Journal of Industrial Information Integration 27 (2022) 100329

Contents lists available at ScienceDirect

Journal of Industrial Information Integration

journal homepage: www.elsevier.com/locate/jii

Full length article

Model-based system engineering supporting production scheduling based on satisfiability modulo theory

Jingqi Chen [a], Guoxin Wang [a], Jinzhi Lu [b,*], Xiaochen Zheng [b], Dimitris Kiritsis [b]

[a] Beijing Institute of Technology, Beijing 100081, China
[b] EPFL SCI-STI-DK, Station 9, CH-1015 Lausanne, Switzerland

ARTICLE INFO

Keywords:
Production scheduling
MBSE
SMT
Property verification
KARMA language

ABSTRACT

Production scheduling enables a production system to allocate resources for the efficient and low-cost production. However, pure mathematical methods are typically utilized for production scheduling, which are not understandable and practical for different domain engineers. Moreover, heterogeneous information during production scheduling may result in communication ambiguity or an information delay. Furthermore, the interaction and reuse of production data is limited, owing to the lack of a unified expression of the production information. To overcome these challenges, a model-based systems engineering (MBSE) approach based on the satisfiability modulo theory (SMT) is proposed to support production scheduling. A multiple architectural view modeling language, KARMA, is used as the basis to construct production scheduling elements and formalize the production scheduling processes using architecture models. Such graphical models are used to improve the understanding and communication among engineers in different domains. Then, property verification based on the MBSE and SMT is applied to solve the model constraints and select optimal schemes for the production scheduling. A case study of a package production line is proposed to evaluate our approach. Its scheduling goal is to maintain high production efficiency, decrease the working time and the cost of workers when increasing working distances during the COVID-19 pandemic. From the case study, we observed that KARMA language enables a description of modeling production scheduling and the optimization of scheduling schemes in a unified manner. Using the SMT solver, the constraints on scheduling corresponding to the sudden decrease in workers are evaluated to exclude inappropriate schemes and generate the optimal one.

1. Introduction

A production system produces items according to certain specifications through the integration of human resources, equipment, raw materials, and other resources [1]. In this process, complex production resources need to be allocated according to the needs of multiple stakeholders. If the allocation of resources is unreasonable, it may lead to the waste of production resources or the overload of workers, which results in reducing production efficiency and increasing production costs. Production scheduling refers to allocating equipment, human resources, and operations to optimize work loads [2], and it is applied as a common method to balance the production requirements and resources to maximize productivity [3].

Production scheduling process primarily includes modeling and solving. Although the production scheduling method has been applied in the production field for a long time, the following problems still exist. (1) The production scheduling process involving multiple stakeholders and complex cooperation results in a high communication cost. Stakeholders participate in different aspects of the scheduling process to interact with information and material. It is difficult to describe the real scheduling process completely if the information is only described using natural language when modeling. In addition, there are risks of change and ambiguity in the communication between different stakeholders, which may lead to an information delay and ambiguity. (2) It is difficult to integrate and interact with the production scheduling data when modeling production scheduling, owing to the lack of a unified expression. During the production scheduling process, different departments produce heterogeneous data, such as domain documents, graphics, or models. Stakeholders may experience difficulties in understanding different formats. (3) The solving process for production scheduling requires professional algorithmic knowledge, which is abstract and lacks graphical expressions when operating the scheduling model elements. Such a method with complex algorithms cannot be visually associated with the scheduling model; thus, it is not

* Corresponding author.
E-mail address: jinzhi.lu@epfl.ch (J. Lu).

https://doi.org/10.1016/j.jii.2022.100329
Received 14 June 2021; Received in revised form 20 November 2021; Accepted 1 February 2022
Available online 12 February 2022

understandable to stakeholders in different domains, thus the response to unexpected additional needs is not prompt.

Model-based systems engineering (MBSE) has been utilized to overcome the above challenges [4]. MBSE applies the model formalization that supports the modeling and analysis of the system in a graphical form, which enhances the information acquisition, analysis, sharing, and management [5]. First, MBSE supports an accurate and complete information description through modeling to enable stakeholders to communicate without ambiguity. Compared with documented information, the MBSE method uses a formal method to replace documents to help stakeholders reach a consensus with unambiguous understanding of the production scheduling details. Second, MBSE uses a unified modeling language to describe the heterogeneous data generated in the production scheduling process, and realizes the integration, transmission, and use of heterogeneous data. The unified modeling language describes the production scenarios, scheduling process and solving process of production scheduling process. The heterogeneous information is stored in consistent structures and transformed into a unified data source to realize the capture, transfer, and transformation of information for the integrated management of production scheduling process. Finally, MBSE is able to specify each process of the production scheduling in a graphic form and evaluate the constraints of models with property verification using the unified modeling language to obtain the acceptable schedule [6]. The models and solution process are available for interaction based on the unified modeling language; thus, it is feasible to provide a timely respond to constraint changes.

This study proposes an approach to model the production scheduling and generate the solution based on MBSE combined with the satisfiability modulo theory (SMT) [7]. SMT containing a series of mathematical axioms is available to check the logical expression satisfiability. This study presents three fundamental contributions, as follows: (1) the KARMA model library and models for production scheduling based on system engineering are constructed using a unified modeling language; (2) a theoretical method for production scheduling based on models is proposed, which describes and solves the constraints from models based on SMT; (3) based on the proposed method, the property verification applied in MBSE is developed, and this approach is validated using the production scheduling for a package production line to mitigate the impact of COVID-19.

The remainder of this paper is organized as follows: Section 2 introduces some literature related to production scheduling methods and property verification in MBSE; Section 3 presents the theoretical method for modeling and property verification for production scheduling; Section 4 validates the approach by considering the package production line and analyzes the results; Section 5 and Section 6 presents the discussion, conclusions and future work.

2. Related work

2.1. Methods on production scheduling

Production scheduling is a decision-making process, which includes resource allocation and task sequencing. It ensures that limited resources are allocated to satisfy constraints in a reasonable time to perform a series of tasks in a sequence [8–10]. The mainstream approaches to the scheduling problem include the methods that are based on graphs or a network, and those that are not.

The scheduling method based on graphs and networks uses a mathematical model as the algorithm's basis without graphical expressions, such as mathematical programming, scheduling based on rules, and network supporting scheduling [11,12]. When the mathematical programming is applied, the scheduling problem is split into sub-problems with mixed integer linear programming [13] because there are limitations to solving NP(Non-deterministic Polynomial) problems using mathematical programming. The scheduling scheme based on rule-based scheduling is generated through heuristic algorithms with the application of priority scheduling rules [14–17]. Scheduling based on search algorithms provides a method to balance the solution quality and required time [18]. Genetic algorithms [19,20], simulated annealing algorithms [21], and TABU search algorithms [22] may be applied to production scheduling. In addition, Luo proposed an efficient memetic algorithm (EMA) to minimize the completion time, the maximum workload, and the total energy consumption [23].

Scheduling for processes is typically modeled using Petri Net and solved using other tools if the scheduling is based on graphs and networks [24]. Peng combined the timed Petri Net and A-star search algorithms to solve the scheduling problem. Petri Net is used to formalize the production process, and the A-star search algorithm is used to search for the acceptable scheme using the Reachability Graph of Petri Net [25]. The modeling and solution processes of the method are separated, thus the modeling data based on Petri Net cannot be directly applied to the solving algorithm. In some recent studies, data mining [26] and a digital twin [3] are applied in the production scheduling. Cheng discussed the applications of data mining techniques in production management and analyzed the limitations [26]. Ruppert proposed the digital twin which is continuously capable to predict the production status and provide information for monitoring of production performance [3]. Except for the conventional method of using Petri net, the virtual environment [27] is a new way to support the production analysis [28] but the cost of equipment is also hard to be affordable.

There still exist limitations for the above scheduling methods. Modeling the production scheduling in these methods lacks visual model expressions, which is not conducive to the stakeholders' understanding. Moreover, the data interaction between modeling and solving is difficult for scheduling methods based on graphs and networks because the approaches to modeling and solving are not consistent. An approach to unifying the data of modeling and solution with visual and complete models is required for the production scheduling problem.

2.2. Property verification based on model-based system engineering

MBSE provides a formalized methodology to specify models, unify the multiple data resources of information lifecycle management, and validate and verify the systems. According to INCOSE, MBSE is the formalized application of lifecycle modeling, which is used to support system requirements, design, analysis, verification, and validation activities for complex system development [29]. The core of the MBSE process is the system model, which formalizes the system visually and conforms the stored information. The graphic approach to modeling as a visual annotation [30] makes it available for information exchange in different domains and reduces communication ambiguity. A modeling language is required to build models in the MBSE method. The system modeling language (SysML) [31] and unified modeling language (UML) [32] are the most widely used modeling languages; they are composed of graphic symbols and information models for visualizing systems. UML is an object-oriented software modeling language; however, those users who are unfamiliar with software engineering have difficulty in understanding and applying UML [32]. SysML is based on UML; it is the modeling language used for system engineering applications to specify system requirements, functions, or other criteria [33].

The MBSE method is widely applied for product design; however, it is not widely used in the production domain. A limited number of studies combined the methodology of the production domain, such as lean production [34], digital twin [35] or production line management [36], with MBSE, so as to improve the communication between stakeholders and improve production efficiency. Moreover, Batarseh [37] proposed a technical solution of applying MBSE to assembly lines by combining the SysML modeling language and ATLAS transformation language (ATL) to simulate the assembly line automatically [38]. Although MBSE is applied to evaluate the production system, the simulation method that

is used involves two or more independent languages; thus, the learning and simulation costs are large.

Consequently, it is necessary to propose an approach to efficiently evaluate and generate production scheduling schemes based on MBSE. This approach should formalize the unification of production scheduling models and solve the production scheduling problem. The formal verification [39] can evaluate the constraints of models and generate the acceptable scheme, which is different from simulations. The widely used modeling languages are semi-formal languages [40], which lack the formalism to exactly specify models for formal verification. Hence, the constraints of the models cannot be formalized and verified to detect the acceptable scheduling scheme. The property verification, which is a formal verification method in MBSE, refers to the specifying and evaluating of the constraints that consist of model elements [41], which supports the requirements evaluation and quality evaluation for system by verifying the constraints of the models.

The application of property verification in MBSE is limited and the applied process of property verification is complex for various languages and solvers. Gogolla designs the UML-based specified environment (USE) for the specification and verification of information systems [42]. USE combines UML and the object constraint language (OCL) [43] to describe the relationships of model elements and transform them into constraint satisfaction problems. Then, the theorem prover or model checker is called to verify constraints. Feldmann formalized elements of an electronic mechanical manufacturing system by combining OWL [44] and SysML to check compatibility [45]. For the above applications, there is no description of the verification process with complex integrated languages. The data in the modeling and verification processes is not unified, which significantly increases the learning and use costs of users. Additionally, the supported modeling language is single, that is, modeling and verification with heterogeneous models is not allowed.

2.3. Summary

The above literature analysis shows that research on production scheduling based on MBSE is limited. The motivations for this study are listed in comparison with other studies on production scheduling based on MBSE:

- To support the unified and formalized description of the production scheduling process, scheme generation, and evaluation.
  This is to support the formalized description of a production scheduling scenario and the description of the process of solving scheduling problems to generate an optimal scheme with a unified modeling language based on MBSE. Because of the unified modeling language, the models of production scheduling scenarios and solving process can be formally integrated in an unified formalized expression. Consequently, the data interaction between the production scheduling model and solving process is supported.
- To support property verification for heterogeneous modeling to generate optimal scheduling schemes.
  Property verification in MBSE is required for the scheduling problem. Property verification can be done for the constraints of one model or those of heterogeneous models by using constraints in the first order logical forms from the users. Moreover, the approach exhibits low learning and usage costs because this approach is available for multiple modeling languages without requiring learning a new modeling language.

3. MBSE with property verification supporting production scheduling

The proposed approach to production scheduling is detailed in this section. The formalization of basic elements in production scheduling and the theoretical basis of solving the scheduling problem using the constraints of scheduling and the solution framework are stated collectively. The production scheduling scenarios and possible solutions are modeled and formalized in the framework to build the constraints from the models and evaluate the constraint satisfiability to narrow the scheduling space and obtain an acceptable solution.

3.1. Problem statement

Production scheduling is modeled as shown in Fig. 1, where m tasks are allocated to n workers, and the basic assumptions are followed.

1. m tasks are allocated to n workers.
2. Any worker j ($j = 1, 2, 3 \dots n$) are not allowed to change their working stations.
3. Any task i ($i = 1, 2, 3 \dots m$) is executed only once in a production cycle.
4. The execution of some tasks has priority.
5. There is standard operating time $StandardTime_i$ associated to each task i.
6. There are w types of tasks with using w different specialties in this production line and each type of task is divided into y different difficulty levels. The $h(h = 1, 2, 3...w)$ task type refers to the difference operation type such as machining or assembling. The $l(l = 1, 2, 3 \dots y)$ difficulty level refers to the difficulty of tasks' implementation.
7. There are workers with w types of differential specialties are classified into x different grade-levels. There is a one-to-one match between w workers' specialties and w task types. The specialty of some worker refers to an adept operation for the worker such as some machining. The grade-level of some worker refers to the level of operating proficiency.
8. Each worker has own specialty but can operates all task of differential type. If the worker with the h specialty operates the task of h type, the worker's specialty matches to the task's type so that working efficiency increases and the working time decreases. The final working time for the worker with the h specialty operating the task of h type should be multiplied with a time factor tw_h^a while the time factor if the worker with h specialty not operating the corresponding task of h type is tw_h^b. When the tw_h^b is multiplied which means the specialty of the worker does not match the task's type. The tw refers to the time factor and the subscript h indicates task type or the specialty while the superscript a and b refer to the names of the time factors.
9. Each worker of own specialty are different in the level of operating proficiency. Workers in the different grade-level are associated with different worker salary $Salary_k(k = 1, 2, 3 \dots x)$ and time factor $tw_{lev}^{k\sim l}(k = 1, 2, 3 \dots x; l = 1, 2, 3 \dots y)$ where superscript k and l refer to the grade-level of workers and the difficulty level of tasks while the subscript lev refers to the factor's name. The time factor $tw_{lev}^{k\sim l}$ refers to the additional time weight coefficient of workers in different grade-levels operating tasks of different difficulty for showing the different working efficiency, which need to be multiplied when the working time is calculated. Thus, there are y factors associated with the worker of k grade-level and which factor is decided depends on the difficulty level operated task. Generally, the working efficiency of the worker with higher grade-level is higher which is corresponding to the $tw_{lev}^{k\sim l}$ and the working salary is more for the worker with higher grade-level.
10. The amount of tasks m, the required amount of working workers n, each standard time $StandardTime_i$, the amount of task types w, the amount of difficulty levels for tasks y, the grade-levels of workers x, the time factors tw_h^a, tw_h^b and $tw_{lev}^{k\sim l}$, the salaries $Salary_k$ and the priority of task operation for the production plan are known. Difficulty level and standard working time of each task are known.

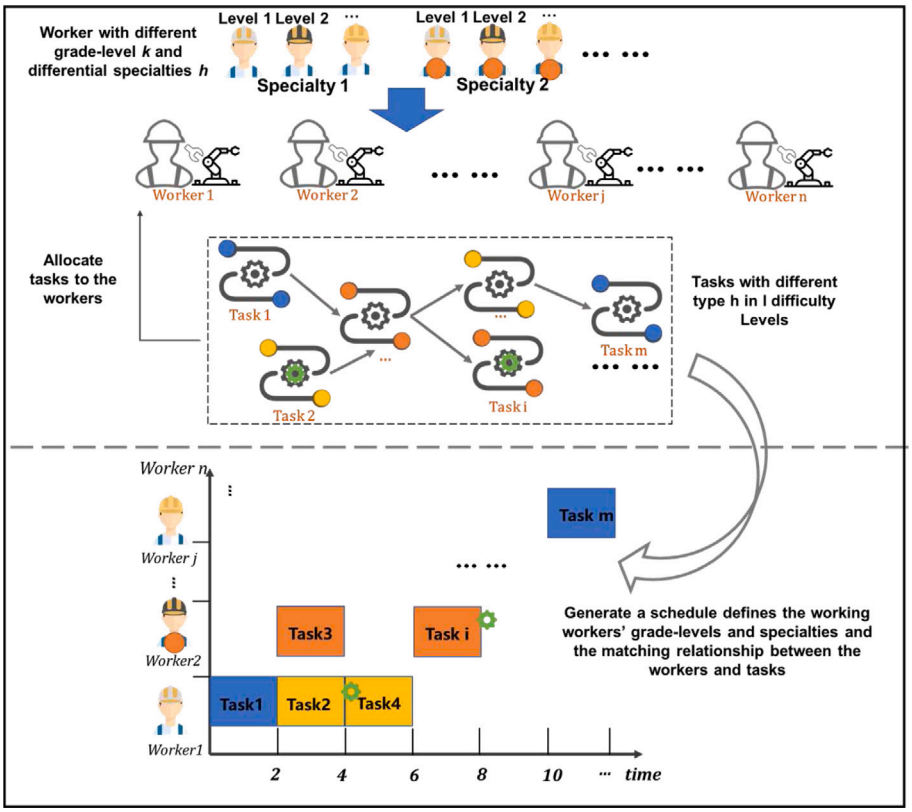

Fig. 1. Problem statement of production scheduling.

11. Which grade-level of each working worker at each working station, what specialty of workers at each station is and the matching relationship between the m tasks and n workers are unknown, which are required to be determined after scheduling.
12. The ability of workers in each level is constant.

The scheduling problem is based on a particular scheduling objective, such as minimizing the processing time of one product, maximizing the productive efficiency, and reducing the worker salary in the production process. Scheduling objectives are set as set O, which includes different scheduling objectives $O = \{O_1, O_2, O_3 ... O_z\}$.

3.2. MBSE supporting production scheduling formalization

Formalization provides the basis for the production scheduling models and the related constraints. The Graph, Object, Property, Point, Relationship, Role, and extensions (GOPPRRE) meta–meta modeling method [46] is selected as the theoretical basis of the production scheduling models. GOPPRRE modeling theory with complete formalization basis is considered to be the best method to develop domain specific models [46], which supports the specification of meta-models and models in different domains. The unified modeling language KARMA, which is based on the GOPPRRE method [47], is used to describe the production scheduling.

The GOPPRRE modeling method extracts the commonness of meta models in different domains and sorts them into six meta–meta models in a higher level of abstraction to support the development of different domain-specific models in a formalized way. The basic elements of GOPPRRE are the six meta–meta models (Graph, Object, Property, Point, Relationship, Role) and the set of their relationships. Graph refers to the collection of the Objects and Relationships. Object refers to an entity in Graphs (e.g., the class in UML). Point refers to a port of an Object. Relationship refers to a connection between Objects or different Points of Objects. Role refers to defined connection rules that map to relevant relationships. Property refers to the attributes of the other 5 meta–meta models, and its data typically includes numerical, Boolean, string, and collection types.

The GOPPRRE modeling method is applied according to the M3–M0 modeling framework based on the six elements [48]. The framework consists of meta–meta models (M3), meta models (M2), models (M1), and views in the real world (M0). The meta–meta models are the basis of meta models, the meta models are developed to be the models and the models are combined to represent the views in the real world. Meta–meta models refer to both the six modelings concepts and their interrelationships for developing the meta-models. Graph contains other five meta–meta models, Object contains Points and Properties, Relationship contains Roles and Properties, and Roles and Points contain Properties. The symbol =:: {} in the equation refers to the collection symbol. The formal expression is given in the following formalization (1):

$$\begin{aligned} &Graph =:: \{Object, Relationship, Role, Point, Property\} \\ &Object =:: \{Point, Property\} \\ &Relationship =:: \{Role, Property\} \\ &Point =:: \{Property\} \\ &Role =:: \{Property\} \end{aligned} \tag{1}$$

Meta models are defined based on the meta–meta models, which also inherit from them. Meta models construct different modeling languages, which are used to model and formally describe the elements of the systems and system development. At the M2 level, the meta model of Graph $GraType$ is marked as $Graph_{GraType}$, which is instantiated based on meta–meta model $Graph$ such as $Graph_{UseCasediagram}$. Object is marked as $Object^{ObjType}_{GraType}$, which refers to the meta model of Object $ObjType$ included in the meta model of Graph $GraType$. $Relationship^{RelaType}_{GraType}$ is a meta model of Relationship $RelaType$ in the Graph meta model $GraType$. $Role^{RoType}_{RelaType}$ refers to a meta model of Role $RoType$ related to the Relationship meta model $RelaType$. $Point^{PoType}_{ObjType}$

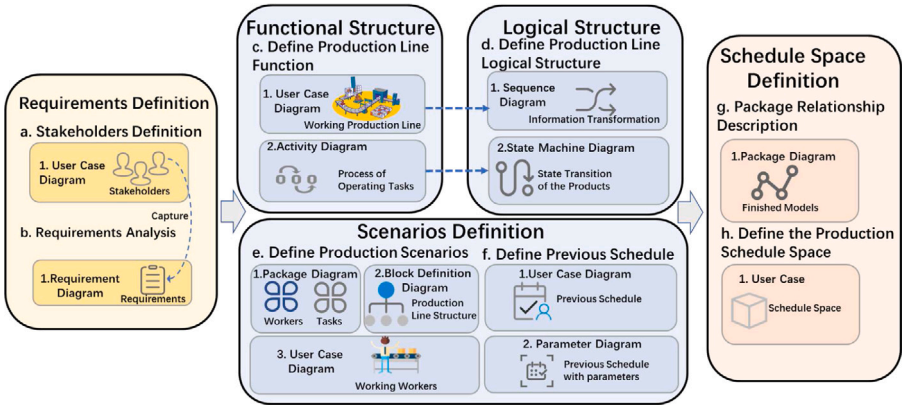

Fig. 2. Modeling process of production scheduling.

refers to a meta model of Point $PoType$ related to the Object meta model $ObjType$. $Property^{ProType}_{otherType}$ refers to Property meta model $ProType$ related to the 5 other meta models whose type is $otherType$.

The models are built based on the meta models. At the M1 level, $Graph_{GraType}(Gra)$ is a Graph model Gra based on the meta model of Graph $GraType$. Similarly, $Object^{ObjType}_{GraType}(GraName, ObjName)$, $Relationship^{RelaType}_{GraType}(GraName, RelaName)$, $Role^{RoType}_{RelaType}(RelaName, RoName)$, $Point^{PoType}_{ObjType}(ObjName, PoName)$, and $Property^{ProType}_{otherType}(otherName, ProName)$ refer to the formalization of Object instance, the Relationship instance, the Role instance, the Point instance, and the Property instance at the M1 level. The models are finally used to describe views in the real world.

3.3. Formalization of production scheduling

In this section, SysML models are developed to support production scheduling based on the GOPPRRE modeling method using the KARMA language. To evaluate the models and exclude the unacceptable schemes in favor of those that are better, model elements are abstracted from the built SysML models and combined as the relevant constraints and scheduling objects. The basic formalization of the production scheduling models and the formalization of an optimal scheduling scheme solving based on the models are stated.

3.3.1. Formalization of production scheduling models

The modeling process used to support the production scheduling analysis is shown in Fig. 2, including 5 stages: (1) requirement definition; (2) functional structure description; (3) logical structure description; (4) scenario definition; and (5) scheduling space definition. To specify how to model the production scheduling, the applied model elements based on SysML are stated in the formalization of models, which are stated as follow.

Requirement definition:. The requirement definition is divided into the stakeholder definition and the requirement analysis, which are described by the Use Case Diagram and Requirement Diagram, respectively. As shown in Fig. 2a.1, three types of stakeholders are captured using use case diagrams, such as customers, workers, and production department. Its formalization is shown in Eq. (2).

$$\begin{aligned} Graph_{UseCase}(a.1) =:: \{ & \Sigma Object^{Package}_{UseCase}(a.1, Package_{Stakeholder}), \\ & \Sigma Object^{Actor}_{UseCase}(a.1, Actor_{Stakeholder}), \\ & \Sigma Relationship^{Contain}_{UseCase}(a.1, Line_{Contain}), \\ & \Sigma Role^{in}_{Contain}(Line_{Contain}, in), \\ & \Sigma Role^{out}_{Contain}(Line_{Contain}, out), \\ & \Sigma Property^{String}_{Package}(Package_{Stakeholder}, ID), \\ & \Sigma Property^{String}_{Actor}(Actor_{Stakeholder}, ID), \\ & \Sigma Property^{String}_{Contain}(Line_{Contain}, ID)\}. \end{aligned} \tag{2}$$

The Σ and $=::$ refer to the label of the set and what it contains, respectively. The formalization rules follow the modeling rules in Section 3.2. This formalization (2) describes a model of the $UseCase$ Diagram at the M1 level, labeled as a.1 in Fig. 2. It includes the object instances *Actor* that represent stakeholders, the object instances *Package* that represent stakeholder sets, the relationship instances *Contain* connecting the *Actor* with the *Package*, and the two end roles of the *Contain*. The role instances referring to the two ends of relationship instances. The property instances in this model indicate ID names of each model elements. The two identifiers in the brackets of each object instance refer to the related graph instance and object instance. For instance, *a.1* refers to the *UseCase* graph instance and $Package_{Stakeholder}$ refers to the object instance *Package* in the formalization $Object^{Package}_{UseCase}(a.1, Package_{Stakeholder})$. The subscript of the object instance $Stakeholder$ is a part of the identifier to distinguish the relevant object instance.

To simplify the formalization, two ends of relationship instance and the property instances of showing identifiers are substituted with symbols. For each relationship instance, the two end *out* and *in* of the relationship instance are attached to it. For each model element such as the object instance, the relationship instance, the role instance, the point instances and the property instances, there is a property instance for stating the identifier of the model elements. The role instances are replaced with ΣR, and the set of property instances referring to the identifiers are replaced with ΣP_i. Except for the property instances referring to the identifiers, the other property instances are still listed separately in the following formalization.

The requirement analysis is used to capture the requirements from stakeholders. The top-level requirements and the sub-requirements decomposed from them are described as the object instances *Requirement* in the requirement diagram. The captured requirements and refined requirements are described in the Requirement Diagram (b.1 in Fig. 2) as the following formalization (3):

$$\begin{aligned} Graph_{Requirement}(b.1) =:: \{ & \Sigma Object^{Requirement}_{Requirement}(b.1, Requirement), \\ & \Sigma Relationship^{Contain}_{Requirement}(b.1, Line_{Contain}), \\ & \Sigma Relationship^{Refine}_{Requirement}(b.1, Line_{Refine}), \\ & \Sigma Property^{int}_{Requirement}(Requirement, val), \\ & \Sigma Property^{String}_{Requirement}(Requirement, Pr) \Sigma R, \Sigma P_i\}. \end{aligned} \tag{3}$$

The object instances *Requirement* represent the requirements with the property instance $Property^{String}_{Requirement}(Requirement, Pr)$ and $Property^{int}_{Requirement}(Requirement, val)$ which refer to the content of requirements and the corresponding value to the requirement respectively. The superscript $String$ and int present the datatype of property instances' value. The relationship instance *Containing*, *Refine*, *Associate* state the connections between the object instances.

Functional structure:. After the requirement definition, the functional structure is formalized which describes the function of the production line, which is defined by Use Case Diagram and Activity Diagrams.

The performance of the production line in the factory and the provided services for stakeholders are described in the Use Case Diagram as c.1 in Fig. 2. The Use Case Diagram includes p stakeholders represented by the object instances *Actor*, the corresponding production system represented by the object instance *System* and the system components represented by the object instance *Block*, the provided use cases represented by the object instance *Use Case*, and the relationships among them. The relationship instances show the connection between the object instances. The formalization is listed as the following formalization (4):

$$\begin{aligned} Graph_{UseCase}(c.1) =:: \{ & \Sigma Object^{Actor}_{UseCase}(c.1, Actor), \\ & \Sigma Object^{Block}_{UseCase}(c.1, Block_{Component}), \\ & Object^{System}_{UseCase}(c.1, O_{ProductionSystem}), \\ & \Sigma Object^{UseCase}_{UseCase}(c.1, UseCase), \\ & \Sigma Relationship^{Association}_{UseCase}(c.1, Line_{Association}), \\ & \Sigma Relationship^{Contain}_{UseCase}(c.1, Line_{Contain}), \\ & \Sigma Relationship^{Generalization}_{UseCase}(c.1, Line_{Generalization}), \\ & \Sigma Relationship^{Extend}_{UseCase} \\ & (c.1, Line_{Extend}), \Sigma R, \Sigma P_i \}. \end{aligned} \tag{4}$$

The Activity Diagram c.2 in Fig. 2 describes the sequence of operating tasks in the production line. The tasks are modeled through the object instance *Action* referring to the action. To complete the whole process, the object instance *Start* and *End* referring to the point of starting and ending, and relationship instance of *Control Flow* are added. The formal expression is given in the following formalization (5):

$$\begin{aligned} Graph_{Activity}(c.2) =:: \{ & \Sigma Object^{Action}_{Activity}(c.2, ActionName), \\ & Object^{Start}_{Activity}(c.2, Start), \\ & Object^{Stop}_{Activity}(c.2, Stop), \\ & \Sigma Relationship^{Control\,Flow}_{Activity} \\ & \times (c.2, Line_{Control\,Flow}, \Sigma R, \Sigma P_i) \}. \end{aligned} \tag{5}$$

Logical structure:. The dynamics views of the production system are described in the logical structure using State Machine Diagram and Sequence Diagram. The d.1 of Fig. 2 describes the information interaction in the production system. The dynamics of the production system states, including the object instance *lifeline* which refers to the entity of processing information, *operators* attaching to the $Lifeline$ which refers to the action token, and the relationship instance *transferred information* showing the information flow, are described as Sequence Diagram models. The formalization (6) is described as the following:

$$\begin{aligned} Graph_{Sequence}(d.1) =:: \{ & \Sigma Object^{Lifeline}_{Sequence}(d.1, Lifeline), \\ & \Sigma Object^{Operator}_{Sequence}(d.1, Operator), \\ & \Sigma Relationship^{SendMessage}_{Sequence} \\ & \times (d.1, Line_{SendMessage}, \Sigma R, \Sigma P_i) \}. \end{aligned} \tag{6}$$

The dynamic perspective of the production line in the production system, which refers to the state change of the product. The object instances *State* refer to the state of product during the production process, is included in the State Machine Diagram Fig. 2d.2. There are three other object instance *Start*, *Stop* showing the starting state ans ending state and the relationship instance showing the state transition. Its formal expression is as follows in formalization (7):

$$\begin{aligned} Graph_{StateMachine}(d.2) =:: \{ & \Sigma Object^{Start}_{StateMachine}(d.2, Start), \\ & \Sigma Object^{State}_{StateMachine}(d.2, Task_i), \\ & \Sigma Object^{Stop}_{StateMachine}(d.2, Stop), \Sigma R, \Sigma P_i, \\ & \Sigma Relationship^{Transiste}_{StateMachine}(d.2, Line_{Transiste}) \}. \end{aligned} \tag{7}$$

Scenarios definition:. Scenario definition describes the application background of the production scheduling, which includes the factory compositions. If there exists an original scheduling scheme for the production, the original scheduling scheme is modeled in this stage in order to compare with the new scheme. The scenario definition is described by Package Diagram, Block Definition Diagram, Use Case Diagram, and Parameter Diagram.

Tasks with a standardized working time in different difficulty level $l = \{1, 2, 3 \ldots y\}$ are stated in Package Diagram e.1, which is defined in the fifth item of Section 3.1. The different difficulty levels are represented by the object instances *Package* and the tasks of production line are represented by the object instances *Block*. The relationship instances between the different difficulty levels and the tasks are represented by the model connection *Association*. The property instance $Property^{String}_{Package}(PackageName, Level_l)$ attached to object instances *Package* represents the task difficulty level, and the property instance $Property^{int}_{Block}(Task_i, StandardTime)$ states the standardized working hours, and the property instance $Property^{String}_{Block}(Task_i, Type)$ refers to the task type, whose corresponding property types are string and integer and string, respectively. The formal expression is shown in formalization (8):

$$\begin{aligned} Graph_{PackageDiagram}(e.1) =:: \{ & \Sigma Object^{Package}_{PackageDiagram}(e.1, PackageName_l), \\ & \Sigma Object^{Block}_{Package}(e.1, Task_i), \\ & \Sigma Relationship^{Association}_{Package}(e.1, Line_{Association}), \\ & \Sigma Property^{String}_{Package} \\ & (PackageName, Level_l), \\ & \Sigma Property^{int}_{Block}(Task_i, StandardTime), \\ & \Sigma Property^{String}_{Block}(Task_i, Type) \Sigma R, \Sigma P_i \}. \end{aligned} \tag{8}$$

The compositions of the production system, such as equipment, workers, and management department, are described in the Block Definition Diagram e.2. The compositions are represented by the object instances *Block* and the composition sets are represented by the *Package*. Its formalization (9) is shown as the following:

$$\begin{aligned} Graph_{BlockDefinition}(e.2) =:: \{ & \Sigma Object^{Block}_{BlockDefinition}(e.2, BlockName), \\ & \Sigma Object^{Package}_{BlockDefinition}(e.2, PackageName), \\ & \Sigma Relationship^{Association}_{BlockDefinition}(e.2, Line_{Association}), \\ & \Sigma R, \Sigma P_i \}. \end{aligned} \tag{9}$$

Different use cases that workers in the grade-level k ($k = 1, 2, 3 \ldots x$) (defined in Section 3.1) operates tasks of different difficulty level l ($l = 1, 2, 3 \ldots y$) (defined in Section 3.1) are defined in Use Case e.3. The object instance *Actor* refers to the worker sets where the workers are in the same grade-level, and the object instance *Use Case* represent the service sets of operating tasks are in the same difficulty level. The property instance $Property^{Real[y+1]}_{Actor}(Worker_k, Level_k)$ attached to the object instance *Actor* is an array with a length of $y + 1$. The array refers to an one dimensional row vector storing numbers. y refers to the amount of time factors of operating tasks in y difficulty levels, which is stated in Section 3.1. Because there are tasks of y difficulty

levels that could be operated for one worker. $y+1$ refers to the y time factors $\Sigma tw_{lev}^{k\sim l}$ and the salary of the worker. The property instance $Property_{Actor}^{String}(Worker_j, Specialty)$ refers to the workers' specialty type. The property instance $Property_{Actor}^{Real[2]}(Worker_j, tw_h)$ refers to an array of time factors. There are two elements of real number in this array. One is the time factor tw_h^a and the other refers to tw_h^b, which are defined in Section 3.1. The formalization (10) is given:

$$\begin{aligned} Graph_{UseCase}(e.3) =:: \{ & \Sigma Object_{UseCase}^{Actor}(e.3, Worker_k), \\ & \Sigma Object_{UseCase}^{UseCase}(e.3, Task_l), \\ & \Sigma Relationship_{UseCase}^{Association}(e.3, Relationship_{T_l-W_k}), \\ & \Sigma Property_{Actor}^{Real[y+1]}(Worker_k, Level_k), \\ & \Sigma Property_{Actor}^{String}(Worker_j, Specialty) \\ & \Sigma Property_{Actor}^{Real[2]}(Worker_j, tw_h), \Sigma R, \Sigma P_i \}. \end{aligned} \tag{10}$$

The original scheduling scheme is also described by Use Case Diagram and Parameter Diagram. The Use Case Diagram f.1 in Fig. 2 describes worker grade-levels and the association between workers and tasks in the original production scheduling scheme. If a worker operates a task, the relationship between the worker and the task is defined as *association*. The formalization (11) is expressed:

$$\begin{aligned} Graph_{UseCase}(f.1) =:: \{ & \Sigma Object_{UseCase}^{Actor}(f.1, Worker_j), \\ & \Sigma Object_{UseCase}^{UseCase}(f.1, Task_i), \\ & \Sigma Relationship_{UseCase}^{Association} \\ & \quad \times (f.1, Relationship_{T_i-W_j}), \\ & \Sigma Property_{Actor}^{String}(Worker_j, Level), \\ & \Sigma Property_{Actor}^{Real[y+1]}(Worker_j, Level_k), \\ & \Sigma Property_{Actor}^{String}(Worker_j, Specialty), \\ & Property_{Actor}^{Real[2]}(Worker_j, tw_h), \Sigma R, \Sigma P_i \}. \end{aligned} \tag{11}$$

In the above formalization (11), $Property_{Actor}^{String}(Worker_j, Level)$ refers to the grade-level of worker j, and $Property_{Actor}^{Real[y+1]}(Worker_j, Level_k)$ refers to the salary and time factors of the worker j in the grade-level k. The $Property_{Actor}^{String}(Worker_j, Speciality)$ and $Property_{Actor}^{Real[2]}(Worker_j, tw_h)$ refers to the worker's specialty and corresponding time factors. The *association* between workers and tasks is defined.

Parameter Diagram f.2 describes the parameter configuration for the processing time of one product and the worker salary, which are represented by the model component *Parameter*, and *Constraint Block*. Its formalization (12) is as follows:

$$\begin{aligned} Graph_{Parameter}(f.2) =:: \{ & \Sigma Object_{Parameter}^{ConstraintBlock}(f.2, Blocks), \\ & \Sigma Object_{Parameter}^{Parameter}(f.2, Operators), \\ & \Sigma Relationship_{Parameter}^{Connector}(f.2, Line_{Connector}), \\ & \Sigma Property_{Parameter}^{Real}(Parameters, value), \Sigma R, \Sigma P_i \}. \end{aligned} \tag{12}$$

Schedule space definition:. Using the information of the above models from Fig. 2 a to f, all possible production scheduling schemes are defined, referring to a scheduling space, which is described by the Use Case Diagram. The interrelationships among requirement definition, functional structure, logical structure, and scenarios definition are defined by a Package Diagram.

Package Diagram g.1 in Fig. 2 describes the interrelationships among the built models. Relationship instances *dependency* which state the one model is built relying on another model exist between the models in scenario definition stage and the models in the requirements definition stage, the functional structure stage and the logical structure stage. The formalization (13) is as follows:

$$\begin{aligned} Graph_{PackageDiagram}(g.1) =:: \{ & \Sigma Object_{PackageDiagram}^{Package}(g.1, PackageName), \\ & \Sigma Object_{PackageDiagram}^{Model}(g.1, ModelName), \\ & \Sigma Relationship_{PackageDiagram}^{Dependency} \\ & \quad \times (g.1, Line_{Dependency}), \\ & \Sigma Relationship_{PackageDiagram}^{Association} \\ & \quad \times (g.1, Line_{Association}), \Sigma R, \Sigma P_i \}. \end{aligned} \tag{13}$$

Use Case Diagram h.1 in Fig. 2 defines the scheduling space by describing the tasks that workers of different grade-levels perform. The formalization (14) is as follows.

$$\begin{aligned} Graph_{UseCase}(h.1) =:: \{ & \Sigma Object_{UseCase}^{Actor}(h.1, Worker_j), \\ & \Sigma Object_{UseCase}^{UseCase}(h.1, Task_i), \\ & Relationship_{UseCase}^{Association}(h.1, Relationship_{T_i-W_j}), \\ & Property_{Actor}^{Real[y+1]} \\ & (Worker_j, Level_k)', Property_{Actor}^{Real[2]}(Worker_j, tw_h)' \\ & Property_{Association}^{int}(Relationship_{T_i-W_j}), \\ & AssignValue', \Sigma R, \Sigma P_i \}. \end{aligned} \tag{14}$$

In the formalization (14), the properties in Use Case Diagram h.1 are variables, which means that the value of the property is not defined and the property value need to be determined after the optimization process. The possible value of the variable property can be formalized as a set including all possible values, which means that the property value can be any in the property set. After the optimization process, the specific property value can be determined from the property set. To distinguish the property with an uncertain value in the property set and the specific defined property, the property with the uncertain value is labeled with an apostrophe ' as the superscript. In the formalization (14), the $Property_{Actor}^{Real[y+1]}(Worker_j, Level_k)'$ refers to the variable property instance with possible value of differential time factors $\Sigma tw_{lev}^{k\sim l}$ and salaries $Salary_k$, which can be extended as the equation L of formalization (16). The variable property instance $Property_{Actor}^{Real[2]}(Worker_j, tw_h)'$ with possible value of time factors tw_h^a and tw_h^b is formalized as the equation S of formalization (16). The value of property instance $Property_{Association}^{int}(Relationship_{T_i-W_j}), AssignValue'$ referring to the association of the tasks and workers is not defined and the possible value set is as formalization (15).

In the equation L of formalization (16), the possible values of worker grade-level are defined. The symbol $\in$ refers to the connection of the property variable and the probable value set of the property. The $Property_{Actor}^{Real[y+1]}(Worker_j, Level_k)'(k = 1, 2, 3...x)$ is the variable property instance with uncertain value in property set. The possible values are from the time factors and salaries of different grade-levels of workers. The property set range is x, because the amount of the worker grade-levels is defined as x in Section 3.1. The formalization in the brace are the possible worker grade-level.

In the equation S of formalization (16), the variable property instances of time factor tw_h showing workers' specialties is defined. According to Section 3.1, the final working time need be multiplied with a time factor tw_h^a when the workers with the specialty h operates the task of matching type h. In contrast, the time factor tw_h^b is multiplied if the worker of specialty h does not operate the task of matching type h. Different specialties of workers defined different time factor tw_h so there are w possible variables.

$$Property_{Association}^{int}(Relationship_{T_i-W_j}), AssignValue' \in \{0, 1\}. \tag{15}$$

In the formalization (15), the property of the model connection *Association* that connects the worker j and task i is specified. The property

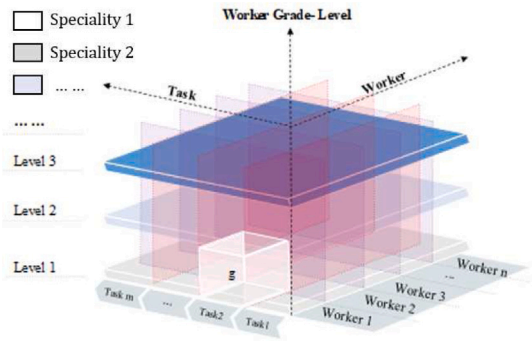

Fig. 3. Scheduling space.

defines whether the specific worker j operates the specific task i or not. The size of property set range is 2 where the probable value is 0 or 1. If the value is 0, the property shows that the worker j does not operate task i and if worker j operates task i, the value is 1.

W, T, S and L in the formalization (16) are the symbols of the worker set, the task set, the specialty set and the worker grade-level set, respectively. A four-dimensional scheduling space is created based on the three sets, depicted in Fig. 3 whose plane axes, z-axis and the cell's color refer to the workers, tasks, grade-levels of worker and the specialty. The range of the four sets considering the h.1 Use Case Diagram is described in the following formalization (16).

$$\begin{aligned}
W =&\{Object^{Actor}_{UseCase}(h.1, Worker_1),\\
&Object^{Actor}_{UseCase}(h.1, Worker_2), \dots, Object^{Actor}_{UseCase}(h.1, Worker_n)\}\\
T =&\{Object^{Block}_{UseCase}(h.1, Task_1),\\
&Object^{Block}_{UseCase}(h.1, Task_2), \dots, Object^{Block}_{UseCase}(h.1, Task_m)\}\\
S =&Property^{Real[2]}_{Actor}(Worker_j, tw_h)'\\
&= \{Property^{Real[2]}_{Actor}(Worker_j, tw_1), Property^{Real[2]}_{Actor}(Worker_j, tw_2),\\
&\cdots, Property^{Real[2]}_{Actor}(Worker_j, tw_w)\}\\
L =&Property^{Real[y+1]}_{Actor}(Worker_j, Level_k)'\\
&= \{Property^{Real[y+1]}_{Actor}(Worker_j,\\
&Level_1), Property^{Real[y+1]}_{Actor}(Worker_j, Level_2),\\
&\dots, Property^{Real[y+1]}_{Actor}(Worker_j, Level_x)\}.
\end{aligned} \tag{16}$$

In the formalization (17), G refers to a method for selecting model elements in the formalization (16). Meanwhile g represents the *association* among the workers, tasks, specialties of workers and the grade-levels of workers, and the value of g is either 0 or 1. Taking the white cube in Fig. 3 as an example, if the g value is 1, the worker 1 of grade-level 1 with specialty 1 performs the task 2.

$$\begin{aligned}
g =&G(Object^{Actor}_{UseCase}(ProductionLineUseCase, Worker_j),\\
&Object^{Block}_{UseCase}(ProductionLineUseCase, Task_i)\\
&Property^{Real[2]}_{Actor}(Worker_j, tw_h)', Property^{Real[y+1]}_{Actor}(Worker_j, Level_k)').
\end{aligned} \tag{17}$$

3.3.2. Formalization of generating the production scheduling scheme based on Satisfiability Modulo Theories

The optimal production scheduling scheme in the scheduling space defined in Section 3.3.1 can be generated by exploring the scheduling space according to the defined objectives and constraints of the scheduling. With the KARMA language formalizing the model elements, the syntax of KARMA are extended and combined with SMT to support property verification [49] for quantification and evaluation of the schemes. SMT refers to a type of logic theory containing a series of axioms, which is able to check whether the first-order logic formulas constructed based on one or more mathematical theories can be satisfied [50]. The included modulo mathematical theories in SMT enrich the high-level information in constraints expression such as the Array theory which can directly describe the array definition and related operations in first-logical expression. The integrated mathematical theories in SMT include Uninterpreted Function Theory, Arithmetic Theory, Difference Logic Theory Bit-vector Theory and so on. The modulo algorithms based on the mathematical theories are combined with the solver based on Boolean satisfiability in corresponding SMT solver to check the satisfiability of first-order expressions. The syntax and semantics of KARMA language is extended according to SMT-LIB [51] to enable KARMA to check satisfiability. Through the extended syntax, the logical constraints can be defined using KARMA. There are main theories of general interest with production scheduling which include theories of Array, Difference Logic and Integer Arithmetic and Real Arithmetic.

Array Theory refers to be applied in array data structures. There are two main operations in Array Theory. One is $select(Array, k)$ which is to select the kth element in the specified Array. The other is $store(Array, k, val)$ which is to assign the kth element in the specified Array to the value of val. In this theory, the main function symbols are $select$ and $store$. The signature set of Array Theory includes the variables including array set, index set, value set and the function symbols. The main axioms which should be satisfied are as following axioms (18), (19), (20).

$$if\ i_{index} = j_{index}, select(store(arr, i_{index}, value), j_{index}) = value \tag{18}$$

$$if\ i_{index} \neq j_{index}, select(store(arr, i_{index}, value), j_{index}) = select(arr, j_index) \tag{19}$$

$$\forall i_{index}, select(arr, i_{index}) = select(arr', i_{index}) => arr = arr' \tag{20}$$

In the above axioms, the arr and arr' refers to arbitrary array while i_{index} and j_{index} are the index of the arrays and $value$ refers to the value of one array element.

Difference Logic Theory can be applied in arithmetic theory without quantifier. The corresponding axiom can be expressed as following (21). In this theory, the signature set includes the key function symbol minus sign, the comparison operators $\circledcirc$, the arithmetical variable and constant.x_1 and x_2 refer to arithmetical variables and J refers to a constant.

$$x_1 - x_2 \circledcirc J, \circledcirc \in \{=, \geq, \leq, \neq\} \tag{21}$$

Integer Arithmetic and Real Arithmetic Theory consists of the function symbols of arithmetic operators, comparison operators, and integer set or real number set. The axioms of Integer Arithmetic and Real Arithmetic can be formalized as the followed expression. The c_i refers to an integer constant and the x_i refers to an integer variable. The signature set of Real Arithmetic Theory are the same except the real number set is instead of integers.

$$c_1x_1 + c_2x_2 + \cdots c_ix_i + ..c_nx_n \circledcirc J, \circledcirc \in \{=, \geq, \leq, \neq\} \tag{22}$$

When exploring the scheduling space, the constraints and objectives of scheduling based on the SMT are listed to generate a group of optimal or sub-optimal schemes that satisfy the constraints. The properties of the models and other elements are captured to construct the constraint formalization (23), which has a return value of Boolean, through the mathematical operators of SMT.

$$C^{num}_{Bool} = \odot(\Sigma Property^{ProType}_{otherType}(otherName, ProName), R). \tag{23}$$

C_{Bool} refers to a unified description of constraints with model properties, the various numeric and operators. $Bool$ indicates that the return

type of the expression is Boolean. The superscript *num* of the C_{Bool} in the formalization (23) presents the order number of constraints. $\odot$ refers to an operation process based on SMT that supports the mathematical expressions based on the Array theory, Difference Logical Theory and Arithmetical Theory of SMT, which means the $\odot$ operation obey the axioms of SMT. When the $\odot$ operation processed, the function symbols of such theories including *select*, *store*, operators and the predicate symbols are applied. The $\Sigma Property^{ProType}_{otherType}(otherName, ProName)$ refers to the set of property instances. The R refers to the numeric elements. It is not essential to define all of the three operated components in one constraint.

Combined with the formal production scheduling models, the first constraint can be expressed as shown in the constraint formalization (24). There are only two probable value 0 and 1 defined for $Property^{int}_{Association}(Relationship_{T_I-W_J}, AssignValue)'$, which is consistent with the definition in the formalization (15). It is stated that the value of $Property^{int}_{Association}(Relationship_{T_I-W_J}, AssignValue)'$ is 1 if the worker J operates the task I. If the worker J does not operate the task I, the corresponding property value is 0. The property of the constraint formalization (24) refers to the property $Property^{int}_{Association}(Relationship_{T_i-W_j}), AssignValue'$in the i.1 graph.

$$\begin{aligned} C^1_{Bool} &= \odot(\Sigma Property^{int}_{Association}(Relationship_{T_i-W_j}, AssignValue)', 0, 1) \\ &= (Property^{int}_{Association}(Relationship_{T_i-W_j}, AssignValue)' == 0) \\ &\quad \| \ (Property^{int}_{Association} \\ &\quad (Relationship_{T_i-W_j}, AssignValue)' == 1). \end{aligned} \tag{24}$$

This constraint formalization (24) includes property values, numeric 0,1 values, the equivalent operator == for connecting 2 exactly equal expressions, and Boolean OR operator ‖ for OR logic operation of 2 Boolean expressions. The another constraint that the probable worker grade-levels are defined as following constraint formalization (25) according to the equation L of formalization (16). Correspondingly, the constraint the probable specialty of worker is defined so that the value domain of time factor tw_h is defined in the formalization (26) according to the equation S.

$$\begin{aligned} C^2_{Bool} &= \odot(\Sigma Property^{Real[y+1]}_{Actor}(Worker_j, Level_k)', Property^{Real[y+1]}_{Actor} \\ &\quad \times (Worker_j, Level_1) \\ &\quad Property^{Real[y+1]}_{Actor} \\ &\quad \times (Worker_j, Level_2), \ldots, Property^{Real[y+1]}_{Actor}(Worker_j, Level_x)) \\ &= ((Property^{Real[y+1]}_{Actor}(Worker_j, Level_k)' == Property^{Real[y+1]}_{Actor} \\ &\quad \times (Worker_j, Level_1)) \| \\ &\quad (Property^{Real[y+1]}_{Actor}(Worker_j, Level_k)' == Property^{Real[y+1]}_{Actor} \\ &\quad \times (Worker_j, Level_2)) \|, \ldots, (Property^{Real[y+1]}_{Actor} \\ &\quad \times (Worker_j, Level_k)' == Property^{Real[y+1]}_{Actor} \\ &\quad \times (Worker_j, Level_x))) \end{aligned} \tag{25}$$

$$\begin{aligned} C^3_{Bool} &= \odot(\Sigma Property^{Real[2]}_{Actor}(Worker_j, tw_h)', Property^{Real[2]}_{Actor}(Worker_j, tw_1) \\ &\quad Property^{Real[2]}_{Actor}(Worker_j, tw_2), \ldots, Property^{Real[2]}_{Actor}(Worker_j, tw_w)) \\ &= ((Property^{Real[2]}_{Actor}(Worker_j, tw_h)' == Property^{Real[2]}_{Actor} \\ &\quad \times (Worker_k, tw_1)) \| \\ &\quad (Property^{Real[2]}_{Actor}(Worker_j, tw_h)' == Property^{Real[2]}_{Actor} \\ &\quad \times (Worker_k, tw_2)) \|, \ldots, (Property^{Real[2]}_{Actor} \\ &\quad \times (Worker_j, tw_h)' == Property^{Real[2]}_{Actor} \\ &\quad \times (Worker_k, tw_w))) \end{aligned} \tag{26}$$

Then, the constraint that each task is performed only once, which is defined in Section 3.1, can be expressed as the following constraint formalization (27) where the big $\sum$ refers to the symbol of summation:

$$\begin{aligned} C^4_{Bool} &= \odot(\Sigma Property^{int}_{Association}(Relationship_{T_i-W_j}, AssignValue)', 1) \\ &= (\sum_{j=1}^{n} Property^{int}_{Association}(Relationship_{T_i-W_j}, AssignValue)' == 1). \end{aligned} \tag{27}$$

There is an adding suppose that all the stated workers in the production system are not redundant, which means that each worker should operate one task at least. The constraint that each worker performs at least one task can be specified as the following constraint formalization (28): .

$$\begin{aligned} C^5_{Bool} &= \odot(\Sigma Property^{int}_{Association}(Relationship_{T_i-W_j}, AssignValue)', 1) \\ &= (\sum_{i=1}^{m} Property^{int}_{Association} \\ &\quad (Relationship_{T_i-W_j}, Assign)' \geq 1) \end{aligned} \tag{28}$$

Finally, the constraint of task priority can be specified as the following formalization (29):

$$\begin{aligned} C^6_{Bool} &= \odot(\Sigma Property^{int}_{Association}(Relationship_{T_i-W_j}, AssignValue)', i, j, 0) \\ &= (\sum_{j=1}^{n}(jProperty^{int}_{Association}(Relationship_{T_b-W_j}, Assign)' - \\ &\quad jProperty^{int}_{Association}(Relationship_{T_a-W_j}, Assign)') \geq 0). \end{aligned} \tag{29}$$

The task a precedes the task b. If there is no conflict among the constraints, they can be satisfied. Based on the satisfied constraints, the scheduling goal O_{num}, which combines the numeric operators and optimization operators, can be expressed using SMT. O_{num} refers to the scheduling objective, which is specified as the following objective formalization (30):

$$O_{num} = M(\odot_{num}(Property^{ProType}_{otherType}(otherName, ProName), R)). \tag{30}$$

O_{num} is the objective that maximizes or minimizes an arithmetic expression, where the symbol M refers to the optimization operator set, including minimization operator $Min()$ and maximization operator $Max()$. The subscript *num* attached to the O shows the order number of objectives. The $\odot_{num}$ refers to an numerical operation process that supports the arithmetic expression based on SMT. When the $\odot_{num}$ operation is processed, the function symbols including *select*, *store*, arithmetic operators and the comparison operators are applied.

After constraints defined, the production scheduling objectives are defined according to a real situation. For example, the objectives are supposed to minimize the worker salary and the processing time of one product, and maximize the production efficiency. The first sub-objective of minimizing the sum of workers' salary can be expressed as the following objective formalization (31).

$$\begin{aligned} O_1 &= Min(\odot_{num}(\Sigma Property^{Real[y+1]}_{Actor}(Worker_j, Level_k)', y+1)) \\ &= Min(\sum_{j=1}^{n}(Property^{Real[y+1]}_{Actor}(Worker_j, Level_k)' \\ &\quad \cdot select(Property^{Real[y+1]}_{Actor}(Worker_j, Level_k), y+1))). \end{aligned} \tag{31}$$

The above formula is the minimum expression of the worker salary. $Min()$ is the minimization operator that returns the minimum in brackets. According to the formalization (10), the salary of the worker j of the grade-level k is stored as the last value of the array property model

$Property_{Actor}^{Real[y+1]}(Worker_j, Level_k)'$. To get the salary, the operation of *select* is used to get the $y+1^{th}$ value which refers to the value of salary.

The second sub-objective to minimize the total processing time for one product can be expressed as the following objective formalization (32). The total processing time for one product refers to the sum of the processing time of each worker, which is a period for one product being operated from the first process to the last process.

$$\begin{aligned}O_2 =& Min(\odot_{num}(\Sigma Property_{Actor}^{Real[y+1]}(Worker_j, Level_k)',\\ & \Sigma Property_{Association}^{int}(Relationship_{T_i-W_j},\\ & AssignValue)', \Sigma Property_{Block}^{int}(Task_i, StandardTime),\\ & Property_{Actor}^{Real[2]}(Worker_j, tw_h)',\\ & v_{i-l}[y+1]^T, u_h[2]^T))\\ =& Min(\sum_{j=1}^{n}(\sum_{i=1}^{m} Property_{Actor}^{Real[y+1]}(Worker_j, Level_k)' \cdot v_{i-l}[y+1]^T\\ & \cdot Property_{Association}^{int}\\ & (Relationship_{T_i-W_j}, AssignValue)'\\ & \cdot Property_{Block}^{int}(Task_i, StandardTime) \cdot Property_{Actor}^{Real[2]}\\ & (Worker_j, tw_h) \cdot u_h[2]^T)).\end{aligned} \tag{32}$$

The above expression refers to the sum of the processing time for n workers to complete the assigned tasks for one product. The processing time for one worker is the sum of processing time for each operated task as the $\sum_{i=1}^{m}$ in formalization (32). The processing time for each operated task is the product of the standardized time of the task, the time factor $tw_{lev}^{k\sim l}$ of the corresponding task difficulty level and the time factor tw_h(defined in Section 3.1). The standardized time of the task i is represented as the $Property_{Block}^{int}(Task_i, StandardTime)$, which refers to the standard time of operating task i in e.1 Package Diagram of Fig. 2. As for the time factor, the time factors of different task difficulty level are stored with the worker's salary in the property $Property_{Actor}^{Real[y+1]}(Worker_j, Level_k)'$(defined in the formalization (10)). The first y value of the property are matched to the time factors of operating tasks in y different difficulty levels. Which specific time factor of the y factors is the selected one is relevant to the difficulty level of the operated task. To determine the time factor, $Property_{Actor}^{Real[y+1]}(Worker_j, Level_k)'$ provides the possible time factors for the worker j of the grade-level k, $Property_{Association}^{int}(Relationship_{T_i-W_j}, AssignValue)'$ is to specify whether the worker j operates the task i, $v_{i-l}[y+1]^T$ is the matching factor to specify which time factor is matched to the corresponding task. $v_{i-l}[y+1]^T$ is a column vector of length $y+1$ for determining the association between the task i and the task difficulty level l, where the T is a symbol to mark the vector as a column vector. It is supposed that the difficulty level of task I is L level, the Lth value of the matching factor $v_{i-l}[y+1]^T$ is 1, and the others are 0. The time factor corresponding to the task I that should be used is the product of the matching factor $v_{I-L}[y+1]^T$ and the property $Property_{Actor}^{Real[y+1]}(Worker_j, Level_k)$. As for the time factor tw_h, which is from the property instance $Property_{Actor}^{Real[2]}(Worker_j, tw_h)$ including the time factor tw_h^a and tw_h^b, the specified factor need to be selected by the column vector $u_h[2]^T$. The column vector $u_h[2]^T$ including 0 and 1 is to define whether the worker of specialty h operates the task of h type.

The maximization of production efficiency can be represented by the minimization of balance delay time. The balance delay time is the amount of idle time on the production line, owing to the uneven division of the production process for workers [52]. If the balance delay time decreases, the waiting and idle time will decrease so that the production efficiency will increase. The balance delay time can be calculated as the following Eq. (33). The T_B refers to the balance delay time, n refers to the amount of workers. T_{max} refers to the maximum processing time among n workers. T_P refers to the processing time for one product (defined in the formalization (31)).

$$T_B = n \cdot T_{max} - T_P \tag{33}$$

The formalization of minimizing balance delay time is shown in the following objective formalization (34):

$$\begin{aligned}O_3 =& Min(\odot_{num}(\Sigma Property_{Actor}^{Real[y+1]}(Worker_j, Level_k)',\\ & Property_{Association}^{int}(Relationship_{T_i-W_j}, AssignValue)',\\ & \Sigma Property_{Block}^{int}(Task_i, StandardTime),\\ & Property_{Actor}^{Real[2]}(Worker_j, tw_h)', v_{i-l}[y+1]^T, u_h[2]^T))\\ =& Min(n \cdot max(Property_{Actor}^{Real[y+1]}(Worker_j, Level_k)' \cdot v_{i-l}[y+1]^T\\ & \cdot Property_{Association}^{int}(Relationship_{T_i-W_j}, AssignValue)'\\ & \cdot Property_{Block}^{int}(Task_i, StandardTime)\\ & \cdot Property_{Actor}^{Real[2]}(Worker_j, tw_h) \cdot u_h[2]^T)\\ & - \sum_{j=1}^{n}(\sum_{i=1}^{m} Property_{Actor}^{Real[y+1]}(Worker_j, Level_k)' \cdot v_{i-l}[y+1]^T\\ & \cdot Property_{Association}^{int}(Relationship_{T_i-W_j}, AssignValue)'\\ & \cdot Property_{Block}^{int}(Task_i, StandardTime)\\ & \cdot Property_{Actor}^{Real[2]}(Worker_j, tw_h) \cdot u_h[2]^T)).\end{aligned} \tag{34}$$

The expression in the minimizing symbol $Min()$ of the formalization (34) is constructed according to Eq. (33). $max()$ refers to returning the maximum value in the brackets. The expression in the brackets of the symbol $max()$ represents the processing time for each worker. The expression at the right of the minus sign represents the processing time for one product.

An overall objective of formalization (35) is created with the weighted sum of the other sub-objectives:

$$O_o = \omega_1 \cdot O_1 + \omega_2 \cdot O_2 + \omega_3, \cdot, O_3 + \cdots + \omega_n \cdot O_n. \tag{35}$$

Different weighting values represent the importance of different scheduling objectives. ω_1 is the weight of minimizing workers' salaries, ω_2 is the weight of minimizing the processing time, and ω_3 is the weight of maximizing the production efficiency which also is minimizing the balance delay time.

The Algorithm 1 is proposed to generate the optimal scheduling scheme based on the defined objectives that satisfies the above constraints. And the customized overall objective is constructed according to the specified scheduling space and the defined constraints and objectives. Constraint (24) is added to the algorithm to state the two probabilities of the matching relationship between workers and tasks. Constraint (25) is added to state the domain of worker grade-levels. Constraint (26) is added to state the domain of specialty of workers. Constraint (27) is added to state that each task is performed once. The constraint (28) indicates that each worker must perform a task at least once, and the constraint (29) to define the task processing priority is added. The overall objective (35) of the worker salary, balance delay time, and processing time is added.

4. Case study

4.1. Case description

A production line for packages is shown in Fig. 4, whose data is defined in the thesis [53]. When the production department receives the requirements from the defined master plan, the production scheduling scheme is expected to optimize human resources based on workers' salary, worker capacity, worker specialty, operating sequence, and the standardized time of the operating tasks during COVID-19. According

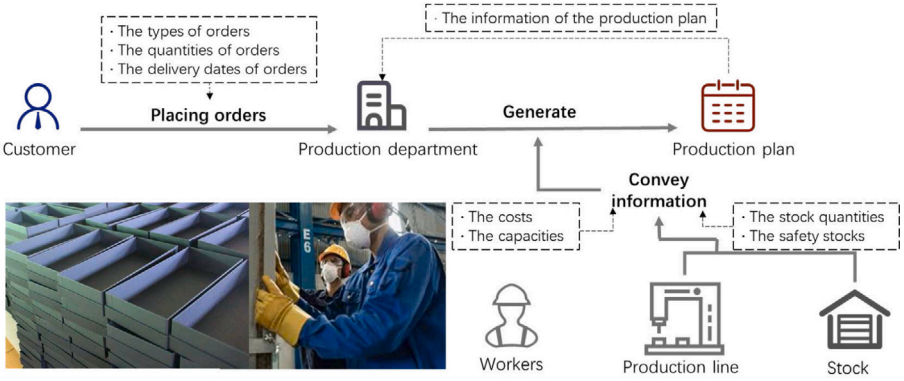

Fig. 4. Production scheduling including human resources.

Algorithm 1: Framework to generate the optimal scheduling scheme.

Input: R: initial the association between each worker and each task;
T: initiate set of tasks;
t: each task in the T task set;
W: initiate set of workers;
w: each worker in the W task set;
L: initiate set of worker grade-level;
ws: initial the specialty of each worker.
wl: initial the grade-level of each worker.

Output: the grade-level of each worker w, the matching relationship of T and W

for *each association* R **do**
  $Add\ Constraints(R \leftarrow 0\ or\ R \leftarrow 1)$
end for
  % add the constraint that there are only two probable association between the tasks and workers.
for each worker grade-level wl **do**
  $Add\ Constraints(wl\ belongs\ to\ L)$
end for
  % add the constraint that the grade-level of each worker should in the defined grade-level set.
for each worker specialty ws **do**
  $Add\ Constraints(ws\ belongs\ to\ S)$
end for
  % add the constraint that the specialty of each worker should in the defined specialty set.
for *each element* t *in the* T **do**
  $Add\ Constraints(t\ is\ performed\ once)$
end for
  % add the constraint that each task is operated once.
for *each element* w *in the* W **do**
  $Add\ Constraints(w\ performs\ one\ task\ at\ least)$
end for
  % add the constraint that each worker should operate a task at least.
for *each element* t *in the* T **do**
  $add\ Constraints(TaskPriority)$
end for
  % add the constraint that the tasks should follow the given operating priority.
$Min(w_1 \cdot Salary + w_2 \cdot BanlanceDelayTime + w_3 \cdot ProcessingTime)$
  % minimize the expression in the brackets, the w_n *refers to weight of the objective formalization 35, the* $Salary$ *refers to the expression of worker's salary, the* $BanlanceDelayTime$ *refers to the expression of balance delay time and the* $ProcessingTime$ *refers to the expression of processing time for one product.*

to the obtained scheme, the tasks are allocated to the production department to be operated. The processed products are sent to the stocks after the entire process.

Owing to COVID-19, workers need to keep adequate distance to minimize their probability of infection when working. Meanwhile, the production department needs to ensure that the cost and productivity are significantly affected by a more health-conscious environment. Therefore, the production department decided to reduce the number of workers in the package production line to increase the distance between workers. In this case, the original production scheduling scheme is optimized to maintain production efficiency and decrease the processing time and worker salary when the number of workers is reduced.

There are 22 tasks to be performed in the package production line, including cutting, carving, drilling, cleaning, painting, assembling and other relevant tasks. The 22 tasks can be classified by the applied specialty. The *machining* tasks using specialties that are used to reduce the material such as cutting, carving, drilling is the first type. The *other* tasks using specialties that are not used to reduce the material such as cleaning, painting and assembling is the second type. The processing priority of 22 tasks is shown in Fig. 5, and the standard working time of each task is shown in Table 1. Besides, the task difficulty levels and the task types using different specialty are shown in the Table 1.

The production requirement is to generate 4800 pieces of box for this package production line in one complete production period (defined as 10 days). It is supposed that the eight-hour work schedule with two shifts is applied, which means the available working time is 16 h for this production line and each worker works for 8 h. Therefore, the takt time, which is the required pace to finish the production process in order to meet customer demand, is 120 s according to the ratio of the amount of the required products and the planned working time, that is the time of finishing one product should not exceed 120 s.

In addition, workers are also divided into three grade-levels according to their proficiency in operating tasks. The three grade-levels are senior (A-level) workers, intermediate (B-level) workers, and junior (C-level) workers. All workers are available to perform tasks of any difficulty level, but the time factors of operating different tasks are different, as shown in Table 2. The salary of different worker for one complete production period (10 days) are listed at the Table 2.

From the table, the senior (A-level) workers are best at performing tasks; however, their required salary is highest. The processing time of the same task by senior workers is shorter than the workers of the other grade-levels. If a senior worker performs the 21th task with standardized working time of 30 s, he typically only require 27 s that is the product of the time factor 0.9 which is a value of operating a moderate task for a senior worker and the task standardized time 30 s. However, it takes 45 s for junior workers to complete the 21th task.

Besides, there are workers of two different specialty with different time factors tw_h. Corresponding to the task types, there are workers of Specialty 1 and the workers of Specialty 2. The time factors of workers

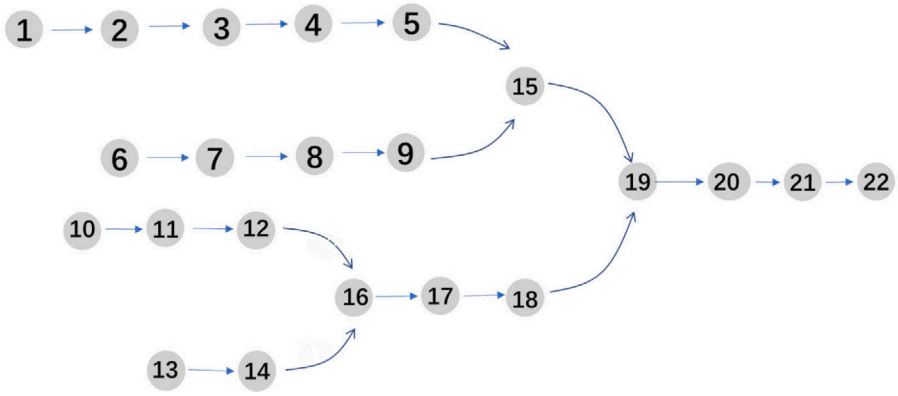

Fig. 5. Procedure priority diagram of a package box.

Table 1
The standard time of operating tasks with the difficulty levels and task types.

Task	Time	Task difficulty level	Type	Task	Time	Task difficulty level	Type
Task1	7	Moderate level (Level 2)	1	Task 12	15	Complex level (Level 3)	1
Task2	26	Simple level (Level 1)	1	Task 13	38	Simple level (Level 1)	2
Task3	15	Moderate level (Level 2)	1	Task 14	19	Simple level (Level 1)	2
Task4	13	Moderate level (Level 2)	2	Task 15	49	Simple level (Level 1)	1
Task5	23	Complex level (Level 3)	1	Task 16	25	Complex level (Level 3)	2
Task6	11	Moderate level (Level 2)	2	Task 17	30	Simple level (Level 1)	1
Task7	9	Simple level (Level 1)	2	Task 18	25	Complex level (Level 3)	1
Task8	18	Moderate level (Level 2)	2	Task 19	58	Simple level (Level 1)	2
Task9	15	Moderate level (Level 2)	2	Task 20	22	Complex level (Level 3)	2
Task10	26	Moderate level (Level 2)	1	Task 21	30	Moderate level (Level 2)	1
Task11	9	Simple level (Level 1)	1	Task 22	51	Simple level (Level 1)	2

Table 2
Subordinate relationship between each worker and their grade-level including the time factors of operating tasks and salaries.

	Simple tasks	Moderate tasks	Complex tasks	Salary (Chinese Yuan)
Senior workers (A)	0.6	0.9	1.1	5200
Intermediate workers (B)	0.9	1.1	1.6	4000
Junior workers (C)	1.1	1.5	2.1	2200

Table 3
The time factors tw_h of different specialties.

Time factor tw_h	Task of type 1	Task of type 2
Workers of specialty 1	0.5	1.8
Workers of specialty 2	1.3	0.7

Table 4
The system environment of MetaGraph.

Parameter	Value
Type	PC
Windows	Windows 10
Processor	Intel(R) Core(TM) i5-1035G1
RAM	16.0GB
System type	64-bit os
Occupied memory	381 MB

with different specialties operating the tasks of same specialty or not are stated in the Table 3. When the workers of specialty 1 operates the task of type 1, the final working time should multiply with the time factor 0.5.

When the company made use of the original plan before the COVID-19 happened, 11 workers were arranged to work and the original scheduling scheme is defined in the site [54]. According to the guidelines for infection prevention and control of the pandemic, maintaining distance between workers is mandatory. The company decides to reduce the team of 11 workers to 8 workers to maintain the minimum-required healthy distance. After reducing the number of workers, the company need to reschedule the workers and tasks to maintain the production demands.

In this case, models describing the scheduling requirements definition, functional structure, logical structure, and production scenarios according to Section 3.3 are developed using KARMA language. Based on these models, the constraints and objectives of production scheduling are described using KARMA, where the property verification is performed to evaluate the constraints to eliminate the inappropriate schemes and generate the optimal scheme.

4.2. Implementation

The process of solving the production scheduling problem is implemented in the software MetaGraph [47] as shown in Fig. 6. MetaGraph [55] is a platform for building heterogeneous models based on KARMA with additional modules for model analysis. The system environment of MetaGraph are listed including the specification of PC and the occupied memory for running the MetaGraph as Table 4.

First, the meta–meta models are instantiated and combined to be meta models according to SysML based on KARMA at the modeling interface of MetaGraph. Besides, the additional properties relevant to the production scheduling are instantiated and attached to the object meta models and relationship meta models. When the meta models completed, the meta models can be combined to be the models. Second, KARMA is used to specify the property models and operate the property models to be the constraints and objectives at the editor interface. The KARMA script should be organized according to the KARMA language specification [56]. Then, the input KARMA script is parsed to construct

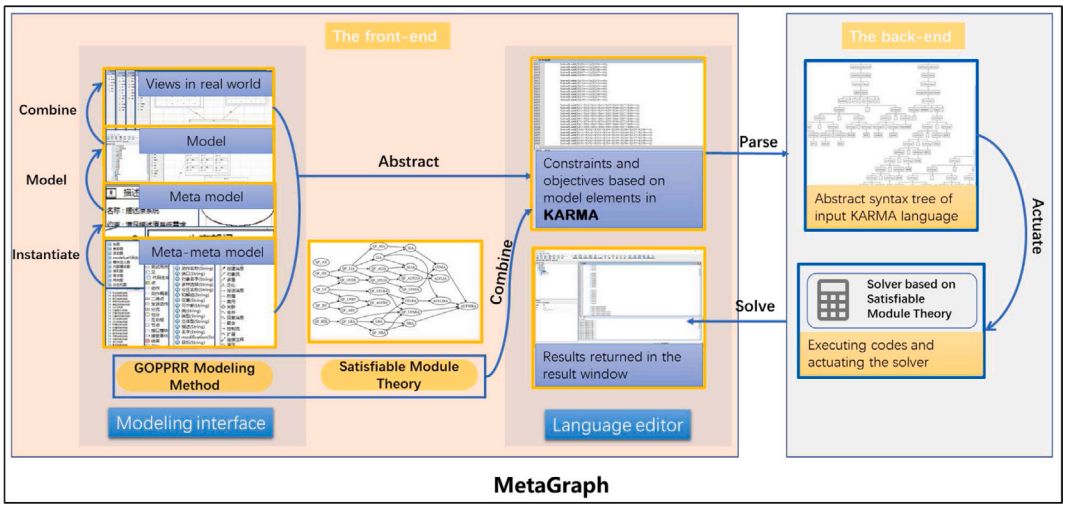

Fig. 6. Implementation of property verification.

and traverse the generated abstract syntax tree in the back-end of MetaGraph. And the corresponding solver based on SMT which is z3 solver in this paper [57] is actuated when the corresponding codes are executed to verify whether there is a solution set that satisfies all constraints. The solution is automatically returned to the result interface of language editor in MetaGraph and displayed to the engineers.

4.3. Production scheduling modeling

According to the implementation method described in Fig. 2, the models for production scheduling are developed in MetaGraph [58]. The SysML model library based on the KARMA language is used to build the models, as shown in Fig. 7. The types of diagrams are listed with abbreviation at the top right-hand corner of each diagram which is labeled as their related graph in Fig. 2.

The first part represents the models for requirement definition in Section 3. Stakeholders, including customers, worker groups, and production planning departments, are defined based on Use Case Diagram (A.1 in Fig. 7). The requirements are defined in requirement diagram (B.1 in Fig. 7) from the perspective of the stakeholders. From the customers' view, the requirement is timely and sufficient delivery. The workers require the appropriate working time and a healthy working environment. The production department requires a production line with high productivity and low cost. The specific requirement list is presented in the following list.

- 8 workers are required to work together for 2 shifts and working hour of each worker should not exceed 8 h per day.
- Workers are not allowed to change the tasks they perform.
- Each task of the 22 tasks must be performed and performed only once.
- The priority of operating tasks must be followed.
- The cycle time of each worker should not exceed the takt time 120 s which is the working pace of production defined in Section 4.1.
- The station of each worker is fixed.
- The scheduling objectives aim to minimize worker salary for 8 workers in one working shift and balance the delay time and total processing time for one product.

The function of the production system is described using Use Case Diagram (C.1 in Fig. 7). The Use Case diagram describes the production process that the production planning department uses the order from the customer to formulate the scheduling scheme to arrange the workers on the production line, deliver produced products to the stock, and deliver the products to the customer after meeting the demands required for the order. The Sequence Diagram (D.1 in Fig. 7) describes the information exchange of the production system. The Activity Diagram (C.2 in Fig. 7) and State Machine Diagram (D.2 in Fig. 7) jointly describe the processing action, state change, and sequence of the 22 tasks in the production process of packing box. The classification of workers and tasks are described using Package Diagrams (E.1 in Fig. 7) separately. The property instances showing the standardized time of operating the task, the time factors and salaries corresponding to the worker grade-level and the time factors tw_h corresponding to the specialty of workers are added in the Package Diagrams (E.1 in Fig. 7). The compositions and the structure of the production system, which include the workers, production line, and the department, are defined in the Block Definition Diagram (E.2 in Fig. 7). Tasks of different difficulty levels performed by workers of different grade-levels and their corresponding time factors are described in the Use Case Diagram (E.3 in Fig. 7). The performance of the allocated workers in the original scheme is defined based on graph F.1 in Fig. 7, and the parameter for optimizations to determine the working time and worker wages is defined in the Parameter Diagram (F.2 in Fig. 7). The process to determine the scheduling space from the requirement definition, the function structure, the logic structure, and the scenario definition is defined in graphs G and H in Fig. 2. When the models are completed, the corresponding constraints and objectives are defined using KARMA in MetaGraph.

4.4. Constraints of production scheduling

The models built in Section 4.3 are used to describe the constraints and objectives of the case on the basis of Section 3. There are supposed to be 8 workers of unknown grade-levels and 22 tasks of 3 difficulty levels to be allocated. Thus, the corresponding constraints in Section 3.3 are defined as follows. Constraints of this case for production scheduling can be generated by replacing the task amount m and the worker amount n in the constraints (24), (27), and (28) with 22 and 8 and specified the workers' grade-levels and specialty.

To satisfy the customer's command timely, another additional constraint that the processing time of each worker is not allowed to exceed the takt time is considered. t_p refers to the takt time, which can be calculated as 120 s from the known information of this case. In this equation, the amount of tasks m, the amount of workers n, the amount of worker grade-level x and the number of task difficulty level y are substituted with 22, 8, 3 and 3. Adding constraint (36) gives the

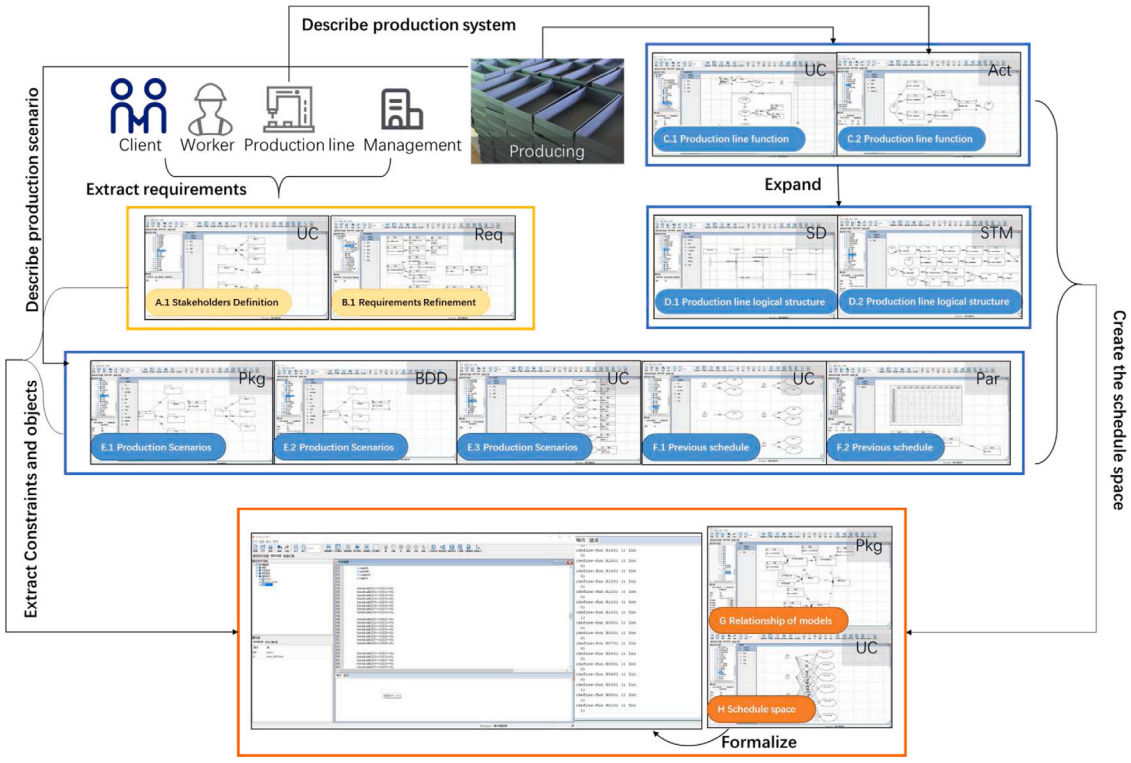

Fig. 7. Model for production scheduling using SysML based on the GOPPRRE modeling method.

Table 5
Weight value of optimization objectives.

	W_S	W_B	W_P
Important	1	100	100
More important	2	200	200
Most important	3	300	300

following:

$$\begin{aligned}\forall j, \sum_{i=1}^{22} & Property_{Actor}^{Real[4]}(Worker_j, Level_k)' \cdot v_{i-l}[4]^T \\ & \cdot Property_{Actor}^{Real[2]}(Worker_j, tw_h) \cdot u_h[2]^T \\ & \cdot Property_{Association}^{int}(Relationship_{T_i-W_j}, AssignValue)' \\ & \cdot Property_{Block}^{int}(Task_i, StandardTime) \leq 120.\end{aligned} \tag{36}$$

Considering the objective formalization (35), the minimizing of the worker salary C_s, the balance delay time T_B, and the processing time for one product T_P can be combined, as shown in objective formalization (37):

$$O_o = W_S \cdot Min(C_s) + W_B \cdot Min(T_B) + W_P \cdot Min(T_P). \tag{37}$$

The weight is divided into "important", "more important", and "most important" according to the corresponding importance level. The specific weights of each objectives are shown in Table 5 according to the true situation.

Considering the multi-objective of the three sub-objectives, C_s and T_P are set as "important", and T_B is set as "more important". The multi-objective (38) is as follows:

$$O_o = Min(C_s + 100 \cdot T_P + 200 \cdot T_B). \tag{38}$$

Moreover, additional constraints to ensure that the worker salary, balance delay time, and total processing time in the new scheme are less than those of the original scheme are added. Finally, the KARMA script, including constraints and objectives from the model information, is defined. The models and the corresponding KARMA script can be found at the opendepository.

4.5. Result

Except for optimizing for one multi-objective (38), more objectives with different weights are used for a comprehensive evaluation of the scheduling schemes. According to the number of scheduling objectives, the scheduling schemes are divided into three categories. Schemes in the same categories are evaluated to obtain the optimal scheme among them. Table 6 shows the different objectives and the corresponding constraints.

The solutions of different schemes with execution time are shown in Fig. 8. The first row refers to the worker number and the execution time of running the programs, the first column refers to the scheme number, and the color represents the type of workers: yellow, blue which the workers with specialty 1 and the workers with specialty 2, respectively. The number in the cell refers to the task number and the grade-levels of workers are specified below the task number. The A level refers to the senior workers, B level refers to the intermediate workers and the C level refers to the junior workers. The last column refers to the execution time of running program for each scheme. The differences of the execution time can be analyzed from the complexity of the constraints, the amount and the label of variables, the order of adding constraints and others.

The Fig. 9 shows the data analysis of the scheduling schemes in Fig. 8. The sub Fig. 9a–c are the results that consider a single objective. The others are the results that consider any two objectives.

It is observed from Fig. 9a that the total processing time of scheme 7 is the shortest at 355.55 s, which is 40.01% lower than that of the original scheme. A minimum worker salary of 17,600 Chinese Yuan for one shift in the production period(10 days) can be observed in Fig. 9b. If the scheme 1 is executed, the worker salary is the least, which is 47.93% lower than that of the original scheme. The minimum production balance time is 13.45 s in scheme 2, that is, the production balance rate of the original scheme is increased from 68.6% to 97.71%.

Table 6
Groups of different scheduling objectives and constraints.

Group	Illustration	Number	C_S	T_P	T_B	Adding constraints
Single objective (Group 1)	T_P and T_B are ignored	1	1	0	0	C_S, T_P and T_B are less than the original value
	C_S and T_P are ignored	2	0	0	1	
2 objectives (Group 2)	T_P is ignored and C_s is set as "important" and T_B as "more important"	3	1	0	100	C_S, T_P and T_B are less than the original value
	T_P is ignored and T_B is set as "more important" and C_S as "important"	4	1	0	200	
	C_S is ignored and T_B and T_P are set as "important"	5	0	100	100	
	C_S is ignored and T_B and T_P are set as "important" and "more important"	6	0	100	200	
3 objectives (Group 3)	C_S, T_P, and T_B are set as "important"	7	1	100	100	C_S, T_P, and T_B are less than the original value
	C_S and T_P are set as "important" and T_B as "more important"	8	1	100	200	
	C_S and T_P are set as "important" and T_B as "most important"	9	1	100	300	

Task	Worker1	Worker2	Worker3	Worker4	Worker5	Worker6	Worker7	Worker8	Running time(s)	NOTES
Schedule1	1,2,3,10 (C level)	4,6,7,8,11,13,14 (C level)	12 (C level)	5,9 (C level)	16 (C level)	15,17,18 (C level)	19,20 (C level)	21,22 (C level)	53.89	Speciality 1
Schedule2	1,10,13 (B level)	2,3,6,11,12 (A level)	4,14 (C level)	16 (B level)	5,7,8,17 (A level)	18 (C level)	9,15,19 (A level)	20,21,22 (A level)	617.61	Speciality 2
Schedule3	1,2,6,7 (C level)	3,4,13 (C level)	10,11,14 (C level)	12,16 (C level)	8,9,17 (C level)	5,15,18 (C level)	19,20 (C level)	21,22 (B level)	806.23	
Schedule4	1,2,3,6,7,10 (B level)	8,9,11,13 (C level)	4,5,12 (C level)	15 (C level)	14,16,17 (B level)	18 (C level)	19,20 (C level)	21,22 (B level)	1584.89	
Schedule5	1,2,6,13,14 (A level)	3,4,5,10,11 (A level)	7,8,12,16 (A level)	9,15,17 (A level)	18 (C level)	19 (C level)	20 (C level)	21,22 (A level)	1018.33	
Schedule6	1,2,3,10,11 (C level)	12,13 (B level)	4,14 (B level)	5,6,7 (B level)	8,9,16 (B level)	15,17,18 (B level)	19,20 (A level)	21,22 (A level)	811.12	
Schedule7	1,2,3,10,11,12 (B level)	4,6,7,8,13,14 (B level)	9,16 (C level)	5,15 (C level)	17,18 (C level)	19 (C level)	20,21 (A level)	22 (C level)	1390.4	
Schedule8	1,6,7,8,9,13 (A level)	2,3,4,10 (A level)	5,11,12,15 (B level)	14,16 (C level)	17 (C level)	18 (B level)	19,20 (A level)	21,22 (A level)	218.95	
Schedule9	10,11,13 (B level)	1,12,14 (C level)	2,16 (C level)	6,17,18 (C level)	7,8 (C level)	3,4,5 (C level)	9,15,19 (A level)	20,21,22 (A level)	1365.47	

Fig. 8. Scheme options.

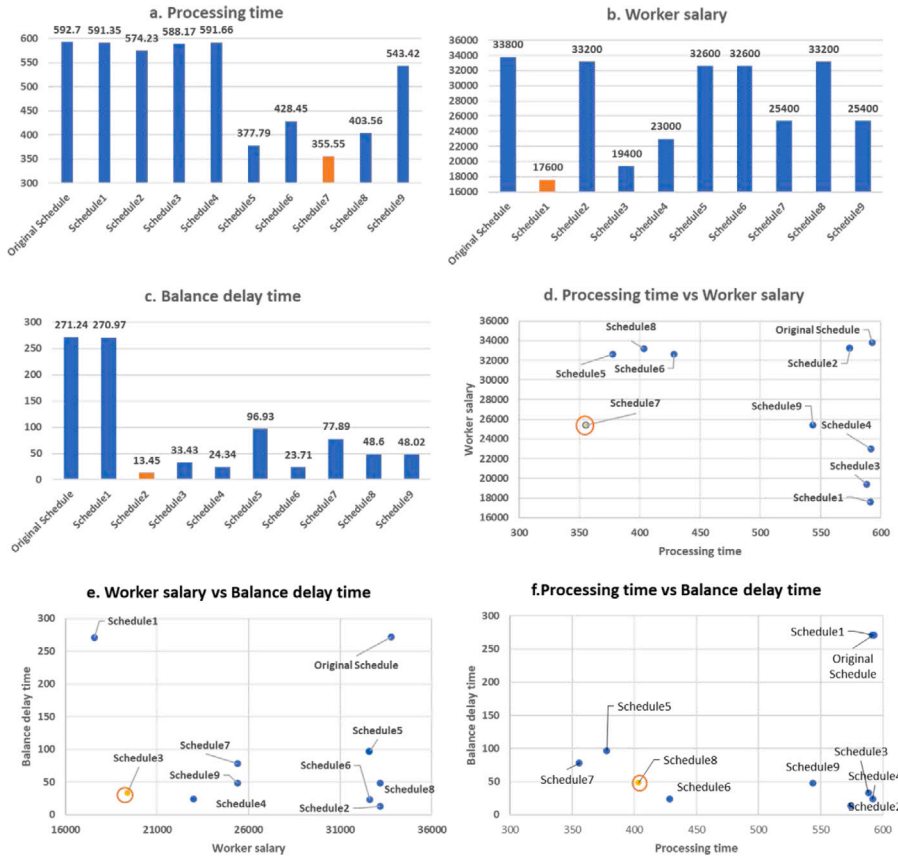

Fig. 9. Optimization results of a single objective and double objective.

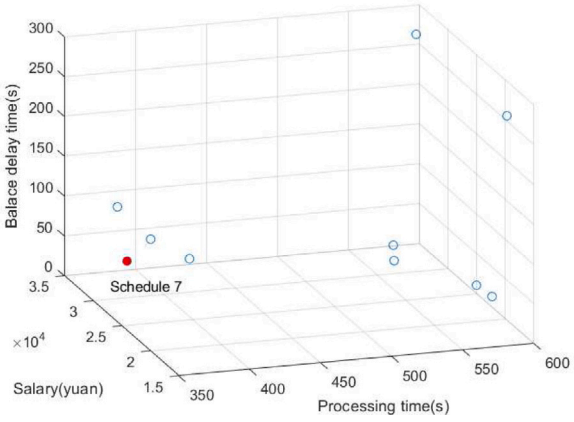

Fig. 10. Three-dimensional scatter diagram considering three objectives.

Table 7
Quantitative analysis form of the case.

Quantitative analysis of models	
Models	20
Stakeholders	10
Scheduling sub objectives	3
Groups of objectives	3
Options of objectives	9
Quantitative analysis of KARMA script	
Formalized models in KARMA	9488 lines
Property verification in KARMA	170 lines

When any two of the three objectives are considered, the ranges of the x and y axes are normalized to one unit and the positions of scatters represent the relative value. If the scatter that refers to a scheme is closer to the origin of the coordinates, the scheme is considered closer to the scheduling objective. In Fig. 9d, the processing time is the x-axis, and the worker's salary is the y-axis. It is observed that the scatter referring to scheme 7 is closest to the origin of the coordinate. Therefore, if T_B is ignored and only T_P and C_S are considered, scheme 7 is optimal. In Fig. 9e, the processing time is the x-axis, and the balance delay time is the y-axis. It is observed that the scatter that refers to scheme 3 is closest to the origin of the coordinate, which means that if C_S is ignored and only T_B and C_P are considered, scheme 3 is optimal. In Fig. 9f, when the worker salary is the x-axis, and the balance delay time is the y-axis, scheme 8 is optimal if the T_P is ignored.

When three objectives are considered, the three-dimensional scatter diagram is produced, as shown in Fig. 10. The ranges of the x-axis and the y-axis and the z-axis are normalized to one unit. The positions of scatters show the relative value. The relative distance of the scatter to the coordinate origin is calculated, and the nearest scatter is considered optimal, which is scheme 7.

5. Discussion

In the case study, an MBSE method is applied to optimize a production scheduling during COVID 19, and the expression of the graphic models and the optimization descriptions of production scheduling is described by one unified language KARMA. Based on such KARMA models, property verification is performed to obtain an acceptable scheduling scheme.

In Table 7, quantitative analysis is shown, including the numbers of models and KARMA language lines. Ten stakeholders, including eight workers who participate in production, the production department, and customers, are involved in this case. Twenty models are used to describe the scheduling problem, including the requirement definition, functional structure, logical structure, scenario definition, and scheduling space description. There are 9488 lines of KARMA language to describe the 20 models. The 9 options for the objectives are divided into 3 groups with different weights. Finally, 170 lines of KARMA language are used to describe the constraints and objectives of the production scheduling for performing property verification.

From a qualitative perspective, compared with other approaches to production scheduling optimizations, the given method is used as the basis to describe the production contents and constraints for optimizations. KARMA language is developed based on a GOPPRRE modeling theory with a high level of abstraction that supports the specification of meta-models and models for different domains. Moreover, using KARMA models to describe the production contents can enhance the ability to express scenarios and reduce the inconsistency of communication through graphic descriptions. The combination of the GOPPRRE modeling method and SMT is used for defining the specification of constraints and the model information which integrate modeling and simulation.

To consider the scalability of the approach, KARMA language is developed based on the GOPPRRE meta–meta modeling method, which is the most powerfully expressive meta–meta modeling method because GOPPRR supports the most first class modeling concepts and the most relationship patterns [46]. Moreover, the constraints and objectives based on SMT are the expressions involving complex mathematical theories with the Boolean returning type, which can be extensively expressive for most applications. Therefore, this approach, which widely supports the unified expression of models in multiple domains and different constraints, provides a good scalability.

The internal validity of the method is analyzed from both the qualitative and quantitative aspects. First, we propose an approach to specify graphic models and optimization in a unified way to improve the capability of the production scheduling based on KARMA. Second, the constraints of the models of production scheduling is defined with KARMA, which is used to obtain the optimal scheme when analyzing the constraints with defined objectives. The graphic expression of the approach helps stakeholders to obtain a unified understanding and expression of the models. The generated scheduling schemes of the case study showing the internal validity of this approach. The proposed approach can improve the capacity of information management. However, the limitation of this proposed approach is obvious that the speed of solving scheduling problems is slow because of the limitation of the SMT solver such as the constraints complexity, the amount of variables in the KARMA script and others. Moreover, another limitation is the uncertain external validity for this approach for the limited and atypical case. About the future work, we will focus on the more examples including mass production applied in the real industrial situation. And the application of SMT solvers with vectors to be dimensionally reduced to facilitate computation from the statistics aspect will be the future working emphasis [59].

6. Conclusions

We proposed an approach to generate the optimal production scheduling scheme based on MBSE combining with SMT for stakeholders in different domains. The approach based on MBSE is capable of modeling the production scenarios in a graphic way and unifying the expression of modeling and solving. First, the production scheduling is formalized with models based on KARMA. Second, the constraints of production scheduling are specified using KARMA to capture related attributes from the models. Third, an SMT solver is executed based on KARMA to generate the final scheduling scheme. The proposed approach is evaluated using a package production line under the special conditions of the COVID-19 pandemic. The case study shows that the tasks are re-allocated to different workers to maintain the production efficiency and cost when the number of workers decreases. Nine schemes are generated according to different constraints and

optimizing objectives using the proposed approach. The validity of the approach is proven via the case study. From the case study, we identify that KARMA is competent as a unified specification to build graphic models in multiple domains and to describe constraints related to the models and solving processes. Such unified graphic expression is helpful to enable different stakeholders to understand and manage production scheduling in an expressive way. The formalized expression of the production scheduling is available for reducing communication ambiguity, which is helpful to promote the efficiency of management and production. Moreover, the unified expression of KARMA is able to interact models with SMT executions, which enables the management of the information of models and the solving process.

CRediT authorship contribution statement

Jingqi Chen: Methodology, Investigation, Resources, Software, Validation, Formal analysis, Data curation, Project administration, Visualization, Writing – original draft. **Guoxin Wang:** Writing – review & editing, Project administration, Resources, Funding acquisition. **Jinzhi Lu:** Conceptualization of this study, Formal analysis, Resources, Project administration, Writing – review & editing, Funding acquisition, Supervision. **Xiaochen Zheng:** Writing – review & editing. **Dimitris Kiritsis:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by The National Key Research and Development Program of China (Grant No. 2020YFB1708100) and the CN pre-study common technology (No. 50923010101).

References

[1] D.A. Rossit, F. Tohmé, M. Frutos, A data-driven scheduling approach to smart manufacturing, J. Ind. Inf. Integr. 15 (2019) 69–79, http://dx.doi.org/10.1016/j.jii.2019.04.003.

[2] Y. Cheng, F. Sun, Y. Zhang, F. Tao, Task allocation in manufacturing: A review, J. Ind. Inf. Integr. 15 (2018) http://dx.doi.org/10.1016/j.jii.2018.08.001.

[3] T. Ruppert, J. Abonyi, Integration of real-time locating systems into digital twins, J. Ind. Inf. Integr. 20 (2020) http://dx.doi.org/10.1016/j.jii.2020.100174.

[4] ISO/IEC/IEEE 15288:2015 - Systems and software engineering — System life cycle processes, URL https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/06/37/63711.html.

[5] J. Lu, A Framework for Cyber-Physical System Tool-Chain Development :A Service-Oriented and Model-Based Systems Engineering Approach (Ph.D. thesis), 2019.

[6] ISO/IEC/IEEE DIS 24641 - Systems and Software engineering — Methods and tools for model-based systems and software engineering, URL https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/07/91/79111.html.

[7] C.W. Barrett, C. Tinelli, Satisfiability modulo theories, in: Handbook Of Model Checking, 2018.

[8] L. John Wiley & Sons, Introduction, in: Principles of Sequencing and Scheduling, 2009, pp. 1–9, http://dx.doi.org/10.1002/9780470451793.ch1.

[9] S.C. Graves, A review of production scheduling, Oper. Res. 29 (4) (1981) 646–675, http://dx.doi.org/10.1287/opre.29.4.646.

[10] D.N. Burghes, "Sequencing and scheduling." by S. French. (Ellis Horwood ltd., 1982.) [pp. 245.] price £15-50., Int. J. Prod. Res. 22 (3) (1984) 532, http://dx.doi.org/10.1080/00207548408942474.

[11] W.J. Davis, A.T. Jones, A real-time production scheduler for a stochastic manufacturing environment, Int. J. Comput. Integr. Manuf. 1 (2) (1988) 101–112, http://dx.doi.org/10.1080/09511928808944350.

[12] M. Zentner, J. Pekny, G. Reklaitis, J. Gupta, Practical considerations in using model-based optimization for the scheduling and planning of batch/semicontinuous processes, J. Process Control 4 (4) (1994) 259–280, http://dx.doi.org/10.1016/0959-1524(94)80046-4.

[13] N.F. Giannelos, M.C. Georgiadis, Efficient scheduling of consumer goods manufacturing processes in the continuous time domain, Comput. Oper. Res. 30 (9) (2003) 1367–1381, http://dx.doi.org/10.1016/S0305-0548(02)00076-X.

[14] J.H. Blackstone, D.T. PHILLI.P.S., G.L. Hogg, A state-of-the-art survey of dispatching rules for manufacturing job shop operations, Int. J. Prod. Res. 20 (1) (1982) 27–45, http://dx.doi.org/10.1080/00207548208947745.

[15] X. Zhao, J. Xie, Q. Jiang, LOT-sizing rule and freezing the master production schedule under capacity constraint and deterministic demand, Prod. Oper. Manag. 10 (1) (2001) 45–67, http://dx.doi.org/10.1111/j.1937-5956.2001.tb00067.x.

[16] Y. Ma, F. Qiao, F. Zhao, J.W. Sutherland, Dynamic scheduling of a semiconductor production line based on a composite rule set, Appl. Sci. 7 (10) (2017) http://dx.doi.org/10.3390/app7101052.

[17] S. Panwalker, W. Iskander, A survey of scheduling rules, Oper. Res. 25 (1988).

[18] A. Bhongade, P. Khodke, Heuristics for production scheduling problem with machining and assembly operations, Int. J. Ind. Eng. Comput. 3 (2012) 185–198, http://dx.doi.org/10.5267/j.ijiec.2011.09.003.

[19] L. Asadzadeh, A local search genetic algorithm for the job shop scheduling problem with intelligent agents, Comput. Ind. Eng. 85 (2015) 376–383, http://dx.doi.org/10.1016/j.cie.2015.04.006.

[20] M. Kurdi, An effective new island model genetic algorithm for job shop scheduling problem, Comput. Oper. Res. 67 (2016) 132–142, http://dx.doi.org/10.1016/j.cor.2015.10.005.

[21] M. Aydin, T. Fogarty, A distributed evolutionary simulated annealing algorithm for combinatorial optimisation problems: Special issue: New advances on parallel meta-heuristics for complex problems (Guest editor: Enrique Alba), J. Heuristics 10 (2004) http://dx.doi.org/10.1023/B:HEUR.0000026896.44360.f9.

[22] X. Song, Q. Men, Y. Cao, Improved taboo search algorithm for job shop scheduling problems, Syst. Eng. Electron. 30 (2008).

[23] Q. Luo, Q. Deng, G. Gong, L. Zhang, W. Han, K. Li, An efficient memetic algorithm for distributed flexible job shop scheduling problem with transfers, Expert Syst. Appl. 160 (2020) 113721, http://dx.doi.org/10.1016/j.eswa.2020.113721.

[24] W. Aalst, Petri net based scheduling, Oper. Res. Spektrum 18 (1996) 219–229, http://dx.doi.org/10.1007/BF01540160.

[25] S. Peng, T. Li, J. Zhao, Y. Guo, S. Lv, G.Z. Tan, H. Zhang, Petri net-based scheduling strategy and energy modeling for the cylinder block remanufacturing under uncertainty, Robot. Comput.-Integr. Manuf. 58 (2019) 208–219, http://dx.doi.org/10.1016/j.rcim.2019.03.004, URL https://www.sciencedirect.com/science/article/pii/S0736584518303806.

[26] Y. Cheng, K. Chen, H. Sun, Y. Zhang, F. Tao, Data and knowledge mining with big data towards smart production, J. Ind. Inf. Integr. 9 (2018) 1–13, http://dx.doi.org/10.1016/j.jii.2017.08.001, URL https://www.sciencedirect.com/science/article/pii/S2452414X17300584.

[27] J. Chen, T. Zhu, O. Bälter, J. Xu, W. Zou, A. Hedman, R. Chen, M. Sang, Fishbuddy: Promoting student engagement in self-paced learning through wearable sensing, 2017, pp. 1–9, http://dx.doi.org/10.1109/SMARTCOMP.2017.7947008.

[28] O. Romanov, M. Nesterenko, N. Fesokha, V. Mankivskyi, Evaluation of productivity virtualization technologies of switching equipment telecommunications networks, Inf. Telecommun. Sci. (2020) 53–58, http://dx.doi.org/10.20535/2411-2976.12020.53-58.

[29] T.M. Shortell, INCOSE Systems Engineering Handbook: a Guide for System Life Cycle Processes and Activities, John Wiley & Sons Hoboken, NJ, 2015.

[30] E. Trunzer, A. Wullenweber, B. Vogel-Heuser, Graphical modeling notation for data collection and analysis architectures in cyber-physical systems of systems, J. Ind. Inf. Integr. 19 (2020) 100155, http://dx.doi.org/10.1016/j.jii.2020.100155.

[31] T. Weilkiens, CHAPTER 4 - SysML—The systems modeling language, in: T. Weilkiens (Ed.), Systems Engineering With SysML/UML, in: The MK/OMG Press, Morgan Kaufmann, Burlington, 2007, pp. 223–270, http://dx.doi.org/10.1016/B978-0-12-374274-2.00004-3.

[32] T. Weilkiens, CHAPTER 3 - UML—Unified modeling language, in: T. Weilkiens (Ed.), Systems Engineering With SysML/UML, in: The MK/OMG Press, Morgan Kaufmann, Burlington, 2007, pp. 143–221, http://dx.doi.org/10.1016/B978-0-12-374274-2.00003-1.

[33] S.C. Spangelo, D. Kaslow, C. Delp, B. Cole, L. Anderson, E. Fosse, B.S. Gilbert, L. Hartman, T. Kahn, J. Cutler, Applying model based systems engineering (MBSE) to a standard CubeSat, in: 2012 IEEE Aerospace Conference, 2012, pp. 1–20, http://dx.doi.org/10.1109/AERO.2012.6187339.

[34] E. Brusa, Synopsis of the MBSE, lean and smart manufacturing in the product and process design for an assessment of the strategy "industry 4.0", in: E. Mancin, A. Garro, L. Tirone, D. Fierro, P. Gaudenzi, A. Falcone (Eds.), Proceedings of the 4th INCOSE Italia Conference on Systems Engineering, Rome, Italy, November 28-30, 2018, in: CEUR Workshop Proceedings, vol. 2248, CEUR-WS.org, 2018, pp. 21–30, URL http://ceur-ws.org/Vol-2248/paper3.pdf.

[35] F. Psarommatis, A generic methodology and a digital twin for zero defect manufacturing (ZDM) performance mapping towards design for ZDM, J. Manuf. Syst. 59 (2021) 507–521, http://dx.doi.org/10.1016/j.jmsy.2021.03.021.

[36] C. Wang, MBSE-Compliant product lifecycle model management, in: 2019 14th Annual Conference System of Systems Engineering, SoSE, 2019, pp. 248–253, http://dx.doi.org/10.1109/SYSOSE.2019.8753869.

[37] F. Jouault, F. Allilaire, J. Bézivin, I. Kurtev, ATL: A model transformation tool, Sci. Comput. Program. 72 (1) (2008) 31–39, http://dx.doi.org/10.1016/j.scico.2007.08.002, URL https://www.sciencedirect.com/science/article/pii/S0167642308000439, Special Issue on Second issue of experimental software and toolkits (EST).

[38] O. Batarseh, L. McGinnis, J. Lorenz, 6.5.2 MBSE supports manufacturing system design, INCOSE Int. Symp. 22 (1) (2012) 850–860, http://dx.doi.org/10.1002/j.2334-5837.2012.tb01375.x.

[39] E. Seligman, T. Schubert, M.V.A.K. Kumar, Chapter 2 - basic formal verification algorithms, in: E. Seligman, T. Schubert, M.V.A.K. Kumar (Eds.), Formal Verification, Morgan Kaufmann, Boston, 2015, pp. 23–47, http://dx.doi.org/10.1016/B978-0-12-800727-3.00002-2.

[40] S. Herzig, A. Qamar, A. Reichwein, C. Paredis, A conceptual framework for consistency management in model-based systems engineering, in: Proceedings of the ASME Design Engineering Technical Conference, Vol. 2, 2011, pp. 1329–1339, http://dx.doi.org/10.1115/DETC2011-47924.

[41] D. Kolovos, R. Paige, F. Polack, Eclipse development tools for epsilon, 2021, URL https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.96.2464&rep=rep1&type=pdf.

[42] M. Gogolla, F. Büttner, M. Richters, USE: A UML-based specification environment for validating UML and OCL, Sci. Comput. Program. 69 (1) (2007) 27–34, http://dx.doi.org/10.1016/j.scico.2007.01.013, Special issue on Experimental Software and Toolkits.

[43] J. Cabot, M. Gogolla, Object constraint language (OCL): A definitive guide, in: M. Bernardo, V. Cortellessa, A. Pierantonio (Eds.), Formal Methods for Model-Driven Engineering: 12th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2012, Bertinoro, Italy, June 18-23, 2012, in: Advanced Lectures, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 58–90, http://dx.doi.org/10.1007/978-3-642-30982-3_3, URL DOI: 10.1007/978-3-642-30982-3_3.

[44] A.L. Fraga, M. Vegetti, H.P. Leone, Ontology-based solutions for interoperability among product lifecycle management systems: A systematic literature review, J. Ind. Inf. Integr. 20 (2020) 100176, http://dx.doi.org/10.1016/j.jii.2020.100176.

[45] S. Feldmann, K. Kernschmidt, B. Vogel-Heuser, Combining a SysML-based modeling approach and semantic technologies for analyzing change influences in manufacturing plant models, Proc. CIRP 17 (2014) 451–456, http://dx.doi.org/10.1016/j.procir.2014.01.140, Variety Management in Manufacturing.

[46] H. Kern, A. Hummel, S. Kühne, Towards a comparative analysis of meta-metamodels, in: Proceedings Of the Compilation of the Co-Located Workshops on DSM'11, TMC'11, AGERE! 2011, AOOPES'11, NEAT'11, & VMIL'11, in: SPLASH '11 Workshops, Association for Computing Machinery, New York, NY, USA, 2011, pp. 7–12, http://dx.doi.org/10.1145/2095050.2095053.

[47] J. Lu, G. Wang, J. Ma, D. Kiritsis, H. Zhang, M. Törngren, General modeling language to support model-based systems engineering formalisms (Part 1), INCOSE Int. Symp. 30 (1) (2020) 323–338, http://dx.doi.org/10.1002/j.2334-5837.2020.00725.x.

[48] H. Wang, G. Wang, J. Lu, C. Ma, Ontology supporting model-based systems engineering based on a goprr approach, in: Worldcist'19 - 7th World Conference on Information Systems and Technologies, Springer International Publishing, Cham, 2019, pp. 426–436, http://dx.doi.org/10.1007/978-3-030-16181-1_40.

[49] L. de Moura, N. Bjørner, Satisfiability modulo theories: An appetizer, 2009, pp. 23–36, http://dx.doi.org/10.1007/978-3-642-10452-7_3.

[50] L. de Moura, N. Bjørner, Satisfiability modulo theories: Introduction and applications, Commun. ACM 54 (2011) 69–77, http://dx.doi.org/10.1145/1995376.1995394.

[51] C. Barrett, A. Stump, C. Tinelli, et al., The SMT-lib standard: Version 2.0, in: Proceedings of the 8th International Workshop on Satisfiability Modulo Theories (Edinburgh, England), 13, 2010, p. 14.

[52] M. Kilbridge, L. Wester, The balance delay problem, Manage. Sci. 8 (1961) 69–84, http://dx.doi.org/10.1287/mnsc.8.1.69.

[53] N.M. Rong Huang, Human resource configuration optimization method for the assembly line of packaging production subject to workers with different skills, 2016, URL https://kns.cnki.net/KCMS/detail/detail.aspx?dbname=CMFD201602&filename=1016156953.nh.

[54] Original scheduling scheme, URL https://gitee.com/zkhoneycomb/open-share/tree/master/Papers/MBSE_supporting_production_scheduling_based_on_smt/production%20scheduling%20scheme.

[55] MetaGraph-download, URL http://www.zkhoneycomb.com/productinfo/101503.html.

[56] KARMA language specification, URL https://gitee.com/zkhoneycomb/open-share/tree/master/KARMA_language_specification.

[57] L. de Moura, N. Bjørner, Z3: An efficient SMT solver, in: C.R. Ramakrishnan, J. Rehof (Eds.), Tools and Algorithms for the Construction and Analysis of Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 337–340.

[58] Built models in SysML, URL https://gitee.com/zkhoneycomb/open-share/tree/master/Papers/MBSE_supporting_production_scheduling_based_on_smt/MBSE%20supporting%20production%20scheduling%20based%20on%20smt.

[59] C.B. Mulligan, Quantifier Elimination for Deduction in Econometrics, Working Paper Series, 24601, National Bureau of Economic Research, 2018, http://dx.doi.org/10.3386/w24601, URL http://www.nber.org/papers/w24601.