A Hierarchical Modeling Approach to Improve Scheduling of Manufacturing Processes

Sebastiano Gaiardelli, Stefano Spellini, Michele Lora, Franco Fummi
Department of Computer Science
University of Verona, Italy
name.surname@univr.it

***Abstract*—Timely response to sudden production events and requirements shifts is a key feature of Industry 4.0. It requires techniques to manipulate and optimize the production processes, and components providing an high degree of reconfigurability.**

To acknowledge to such demands, this paper presents a multi-level and hierarchical approach to manufacturing processes modeling. Models are structured to represent the production hierarchically: partitioning recipes in a set of tasks, allocated to machines' manufacturing services and expressed as a sequence of elementary actions. Then, we propose a run-time scheduling algorithm able to exploit the novel structure given to knowledge by the proposed modeling approach. The algorithm aims at minimizing the makespan while maximizing machines utilization.

We validate the contributions of this paper on a full-fledged production line. The modeling strategy has been implemented in SysML: a well-known systems modeling language. The experiments show the presented model and the proposed scheduling approach enabling a more precise and more performing control over the manufacturing process.

***Index Terms*—Process modeling, process control, scheduling**

I. Introduction

Knowledge representation in production processes is a fundamental step for transitioning to "Smart Manufacturing" and Industry 4.0 [1], coping with modern market trends. Typically, a production process is executed by specialized machines, each having precise capabilities and characteristics. Therefore, creating a production process, commonly referred to as *Production Recipe*, means defining the set of production tasks, their dependencies, and the set of machines able to carry on the defined tasks. On a production line, different recipes may be executed in parallel and they may need to co-exist while guarantee a certain level of performance. Furthermore, to manage unexpected events efficiently, a production line must be able to interrupt, reallocate and reschedule.

Modern paradigms, such as on-demand production, require to restructure how the concept of recipe is being expressed and represented. This is especially true in Service-oriented Manufacturing [2] architectures, which organizes the work of the machines as a set of *services*. Each service carries out a specific machine's functionality and can be executed on-demand by the system's software architecture. Therefore, a tasks composing a recipe may be further characterized as a set of services. In addition, services may have one or more pre and post-conditions for their correct execution, as well as dependencies with other services. Thus, tasks may be implemented as a logical flow of services, with branches and cycles. Therefore, to correlate all those aspects and, possibly, the interaction with human operators, a structured representation of the production processes is needed [3].

This work is supported by the European Commission through the project DeFacto (grant n. H2020-MSCA-IF-894237), and by the Ministry of Education, University and Research "Dipartimenti di Eccellenza" 2018–2022 grant.

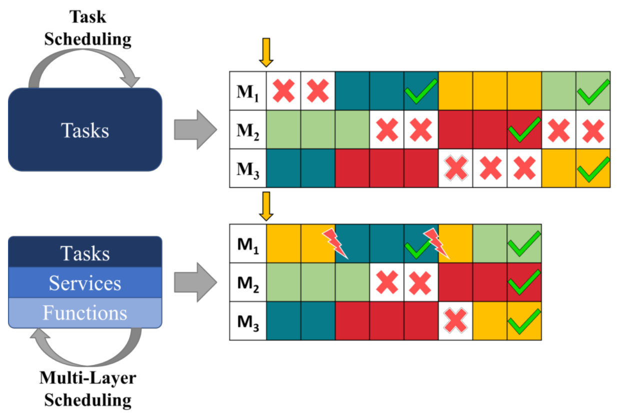

Figure 1: Overview of the proposed conjunction of hierarchical production model with a multi-layer scheduling algorithm. The knowledge encapsulated in the model enables advanced scheduling decisions (*e.g.*, interleaving). The proposed modeling and scheduling approach (bottom) allows achieving better performance than the classical task-based representation of production recipes (top).

Figure 1 depicts the contribution of this paper. Rather than relying on models for production processes based solely on the tasks composing a recipe (top part of the Figure), we propose a multi-level, hierarchical model for production processes which allows achieving better scheduling performance (bottom). The model is structured over three levels, each of them modeling a different abstraction of the production process:

- the *task level* consists of a task-resources graph. It allows describing the bones of the production process, which are the tasks, their dependencies and the machines on which such tasks can be allocated.
- The *service level* refines the concept of "task", describing the sequence of steps required to be carried out to complete the task. It consists of a directed graph where the edges express the execution flow and each node identify a step of the task.
- The *machine function level* describes the interactions that

need to take place between the control software and the machine implementing a service as a directed graph. Therefore, it allows exploiting the basic functionalities provided by the machine to create more complex services.

As each abstraction level carries different information, we also propose a scheduling algorithm exploiting the information structured in the three different abstraction levels of the proposed model. The algorithm takes advantage of the restructured information to make more precise decisions on whether a process can be interrupted, interleaved and preempted, thus potentially improving the performance of the production system. Furthermore, the scheduler is reactive: it is able to react to unforeseen events (*e.g.*, new high-priority orders or machine's failures) and reschedule the production.

We demonstrate the applicability of the proposed approach by modeling a production process on a full-fledged reconfigurable manufacturing system. The equipment in the production line is monitored and controlled by a software layer based on the OPC Unified Architecture (OPC UA). Machine capabilities and production processes have been modeled by following the approach proposed by this paper. We built the models using System Modeling Language (SysML) which provides an intuitive, standardized graphical language, as well as machine readable formats for data exchange. Thus, making more practical both modeling activities and the automation of the entire contribution. We evaluate the results of the proposed model-based scheduling algorithm. We show how exploiting hierarchically structured information allows increasing the average machine utilization and throughput, while minimizing the makespan, especially of high-priority orders.

II. State-of-the-art

In this Section, we analyze the state-of-the-art on scheduling techniques in manufacturing. Furthermore, we clarify the contribution proposed by this paper by evaluating models and standards currently used to describe production processes.

A. Production Scheduling

A good production scheduling is crucial to execute efficiently manufacturing processes. An optimal schedule allows increasing the productivity, maximizing throughput, and minimizing delay and interruption of production. In manufacturing, a schedule is an optimal allocation of *recipes* in a specific time-frame [4]. This problem is known as Job Shop Scheduling (JSS) [5] and Flexible Job Shop Scheduling (FJSS) [6]. In the literature, there are a plethora of solutions based on static techniques that guarantee optimal solutions. However, this is an approximation of the real problem, known as Dynamic Flexible Job Shop Scheduling (DFJSS) [7].

Practical scheduling problems are proved to be NP-hard [8] and, therefore, difficult to solve due to the number and variety of jobs and potentially conflicting goals. In the literature, there are many proposed solutions to solve the scheduling problems. Most of dynamic techniques are based on Artificial Intelligence methodologies, such as Ant Colony Optimization [9], Particle Swarm Optimization [10], Artificial Bee Colony [11] and Genetic Algorithms [12]. There are also other solutions falling into the static scheduling category, based on state-space search algorithms: Partial Order Planning [13], PERT Method [14], Mixed Integer Linear Programming (MILP) [15], and Constraint Satisfaction Programming (CSP) [16]. When manufacturing systems encounter unexpected conditions, such as machine breakdown, rush orders and process time delay, the schedule produced by the aforementioned techniques may no longer be optimal or may become infeasible. [17] proposes an hybrid approach: a static scheduling phase, exploiting MILP optimization, while a run-time scheduler is able to react to production changes and unexpected events. The ability of such a scheduling technique to efficiently react is limited to the amount of knowledge used by the algorithm to make decisions. Thus, it is limited by the models of the production processes and capabilities available to the scheduler.

B. Production Modeling

Scheduling techniques implement a particular optimization model, that is strictly related to both the parameter(s) to optimize and the constraints of the system. Two popular mathematical optimization models in such a context are Resource Task Networks (RTNs) [18] and State Task Networks (STNs) [19]. Both focus on formalizing the production recipes as a directed graph. The main difference between STN and RTN is that the former concentrates on expressing the sequence of material states associated to tasks, while the latter also explicitly includes the allocation of tasks and resources to physical machines. Therefore, the information on such models is limited to the recipe viewpoint, without any knowledge on how tasks are implemented by machines and, thus, without any information about control aspects. To structure recipes within manufacturing information systems (*i.e.*, Manufacturing Execution Systems (MESs)), different standards have been developed in the past. For example, International Society of Automation (ISA)-95 and 88 provide a consistent terminology to define basic components of a production recipe, such as tasks, materials and pieces of equipment. Furthermore, multiple XML-based languages have been developed to concretely express the concepts present in the standard. As an example, Business To Manufacturing Markup Language (B2MML) [20] and Automation Markup Language (AutomationML) [21] can be used to define not only the production process viewpoint, but also the topology and the architecture. Recent works [22] [23] proposed to unify the manufacturing system knowledge in SysML, also enabling models reuse. However, authors concentrated their efforts on architectural and control models and, therefore, did not propose a unified representation for production processes.

To the best of authors' knowledge, a model capable of representing the production in a hierarchical structure has not been proposed. In fact, the analyzed models and standards are focused on a single conceptual production level, such as the recipe level or the control automation level. Meanwhile, a unified model, able to capture at different abstraction levels, the different aspects of a production process, is still missing.

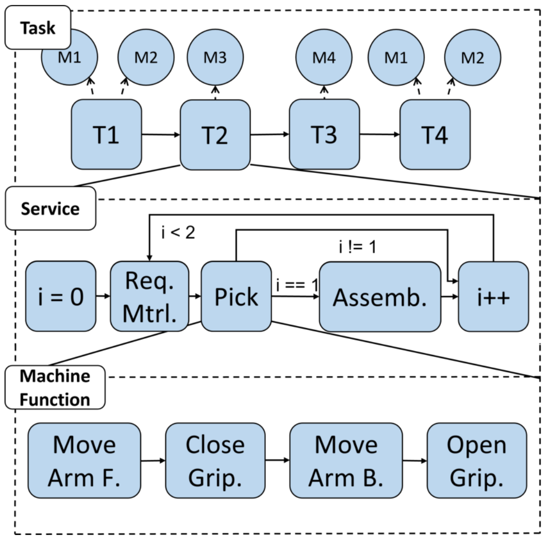

Figure 2: An example of the proposed three-layer representation. The first layer defines the set of tasks and their allocation onto machines. The second layer depicts "services" specified in a control flow graph implementing the task *T2*. The third layer outlines machines' functions implementing the *Pick* a service.

III. Hierarchical Process Model

In the following we present the proposed modeling strategy for production processes. The model is organized through three layers of knowledge, each depicting a particular abstraction perspective: the layers span from a high-level representation of production recipes as a set of tasks, to the sequence of functions that realize the actual machines. This section will provide a in depth presentation of the three levels, paired with their exemplification on the example depicted in Figure 2. The figure reports the model of a production process modeled using the proposed three-layer representation. The example shows a production recipe composed of four tasks, which are "T1", "T2", "T3" and "T4". In the uppermost layer, the production recipe is split into the four tasks, which are associated to a subset of capable machines. Then, each task is further refined in the middle layer, where a task is composed of a sequence of services. The concept of service is realized through different implementations. For instance, a service may be related to the behavior of a machine or an interaction with a sensor or an information system (*i.e.*, the MES). These services are further refined in a lower layer, defining precise machine functionality implementing the behavior represented by the service. For example, the bottom layer of Figure 2 depicts the specification of the *pick* service modeled as a sequence of four machine functions. The "Pick" service is related to a robotic manipulator arm. As such, it outlines the ability of the arm to physically pick an object within its operating space. While the Figure 2 reports only the specification of the "Pick" operation, the same type of lower level representation is available for all the services. The ensemble of the three levels provides a complete description of the production process, spanning from a business-oriented viewpoint to a control-focused perspective.

A. Task Level

At the highest abstraction level of the representation, a production recipe is modeled as set of tasks. Concretely, it is expressed by a task-resource graph, which is similar to a RTN. The nodes of the network represent tasks and the solid edges describe a partial order over the execution of the tasks, as shown in the upper layer of Figure 2. Each task identifies a different macro-step of the recipe, and is characterized by the required raw materials and the transformed material states. In addition, one or more machines are associated to each task (dashed edges in Figure). A machine is associated to a task if it provides the functionalities required to carry out such a task. Moreover, a set of attributes annotates the relations between machines and tasks. The attributes, specifically, are the execution time, the hourly cost, the electrical consumption, the estimated efficiency and the necessary tools. For the sake of clarity, such a set of information is not reported in Figure 2. The proposed model allows representing all the alternative sequences and allocations choices allowing to carry out a task.

B. Service Level

The second level refines the tasks-resources representation defined in the upper level. Current manufacturing systems are more and more based on the concept of Service-oriented Manufacturing, which models the interaction with the machines and sensors through services. Therefore, to describe all the steps in which a task is composed, it is necessary to portray the relation between tasks and services. For this reason, the second layer describes the sequence of services called to complete the task. Its representation consists of a control flow graph. Each node in the graph represents either a service or a control flow statement. *Control flow statements* model either the definition of variables, arithmetical operations, conditional and iterative clauses, enabling the representation of complex logical flows of services. Edges in the graph specify the order in witch the behaviors determined by the nodes are executed. A service can be either a *machine service* or an *infrastructure service*:

- a *machine service* models a machine behavior as a sequence of simpler operations called machine functions.
- An *infrastructure service* models the interaction with sensors, actuators and computational resources available in the production system.

Each service is characterized by a set of input parameters, constants and output parameters used by the control flow statements. Then variables defined in a task have global scope with respect to the production recipes, allowing to share variables between subsequent tasks. In addition, infrastructure services allow pausing a task being executed at the end of each machine service. Thus, the model allows identifying when the task can be preempted in favor of higher priority tasks.

Regarding Figure 2, the service level models a task requesting two raw materials before assembling them. In the example, the possibility to stop the task after the *pick* machine service is enabled by the model. Therefore, this level, combined with the upper level, allows scheduling the task execution as a set of sub-tasks. This, in particular, grants executing high-priority tasks more efficiently, filling the machines' dead times and, consequently, implementing interleaving.

C. Machine Function Level

A sequence of sub-services refines the services defined in the second layer. The purpose of the machine function level is to model the actual control behaviors through machine functions. In this regard, machine functions are implemented as behaviors at the Programmable Logic Controller (PLC) level. Each function is characterized by a set of input and output parameters that can be either variables or constants. Unlike the upper levels, the variables defined within the machine function have private scope. Therefore, each construct also have input and output parameters allowing to access the outer scope. In addition, by integrating the knowledge about the service implementation, this level allows precise machine's reconfiguration. Indeed, the scheduler could propose the adoption of new machine functions, better suited to implement the task according to the actual state of the plant.

With regards to the example reported in Figure 2, the control function "Pick" is refined in a sequence of four machine functions: (1) move the robotic arm, (2) close the gripper, (3) move back the arm, and (4) open the gripper to release the material. This level allows planning the execution of the machine services, taking into consideration when a precise atomic function will take place. In fact, it also allows getting a more precise estimation of the time required to carry out a sub-task. Consequently, it enables a better forecast on when certain materials are necessary and on the used tools. It can also be exploited to further optimize the process, to minimize the machines' setup and waiting times.

It is also necessary to assume that flattening the hierarchy is not a viable option. In fact, the model does not allow to conceptually consider a machine function or a service as a task, to include every possible information within a single level. The reason for such a limitation is that, by doing so, the data on machine's status during the execution of a function or a service would be lost. Therefore, a task could be wrongly preempted, leaving some unprocessed materials in buffers or dropping the machine in an unrecoverable state.

IV. Services-based Scheduling

In this section, we propose a reactive-dynamic scheduling algorithm exploiting the proposed three-level modeling approach. The scheduling algorithm exploits the information represented in the model, aiming at minimizing the makespan of the production while maximizing the machine utilization. The algorithm exploits the increased granularity of the model to schedule the sub-tasks (*i.e.*, services) specified within the tasks. Furthermore, knowing the encapsulated services within a sub-task, it is possible to identify the required resources (*i.e.*, tools, materials *etc.*). This allows to schedule these sub-tasks taking into account the various delays, such as waiting times for the retrieval of missing resources.

A. Problem Statement

Let M be a set of the machines available in the production system. Let $R = \{r_1, r_2, \dots, r_n\}$ be the set of recipes. The relation $R_p = \{(r_n, p_n) \mid r_n \in R, p_n \in \mathbb{N}\}$ associates a recipe r_n with a priority p_n. Let $T_n = \{t_1, t_2, \dots, t_n\}$ the set of tasks of a recipe n. For each task $t_i \in T$, $ST_i = \{st_{i1}, st_{i2} \dots, st_{ij}\}$ is the set of sub-tasks of t_i, as modeled in the hierarchical representation of the production process. The set of possible allocations between sub-tasks ST_i and machines M is defined by the relation $PA = \{(st_{ij}, m_k, d_{ij}) \mid st_{ij} \in ST_i, m_k \in M, d_{ij} \in \mathbb{N}\}$, where d_{ij} is the duration of the sub-task st_{ij} allocated on the machine m_k. The actual allocation between sub-tasks and machines is formalized as a relation $S = \{(st_{ij}, m_k, d_{ij}, \tau_{ij}) \mid st_{ij} \in ST_i, m_k \in M, d_{ij} \in \mathbb{N}, \tau_{ij} \in \mathbb{Z}^+\}$, where τ_{ij} is the starting time of the sub-task on a machine expressed as positive integer number.

The scheduling algorithm searches for an assignment S minimizing the makespan while maximizing the machine utilization U_k. The makespan MS is computed as the difference between the starting time of the latest allocated sub-task plus its duration, and the starting time of the first allocated sub-task:

$$MS = \max_{\forall st_{ij} \in S} (\tau_{ij} + d_{ij}) - \min_{\forall st_{ij} \in S} \tau_{ij} \tag{1}$$

The machine utilization u_k, is defined as a percentage of the total production time in which the machine k is busy:

$$U_k = \frac{\sum_{\forall \langle st_{ij}, m_k \rangle \in S} d_{ij}}{MS} \cdot 100 \tag{2}$$

B. Scheduling Algorithm

The scheduling algorithm takes decisions based on the information stored in the model and the priorities assigned at runtime to the recipes. In particular, whenever a new order arrives, the recipes of the requested products are added to the set of recipes to be scheduled. Meanwhile, in case of delays or machine unavailability, the priority of the corresponding recipe is lowered. This allows to allocate other tasks that are actually "runnable", optimizing the machines utilization. Once the missing materials or machines become available, the priority of the tasks is restored. Then, the scheduler is invoked each time an unexpected event occurs. An unexpected event could be the arrival of new orders, delays in retrieving materials or machines incorrectly set-up for an allocated task.

Algorithm 1 Service-based Scheduling

```
 1: $\tau \leftarrow getCurrTime()$
 2: for $\{(st_{ij}, m_k, d_{ij}, \tau_{ij})\}$ in $S$ do
 3:     if $\tau < \tau_{ij}$ then
 4:         $S \leftarrow S \setminus \{(st_{ij}, m_k, d_{ij}, \tau_{ij})\}$
 5:     end if
 6: end for
 7: $R \leftarrow$ sort $R$ by $p$
 8: for $r_n$ in $R$ do
 9:     for $t_i$ in $T_n$ do
10:         $assign(t_i)$
11:     end for
12: end for
```

The algorithm is made of two procedures. The first is described by the Algorithm 1. Its main functionality is to update the schedule S at each invocation. In particular, at line 1, the procedure retrieves the current time. Lines 2-6 remove from the previous schedule all the sub-tasks that are not already allocated and started on a machine at time τ. Line 7 sorts the set of recipes according to their priority. We chose this sorting parameter to handle high-priority orders first. Then, the algorithm evaluates each sorted recipe (line 8-12). For each $r_n \in R$, and each task $t_i \in T_n$, the algorithm invokes the support function `assign(`t_i`)` (line 10).

Algorithm 2 Assign function

```
    Input: $t_i$
 1: for $st_{ij}$ in $t_i$ do
 2:     if $st_{ij}$ in $S$ then
 3:         skip
 4:     end if
 5:     $st\_m \leftarrow 0$
 6:     $st\_end \leftarrow \infty$
 7:     $st\_\tau_{ij} \leftarrow \infty$
 8:     for $\{(st_{ij}, m_k, d_{ij})\}$ in $PA$ do
 9:         $\tau_{ij} = findFreeTimeSlot(st_{ij}, m_k)$
10:         if $(\tau_{ij} + d_{ij}) < st\_end$ then
11:             $st\_m \leftarrow m$
12:             $st\_\tau_{ij} \leftarrow \tau_{ij}$
13:             $st\_end \leftarrow (\tau_{ij} + d_{ij})$
14:         end if
15:     end for
16:     $S \leftarrow S \cup \{(st_{ij}, st\_m, d_{ij}, st\_\tau_{ij})\}$
17: end for
```

Algorithm 2 describes the $assign(t_i)$ function. It provides a machine assignment for every sub-task composing a task. It does so by searching for unused time slots in the schedule of each machine. Therefore it aims at maximizing the busy-times of machines and, thus, at maximizing the utilization of such a piece of equipment. The function only schedules sub-tasks that are not executed or already allocated. In fact, as implemented in lines 2-4, the algorithm checks whether the sub-task st_ij is included in the set of sub-task already allocated S. If it is, the sub-task is skipped. Lines 5-7 define local variables used to save the best solution found, which is the first machine that is able to complete the sub-task. For each possible allocation in PA, the algorithm searches (lines 8-15) for the best one, identified as the earliest starting sub-task on a machine m_k. The function $findFreeTimeSlot(st_{ij}, m_k)$ at line 9 retrieves the starting time-slot of the sub-task st_{ij} executed by the machine m_k. In particular, the function searches the first free time-slot capable of containing st_{ij}, within the schedule S of the machine m_k. It also considers the machine function layer of the model, to assess whether a dependency within two sub-tasks composing the same task is present. A dependency means that the sub-tasks may share resources (*i.e.*, materials) and, thus, must be allocated on the same machine. In such a situation, the function returns $+\infty$. Lines 10-14 check whether the time-slot found by the $findFreeTimeSlot(\dots)$ function is better than the one found in the previous cycle. In such a case, the sub-task is allocated to the machine (lines 11-13) at the found time-slot, and then the schedule is updated (line 16).

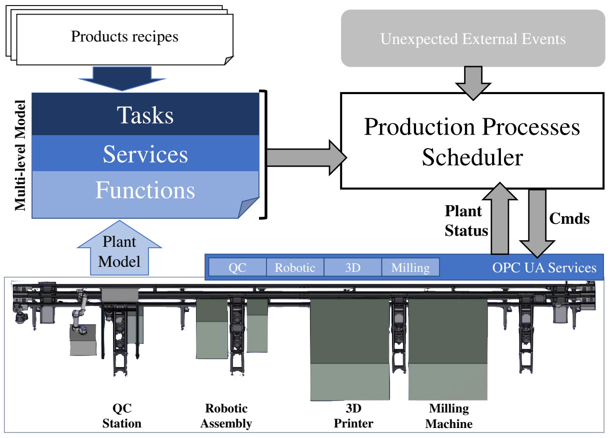

Figure 3: Experimental setup used to assess the methodology. The three-level model of the production processes is built by modeling the production recipes and the manufacturing line equipment. Then, this representation is used to implement a service-based scheduler. Lastly, the scheduler is executed in a real production environment.

The algorithm also maximizes the utilization of the machine: a more granular representation of tasks as sub-tasks allows the scheduler to allocate sub-tasks in smaller time-slots than tasks. This allows to better exploit the unused time of the machines. With regards to equation 2, the implemented allocation strategy allows shrinking the gap between the numerator and denominator. The complexity of the algorithm is linear on the cardinality of S; the complexity of the *findFreeTimeSlot* is also linear as it relies on interval trees.

V. Application and Experimental Result

Figure 3 summarizes the experimental setup used to evaluate the advantages the proposed approach. We applied the presented modeling and scheduling techniques to the full-sized manufacturing line. The plant consists of two 3D Printers, a Quality Control Cell, a collaborative Robotic Cell, a CNC Cell, and a vertical warehouse. The system is able to implement the Service-oriented Manufacturing paradigm: each machine exposes to the user its functionalities as a set of services. The communication is realized through the OPC UA protocol: a well-known standard for industrial machine communication [24]. Furthermore, the system is governed by a commercial MES communicating with the machines.

The production recipes have been modeled according to the three-level modeling approach. Each level in the models are expressed using *SysML activity diagrams*. SysML provides an intuitive graphical language, explicitly tailored to express the structure and behavior of complex systems. Thus, SysML aims at easing the specification and modeling phase for all the three levels of the production model. Furthermore, SysML supports the XML Metadata Interchange (XMI): an XML-based format easing data exchange, manipulation, and analysis. Thus, allowing to easily implement procedures able to analyze and manipulate the data carried by models.

We implemented the algorithm to be able to take as input the proposed three-level model expressed using the SysML syntax.

The algorithm implementation is interfaced with the MES to monitor unexpected external events. Finally, it interfaces with the OPC UA servers deployed on the production plant, in order to monitor the state of the system, and to send commands to the machines to execute the decisions the algorithm takes.

This section, first describes how production recipes are expressed in SysML, while following the proposed hierarchical modeling strategy. Then, we provide more details about the algorithm implementation and its interfacing with the production plant's infrastructure. Lastly we report the results obtained by exploiting the implemented scheduler. We compare the results to those achieved by using the native scheduler provided by the commercial MES governing the system.

A. Implementation

The production recipes described by applying the proposed hierarchical modeling approach are concretely expressed using SysML. Each level described in Section III is modeled as an *activity diagram*. We started defining the data types related to *task*, *service*, and *machine*: the essential types allowing to express all the information necessary within each model. Then, we described the production recipes at the first level of the hierarchy. We modeled the nodes of the graph as an object of type *task*, on which we specified the parameters, such as *name*, *materials*, *etc.*. Each task is assigned to one or more *machine* objects. On these objects, we specified the parameters related to the machine-task relation, such as tools, electrical consumption, execution time, *etc.*. The dependencies between tasks are specified through control-flow arrows.

Each task described in the first level is further refined at the second level of the hierarchy. The correspondent nodes of the graph are represented as an object of type *service*. The service nodes store the information of each service, *i.e.*, *inputs*, *outputs*, *etc.*. Thus, leading to activity diagrams as the one reported in Figure 4. It represents the service-level model, expressed as a SysML activity diagram, of task *T2* of the case study previously shown in Figure 2.

Furthermore, the model is annotated with the information about the calls to the OPC UA protocol required to invoke each service. The order of OPC UA service calls is modeled by using control-flow arrows, while conditional branching in invocation sequences is modeled by *decision nodes*.

Finally, each machine function represented in the lower level is also modeled by a SysML activity diagram. The activity diagram models the PLC behaviors related to machine service. The nodes of the graph are modeled as an object of type *service*, which contains the information related to PLC functionalities, such as *inputs*, *outputs*, *etc.*. The nodes of the graph are specified with control-flow arrows that allow modeling the sequence of the low-level services.

We automatically extract the essential information stored in the XMI description of the SysML model, and we store such information into a JSON file. This representation will provide the input to our implementation of the algorithm described in section IV. The scheduler is implemented on top of a Service Oriented Manufacturing (SOM) software architecture,

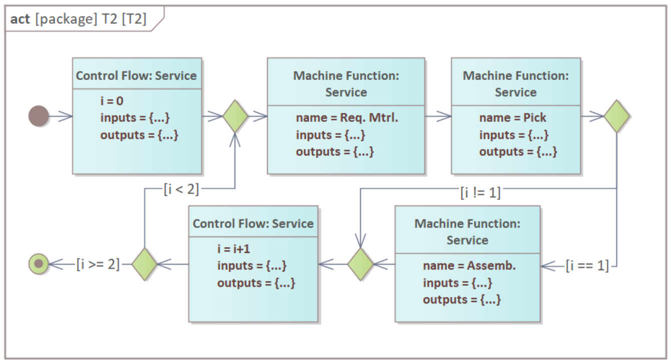

Figure 4: Service-level model of the task *"T2"* represented as a SysML activity diagram. It expresses all the information contained in the second level of our proposed model. Each node is specified by a typed object. The arrows describe the edges of the graph in the middle layer of Figure 2.

extending the functionalities of its automation software component. More specifically, the scheduler is embedded into the *Automation Manager* [25], which interacts with the MES and a set of servers providing access to the manufacturing machines. It retrieves from the MES the set of production orders and the attached bills of materials. The closed-loop communication between the scheduler, the Automation Manager and the MES allows interracting with the machines, retrieving the current state of the plant, and fetching new orders. The a communication is carried out through the OPC UA protocol.

B. Results and discussion

We compare the results obtained by the scheduler implemented by the commercial MES, which relies on a classical RTN-based task representation, against our service-based scheduler, which exploits the proposed hierarchical information model. Table I reports the result obtained with the two different approaches. As a benchmark, we input 450 production processes instances (*i.e.*, production orders), randomly generated from a pool of 4 different recipes. The first row compares the total makespan obtained by the two approaches, calculated using Equation 1 and Equation 2. The proposed scheduler is able to reduce the total execution time needed to complete the 450 orders by almost 40 minutes. Thus, providing an improvement of 3.1% with respect to the state-of-the-practice approach. The next three lines show the comparison of the average amount of time in which the machines are not used (*e.g.*, the change time), the average machines utilization, and the throughput. The proposed approach decreases the average change time, while increasing both the average machine utilization and the system throughput. Thus, the proposed approach successfully hits its target of reducing the makespan, while increasing the average machine utilization.

The improvement is due to the scheduling algorithm enabling a more precise interleaving than the traditional task-resources representation. In fact, dividing one task of a recipe in sub-tasks means partitioning the time necessary to complete such a task. Therefore, the algorithm is able to fill machines

Table I: Comparison between the scheduling of 450 production recipes, using a classical RTN-based representation against the proposed hierarchical modeling approach.

Param	**RTN Representation**	**Hierarchical Representation**	**Diff. (%)**
Cycle Time	20:56:19 (h)	20:17:23 (h)	-3,1%
Avg. Change Overtime	5:42:47 (h)	5:05:16 (h)	-10,95%
Avg. Utilization	74,47%	80,85%	+6,38%
Throughput	235,96 (u)	245,75 (u)	+4,14%
Avg. Time To Complete HP	1:02:08 (h)	0:57:08 (h)	-8,07%
Avg. Time To Complete LP	2:19:25 (h)	2:16:45 (h)	-1,92%

downtimes with sub-tasks of different tasks or even of different recipes. As a consequence, the throughput also increases due to the algorithm ability to fill the production line's execution time-span more efficiently. Lastly, the last two rows compare the average time to complete production recipes at high priority (HP) and low priority (LP). The completion time for a recipe is calculated as the difference between the time instant in which the last task ends and the time instant in which its first task starts. While reducing the completion time for both new high and low priority orders, the proposed representation allows handling more efficiently the arrival of new higher priority orders. In general, the scheduler routine in charge of managing the priorities of the tasks is able to handle more accurately the allocation of sub-tasks whether certain conditions are met (*e.g.*, required materials availability).

Nonetheless, such a methodology can be applied only to manufacturing systems built both conceptually and concretely around the concept of "service". In fact, it assumes that the various machines composing the plant are capable of implementing and exposing functionalities enclosed in such constructs. Furthermore, the presented work assumes that the production domain (*i.e.*, the type of manufacturing industry) handles processes that can be interrupted, with materials temporarily stored in buffers. Therefore, additional constraints and aspects would be needed for this work to be applicable to the production of non-durable goods (*e.g.*, food and beverages).

VI. Conclusions

We presented a multi-level modeling approach to manufacturing processes. According to the service-oriented paradigm, the model organizes hierarchically tasks, services and machine functions. The modeling approach enables new production schedulers able to reason on more comprehensive knowledge about tasks implementation and machine allocation.

We assessed the proposed approach by modeling four production recipes, and scheduling a high number of instances. The results show that the proposed approach offers better performances than a traditional methods. Overall, the achieved increased efficiency should allow reducing the production costs of the single units being produced.

References

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

[2] T. Lojka, M. Bundzel, and I. Zolotová, "Service-oriented architecture and cloud manufacturing," *Acta polytechnica hungarica*, vol. 13, no. 6, pp. 25–44, 2016.

[3] S. Gaiardelli, S. Spellini, M. Lora, and F. Fummi, "Modeling in industry 5.0: What is there and what is missing: Special session 1: Languages for industry 5.0," in *Proc. of Forum on specification Design Languages (FDL)*, 2021, pp. 01–08.

[4] G. P. Georgiadis, A. P. Elekidis, and M. C. Georgiadis, "Optimization-based scheduling for the process industries: From theory to real-life industrial applications," *Processes*, vol. 7, no. 7, 2019. [Online]. Available: https://www.mdpi.com/2227-9717/7/7/438

[5] L. Davis *et al.*, "Job shop scheduling with genetic algorithms," in *Proc. of international conference on genetic algorithms and their applications*, vol. 140, 1985.

[6] F. Pezzella, G. Morganti, and G. Ciaschetti, "A genetic algorithm for the flexible job-shop scheduling problem," *Computers & operations research*, vol. 35, no. 10, pp. 3202–3212, 2008.

[7] A. Rajabinasab and S. Mansour, "Dynamic flexible job shop scheduling with alternative process plans: an agent-based approach," *The International Journal of Advanced Manufacturing Technology*, vol. 54, no. 9, pp. 1091–1107, 2011.

[8] J. Błażewicz, W. Domschke, and E. Pesch, "The job shop scheduling problem: Conventional and new solution techniques," *European Journal of Operational Research*, vol. 93, no. 1, pp. 1–33, 1996.

[9] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," *Theoretical Computer Science*, vol. 344, no. 2, pp. 243–278, 2005.

[10] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization: An overview," *Swarm Intelligence*, vol. 1, 10 2007.

[11] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *Journal of Global Optimization*, vol. 39, pp. 459–471, 11 2007.

[12] M. Affenzeller, S. Winkler, S. Wagner, and A. Beham, "Genetic algorithms and genetic programming," 03 2009.

[13] J. Heizer, *Production and Operations Management*. Allyn and Macon, Needham Heights, Massachusetts, 1991.

[14] S. Nahmias, *Production and operations analysis*. McGraw-Hill, 2015.

[15] H. Askari Nasab, Y. Pourrahimian, E. Ben-Awuah, and S. Kalantari, "Mixed integer linear programming formulations for open pit production scheduling," *Journal of Mining Science*, vol. 47, pp. 338–359, 05 2011.

[16] Y. Krotov, "Csp production planning system," *AISTech - Iron and Steel Technology Conference Proceedings*, vol. 2, pp. 767–772, 01 2006.

[17] O. Cardin, D. Trentesaux, A. Thomas, P. Castagna, T. Berger, and H. Bril, "Coupling predictive scheduling and reactive control in manufacturing hybrid control architectures: state of the art and future challenges," *Journal of Intelligent Manufacturing*, vol. 28, no. 7, 2017.

[18] C. C. Pantelides, "Unified frameworks for optimal process planning and scheduling," in *Proceedings on the second conference on foundations of computer aided operations*, 1994, pp. 253–274.

[19] E. Kondili, C. Pantelides, and R. Sargent, "A general algorithm for short-term scheduling of batch operations—i. milp formulation," *Computers & Chemical Engineering*, vol. 17, no. 2, pp. 211–227, 1993.

[20] I. Harjunkoski and R. Bauer, "Sharing Data for Production Scheduling Using the ISA-95 Standard," *Frontiers in Energy Research*, vol. 2, 10 2014.

[21] N. Schmidt and A. Lüder, "AutomationML in a Nutshell," 2015.

[22] S. Spellini, S. Gaiardelli, M. Lora, and F. Fummi, "Enabling component reuse in model-based system engineering of cyber-physical production systems," in *Proc. of IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2021, pp. 1–8.

[23] L. Berardinelli, S. Biffl, A. Lüder, E. Mätzler, T. Mayerhofer, M. Wimmer, and S. Wolny, "Cross-disciplinary engineering with AutomationML and SysML," *at - Automatisierungstechnik*, vol. 64, pp. 253 – 269, 2016.

[24] "OPC Unified Architecture specification – Part 1: Overview and concepts release 1.04 OPC Foundation," 2017.

[25] S. Gaiardelli, S. Spellini, M. Panato, M. Lora, and F. Fummi, "A software architecture to control service-oriented manufacturing systems," in *Proc. of IEEE/ACM Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2022, pp. 1–4.