps, but maintaining a certain level of detail. The *Abstract-Based* approach with *Transaction-Level* coordination has proved to be the best choice in terms of simulation time.

The results clearly show that the *Abstracted Simplified* and *Abstracted Detailed* models require almost the same time to simulate, but with different simulation accuracy. This important result allows to consider *Abstracted Detailed Model* with the *Transaction-Level* coordinator as the best solution avoiding approximate simulation of the *Multi-Level* approach or inaccuracy of the *Abstracted Simplified Model*.

References

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

[2] Centomo *et al.*, "Cyber-physical systems integration in a production line simulator," in *2018 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*. IEEE, oct 2018.

[3] Vachalek *et al.*, "The digital twin of an industrial production line within the industry 4.0 concept," in *2017 21st International Conference on Process Control (PC)*. IEEE, jun 2017.

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

[5] Reynolds *et al.*, "Consistency maintenance in multiresolution simulation," *ACM Transactions on Modeling and Computer Simulation*, vol. 7, no. 3, pp. 368–392, jul 1997.

[6] Huber and Dangelmaier, "A method for simulation state mapping between discrete event material flow models of different level of detail," in *2013 IEEE International Systems Conference (SysCon)*. IEEE, 2011, pp. 2877–2886.

[7] Bombieri *et al.*, "Hifsuite: tools for hdl code conversion and manipulation," *EURASIP Journal on Embedded Systems*, vol. 2010, no. 1, pp. 1–20, 2010.

[8] Lora *et al.*, "Analog models manipulation for effective integration in smart system virtual platforms," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 2, pp. 378–391, feb 2018.

[9] Vinco *et al.*, "Code Manipulation for Virtual Platform Integration," *IEEE Transactions on Computers*, vol. 65, no. 9, pp. 2694–2708, 2016.

[10] Bombieri *et al.*, "Automatic abstraction of RTL IPs into equivalent TLM descriptions," *IEEE Transactions on Computers*, vol. 60, no. 12, pp. 1730–1743, dec 2011.