

# A New Approach of Modeling Supervisory Control for Manufacturing Systems Based on SysML

Xiaoshan Lu, Laurent Piétrac, Eric Niel  
Laboratory Ampère (UMR 5005)  
INSA Lyon, F-69100 Villeurbanne, France  
{xiaoshan.lu, laurent.pietrac, eric.niel}@insa-lyon.fr

**Abstract**—The supervisory control theory is widely used to deal with problems of controller design in the field of discrete event systems. Despite the academic attention over last several decades, there were few application cases in real industrial systems. Some scientific results have shown the difficulties of implementation of Supervisory Control Theory (SCT) and proposed some possible solutions. On the other hand, the difficulty of using the theory for system engineers should also be taken into consideration. This paper presents a modeling approach of SCT based on SysML. Aimed at typical manufacturing systems, a three-level model template is proposed to bridge the gap between SCT and system design. For each template in the modeling library, there is one equivalent model in three levels: the SCT model level, the interface model level and SCT model level. The construction of modeling template is based on the prototype abstraction of typical manufacturing system elements. The transformation algorithms between each model level are given in the paper. The engineer can design the system by the standardized templates and modeling procedures in system engineering level. The correspondent SCT model will be created according to the transformation algorithm and computed the solution automatically. An example will be posed to validate the modeling methodology application at the end of the paper.

**Keywords**—discrete event system; supervisor control theory; SysML, modeling; manufacturing system

## I. INTRODUCTION

In the field of Discrete Event System (DES), SCT was first introduced by Ramadge and Wonham in 1982 [1]. By the scientific achievements within the past several decades, the framework of SCT forms a systematic formal paradigm to synthesize controllers for plants. Many solutions have been proposed to deal with SCT problems such as supervisor synthesis [2], [3], controllability [4], observability [5], and supremal sublanguage [6], [7]. On the other hand, the number of states and transitions grow sharply with the scale increasing of system [8]. In order to deal with de state-space explosion problem, different control structures were proposed by research works. Wonham and Ramage introduced the concept of modular control [9] and Queiroz and Cury extends the approach to local modular control [10]. Some new control strategies have been put forward during the past several years such as the coordinated distributed supervision and aggregated distributed supervision proposed by Su [11], [12], by which an coordinator is designated to ensure nonblocking among local supervisors.

However, despite the academic achievements of SCT over last several decades, there were few application cases in real industrial manufacturing systems. There exist several obstacles between the theory and application. Some researchers endeavor to deal with the difficulty of Programmable Logic Controller (PLC) implementation of SCT. Scientific result has shown that the main problems in supervisory control implementation are causality, avalanche effect, simultaneity, choice, etc. [13]. There are some possible solutions to deal with the problems: Queiroz and Cury proposed a control structural control implementation of modular control [14] and examples are posed to illustrate the approach [15]. In [16] DECON9 methodology was proposed to provide a standardized approach and solution to many problems that arise while implementing SCT into PLC.

Another problem for application of SCT is the modeling of real system. The SCT provides us a theoretical basis for behavior analysis of components in the form of formal language or automaton. The SCT has its limitation in modeling large-scale structured discrete event systems. The automaton just represents the behavior aspect of the system and the relationship between the components, the interfaces or interaction can hardly be recognized. To model a discrete event system by SCT, system designers should focus on abstract the system into the key elements in SCT such as state, event, alphabet, transition or automata composition etc. The abilities of system engineers to utilize SCT for modeling system can be more or less limited.

The model-based system modeling language gives us the opportunity to deal with the second problem above, by which designers can pay more attention to the system attributes themselves. Some researchers contributed the object-oriented model. Puroo and Vaishnavi [17] proposed an ordered set with three elements (E, A, M) to represent an object-oriented model. Chidamber and Kemerer define a formal object-oriented model describing the system [18]. Huang extend the modeling method based on SysML and proposed to use SysML to model a system to be simulated and to support the automatic generation of simulation models [19]. All the methodologies provide the possibility to propose a new standardized model template.

The template itself is not enough to deal with the modeling problem. Some questions could be posed: If the automata model is correctly abstracted from the real system or if there is any standardized model template can be used to avoid

modeling mistakes. In this paper, a new rapid modeling approach in SysML, a Model Based System Engineering (MBSE) modeling language, will be proposed for template design of controller. The new modeling approach is aimed to model a DES and realize an automatic supervisor synthesis by standard model templates. A three-level model is put forward to achieve the transformation SysML model to SCT model based on standardized templates. With the proposed modeling method, system designers can focus on the system attributes and do not need to abstract the system in traditional SCT ways. This paper will introduce the modeling method by three sections: the proposed model, the model prototype and the modeling procedure. An example will be posed to validate the modeling methodology at the end.

## II. PRELIMINARIES

### A. Supervisory Control Theory

In SCT framework, we regard the uncontrolled systems as plant and synthesize a supervisor to ensure the control actions of the plant according to the specifications. The supervisor makes sure the controlled system behaves within minimal limitations.

The behavior of a plant can be denoted as an automaton  $G$  such that  $G = (X, \Sigma, \delta, x_0, X_m)$ , where  $X$  is the finite set of states;  $\Sigma$  is the alphabet;  $\delta: X \times \Sigma \rightarrow X$  is the state transition function;  $x_0 \in X$  is the initial state and  $X_m \subseteq X$  is the set of marked states. The generated language of  $G$   $L(G) = \{s \in E^* : f(x_0, s) \text{ is define}\}$  represents the free behavior of the plant and  $L_m(G) = \{s \in E^* : f(x_0, s) \in X_m\}$  the marked language represents the marked behavior of the plant.

The specification can also be modeled by an automaton  $H$  such that  $H = (Y, \Sigma', \delta', y_0, Y_m)$  where  $Y, y_0 \in Y$  and  $Y_m \subseteq Y$  denote set of states, the initial state and the set of marked states respectively. A supervisor  $S$  is a function from the language generated by  $G$  to the power set of  $\Sigma$ . Assuming that  $\Sigma' = \Sigma$ , the controlled behavior under supervisor  $S$  can be represented by the product of  $G$  and  $H$ , if  $G \times E$  is controllable with regard to  $G$  and  $\Sigma_c$ :

$$G \times E = Ac(X \times Y, \Sigma, \tau, (x_0, y_0), X_0 \times Y_0) \quad (1)$$

Where,

$$\tau = \begin{cases} (x', y') & \text{if } (x, \sigma, x') \in \delta \cap (y, \sigma, y') \in \delta' \\ \text{undefined} & \text{otherwise} \end{cases} \quad (2)$$

The  $Ac$  operator means the accessible part of an automaton. If all events are controllable, this operation produces the controlled system behavior respecting to the specification. However, if  $G \times E$  can be verified uncontrollable, an algorithm should be performed to compute the maximal behavior [20].

### B. SysML

SysML is one of the graphical modeling languages for systems engineering applications, first proposed by Object

Management Group (OMG) together with the International Council on Systems Engineering (INCOSE) in 2001 and adopted as a standard in May 2006 (<http://www.omg-sysml.org>). SysML is an extension of the subset of Unified Modeling Language (UML) with nine kinds of diagrams such as block definition diagram (BDD) and activity diagram, by which SysML can represent systems and each components, as well as their behavior and structure.

SysML has been widely put into application for modeling in system engineering. For example, an MBSE Challenge project was established to model a hypothetical FireSat satellite system to evaluate the suitability of SysML for describing space systems [21]. Another example is the requirement modeling of smart surface by SysML [22]. The great advantage of visualization and facility of modeling by SysML gives us the great opportunity to modeling the typical manufacturing system.

## III. THE THREE-LEVEL MODELING APPROACH

The traditional modeling process of supervisory control can be divided into three steps: abstraction, modeling and synthesis. At first step, the behaviors of component and requirements are abstracted into plant and specification respectively. Then, all the elements in five-tuple formalism should be determined and transform the formalism model to automata. By SCT, the supervisor is synthesized in different ways according to the control strategies. Finally, the controllability, observability and nonblocking of the supervisor should be verified and purge the forbidden states and transitions.

The new proposed modeling method is quite different from the traditional ways. The automata abstraction may not be necessary when there are objected-oriented and automatically-transformed model templates. The inputs of the modeling method are key parameters of the system attributes and task requirements and the outputs are the target supervisor design. The nature of the modeling method is the rapid transformation and computation from real system design to SCT solutions.

To achieve rapid modeling SCT in SysML by system engineers, the modeling methodology should meet the following requirements:

- The designer can model the system in the way he is familiar with rather than the formalism which is difficult to understand. Besides, The design can just focus on the component organization, the system requirements and key attributes. The templates library offers enough model prototypes for the designer and models can be customized.
- The SysML model can be transformed to SCT model automatically. According to the system structure and requirements input, the back end SCT model is created. The computation of supervisor synthesis under given specification is performed at the same time.

The three-level model template consists of SysML model, interface formal model and SCT model. The traditional SCT model becomes the bottom model, which are not modeled directly by the system designer. SCT model is encapsulated by two other models: the SysML model templates are front end

directly used by the designer. Between the two levels, the interface model should be constructed to transform the SysML template into SCT model. The schematic of three-level model is shown in Fig. 1. When modeling a system, the user models the system by SysML template and organizes them in diagrams such as Block Definition Diagram (BDD). Then this front-end model will be transformed to the SCT model and computer automatically. The core computation algorithms are based on SCT.

#### a) SysML model level

In SysML level, the SysML model is the correspondent model template representing the system element based on SysML. The model template is an abstract of the element in the system. The SysML model is used to record the inputs of system attributes: the structure, the relation between the components, the parameters and the requirements, etc. In the three-level model, it can be considered as the input module. All the necessary information should be defined in SysML model. The SysML model should be in the form of template in order to meet the requirement of rapid modeling.

The SysML templates are constructed based on the prototypes of manufacturing system in Section IV. The key attributes of a real component are extracted and a SCT model of it can be transformed by these attributes. That means the attributes can be mapped to all the five elements of 5-tuple formalism.

All SysML models should be presented in several SysML diagrams for describing the system structures and requirements. SysML provides us nine kinds of diagrams. Not all the diagrams must be used for modeling, but the diagrams should be interpreted by the three-level model semantically.

#### b) Interface model level

The interface formal model is the core of the three-level model. The function of it is to define the transformation specifications from SysML model to SCT model. The interface model can be formalized by a three-tuple:

$$MI = (\mathcal{A}_s, A_{RW}, \mathcal{F}) \quad (3)$$

Where,  $MI$  is denoted as formal model of each corresponding SysML model;  $\mathcal{A}_s$  is the abstract data type (ADT) which saves all the input information of the element from SysML model;  $A_{RW} = (X, \Sigma, \delta, x_0, X_m)$  is five tuple of the corresponding SCT formalism;  $\mathcal{F}$  is denoted as the set of transformation algorithms from  $\mathcal{A}_s$  to  $A_{RW}$ .

ADT  $\mathcal{A}_s$  consists of two parts: data and operation. The data part stores the input information by data structure. The task of operations is pre-treatment of data  $\mathcal{A}_s$  before transformation. For example, if the input data have default value, there should be operations to fill in the appropriate data.  $\mathcal{F}$  is the function set mapping  $\mathcal{A}_s$  to  $A_{RW}$ .  $\mathcal{F}$  should make sure that all the data in can be correctly transformed to  $A_{RW}$ .

#### c) SCT model level

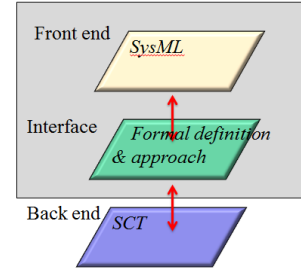


Fig. 1. Schematic of the Three-level model.

SCT model is the output computation core of three-level model. It is encapsulated in the other two models. The SCT model can be obtained directly from because it is the formal model of SCT. In this level, the form of model is an automaton. Based on the R-W theory, the final result of supervisor synthesis can be computed when all the plants and specifications have been built.

The three-level model template is abstracted based on the real manufacturing system components. The SCT abstraction prototype of components is of great importance for the three-level model especially for SysML model and automatic transformation algorithm in interface model. The construction of the three-level model is based on the infrastructure-level SCT prototype and encapsulates it as user-oriented model.

### IV. PROTOTYPE OF TYPICAL MANUFACTURING SYSTEM COMPONENT

The conception of template design for DES was proposed in [23] by using entities and channels. In [24] and [25], the authors parameterize DES template by which the template can be instantiated to different models in application. These papers focused on the algorithms of transforming the parameterized models and supervisor synthesis. The proposition here, however, is based on the modeling prototype of templates of manufacturing system and automatic transformation in three-level model.

The work started from the existing case study of manufacturing system. According to the case studies of [15], [16], [26], [27], the basic components which can compose a manufacturing system under supervisory control at the lower limit are machines, buffers, transportation system and supervisors. Besides, the processing procedures can be considered as specifications and safety requirements. According to SCT, the basic model prototypes must contain the three kinds of prototype: plant, specification and supervisor. In this section, we intend to show the main logic and method of abstracting the typical manufacturing system components.

#### A. Plant

The plant abstracting the real components in manufacturing system can be a machine, a robot, etc. The event sets between plants usually have no intersection. Here, we use the prototype modeling of machine and robot as examples.

##### a) Machine

All the workstation can process a series types of part can be abstracted into machine, for example, a drilling machine, an

assembling station, etc. The SCT prototype can be defined as shown in Fig. 2(a). Different processing can be represented by state  $S_i$  in the automaton. The events of starting processing  $a_i$  and finishing processing  $f_i$  trigger the transitions for each state. According to the paradigm of [14], the starting events are defined as controllable events and the finished events are defined as uncontrollable.

Besides, a machine can also be extended to a machine with failure treatment, shown in Fig. 2(b). The prototype can simulate the behaviors of a machine to which happens an unexpected failure and wait for maintenance. Compared with the aforementioned machine, an additional state and its transitions represent the state of failure and the repair.

#### b) Transportation system (robot)

The Transportation system, usually in form of robot, is to take a part from a buffer and put it to another. Fig. 3 shows the SCT prototype of a robot. The number of buffer decides the state number of automaton. The event sets  $A_i, i=1,2\dots n$  represent move-to-position actions and event  $b_i, i=1,2\dots n$  for each state represent the arrive-at-position actions.

### B. Specification

The specifications represent the behavior constraints of the components in manufacturing system. The events of specification event set usually belong to one or several plants. Here, the prototype modeling of buffer and processing procedure are used as examples.

#### a) Buffer

Buffer can temporarily store several parts between workstations. Buffer is usually regarded as specification in SCT. The prototype of stock buffer, as example, is defined as follows:

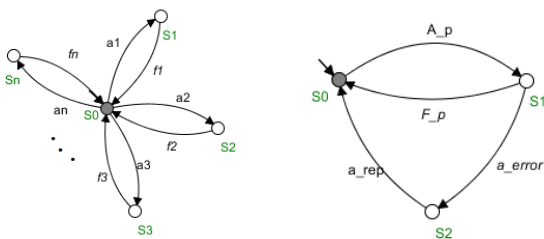


Fig. 2. SCT prototype of machine .

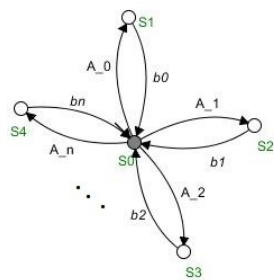


Fig. 3. SCT prototype of robot.

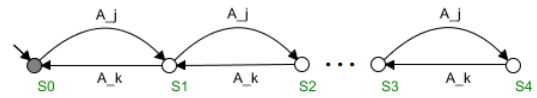


Fig. 4. Example of SCT prototype of buffer.

- The maximum capacity of a buffer is the key parameter to determine the number of state. The state of empty is the initial state and marked state.
- The machines connected to the buffer determine the events and transitions. The end events of machines that enter the buffer consist of the inlet event set  $A_j$ ; the start events of machines that exit from buffer consist of the outlet event set  $A_k$ .

#### b) Processing procedure

The processing procedures are specifications to impose the correct processing sequence of parts to be processed. It establishes requirements as follows:

- Coordinate the behaviors of buffer:
  - The processing procedures of different parts should determine the fork behavior specifications of the relative outlet buffers and the join behavior specifications of the relative inlet buffers.
- Restrain the behavior of the robot:

The robot can move freely between the target components. The processing procedures determine the working trajectory of the robot. Two logics should be taken into consideration: (1) The robot should move between buffers according to the processing procedure; (2) The robot should move from one outlet buffer of machine to inlet buffer to another machine.

### C. Supervisor

The key point of supervisor we concern about is the result of different control strategies. The computation algorithm in the framework of SCT is chosen according to the designated controller design and the supervisor is computed based on the plants and specification. Therefore, the parameter which must be contained in the supervisor model is the control strategy.

A supervisor prototype should have two tasks: the control strategy choice and the computation based on the chosen control strategy when the plants and specifications are given. In our library, there are different control strategies: the centralized control, decentralized control and local modular control, etc.

## V. MODEL TEMPLATE DEVELOPING PROCEDURE

The target participants of the modeling methodology can be divided into two main parts: template designer and model user. Model template designers focus on create and extend the models in the Three-level template library; the modeling user's work is to analyze the system structure and requirements and establish the system model by existed or customized templates. The transformation and computation will be automatically performed after the models are created.

Here, the main development approach of the three-model method template and its principle of modeling, transformation and computation are explained in detail. The SysML modeling tool to use is IBM Rational Rhapsody.

a) Step 1

In section IV, we have mentioned the template prototypes in three types and here templates is defined in SysML for each corresponding prototype. According to the three kinds of prototypes, all the typical manufacturing system element models can be generalized from three stereotypes, shown in Fig. 5. It should be pointed out that the stereotype is just a logical classification and key word of the element to be transformed to SCT model and have no additional attributes. Based on the stereotype categories, all the elements can be classified by generalization from one of them. For example, a machine that process different part types can be abstracted as a block named with stereotype “machine”, generalized by stereotype “plant”.

Now that the prototype automata can be abstracted from the real system, all the prototypes can also be parameterized from logic consideration. The key parameters to be abstracted which are able to offer the enough information for automatic instantiation. The parameterized models are then remodeled in the form of SysML with variables.

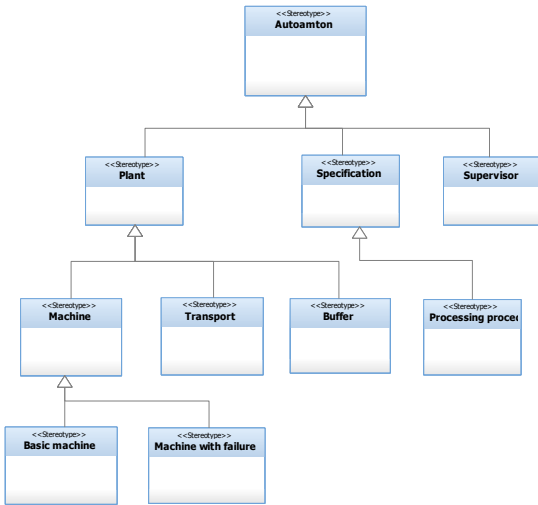


Fig. 5. Stereotype structure.

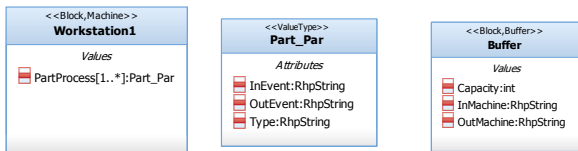


Fig. 6. Definition of machine and buffer in SysML.



Fig. 7. Definition of transport system and supervisor

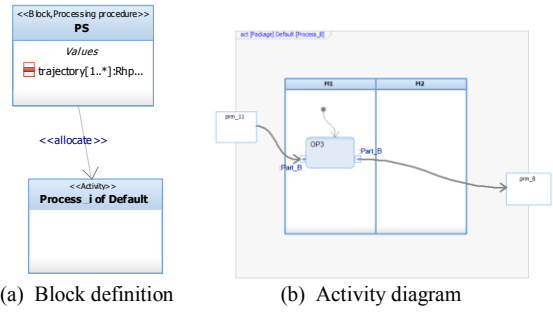


Fig. 8. Definition of processing procedure in SysML.

The attributes of a machine are to input the information of data part: the part name, start events and end events, which are enough for automatic transformation, shown in Fig. 5. The attributes of machine can have default value. A buffer can also be defined as a block with attributes: capacity, Inlet machine and Outlet machine. Fig. 6 shows the logic of the determinacy of the buffer when the three attributes are determined. The block of transport system and supervisor can also be defined, shown in Fig. 7.

It should be pointed out that not only the structure elements but the behavior specifications can be defined by SysML model. Fig. 8 shows an example of SysML model of processing procedure. A block with stereotype “processing procedure” represents the model of processing procedure. Several activity diagrams are allocated to define the processing procedures of different type parts. The parameter “trajectory” defines the exact processing sequence of parts.

b) Step 2

The second step is to create the interface formal model for each block with parameters. In the formal model  $MI$ , the ADT  $\mathcal{S}_s$  save and pre-process the input data and key words from SysML, which are important to transform the SysML model to SCT model. The ADTs of a machine and a buffer are shown in TABLE I.

In Data part of  $\mathcal{S}_s$  of machine, the stereotype of model is stored so as to clarify the SCT model type when transforming. Part\_Par are the key information for transformation. In operation part, the function Default value definition ( $\text{Default}$ ) is to fill in the default values automatically and Type number count ( $\text{Type}$ ) is to pre-process the additional parameter for transformation.

The function set  $\mathcal{F}$  in model  $MI$  is the method set which is to transform  $\mathcal{S}_s$  to SCT formalism  $A_{RW}$ . The nature of the transformation is mapping the parameters given by  $\mathcal{S}_s$  to parameter in  $A_{RW}$  denoted by  $\mathcal{F} : \mathcal{S}_s \rightarrow A_{RW}$ .

$\mathcal{F}$  of machine can be defined as follows:  
Automaton Type = plant;  
 $X = \{0, 1, 2, \dots, n\}$ ,  $n$  is the number of part type;  
 $\Sigma = \{\text{Set of InEvent}\} \cup \{\text{Set of OutEvent}\}$ ;  
 $\delta = \{(0, p, n) \mid p = \text{Part\_Par}[n].\text{InEvent}, n=1, 2, \dots, n\}$   
 $\cup \{(n, q, 0) \mid q = \text{Part\_Par}[n].\text{OutEvent}, n=1, 2, \dots, n\}$ ;  
 $x_0 = \{0\}$ ,  $X_m = \{0\}$ .

TABLE I. THE ADT  $\mathcal{A}_s$  OF MACHINE AND BUFFER

Machine	Buffer
<b>Data:</b> Name of model; Stereotype; Data of Part_Par. <b>Operations:</b> Default value definition (); Type number count ();	<b>Data:</b> Name of model; Stereotype; Capacity; InMachine; OutMachine. <b>Operations:</b>

The Automaton Type assignment should be retrospect to its father stereotype. The developing of a buffer is similar to the machine. The ADT  $\mathcal{A}_s$  of a buffer is shown in TABLE I. There is no operation in buffer because data part is enough for the transformation. The  $\mathcal{S}$  of a buffer can be defined as follows:

$$\begin{aligned}
& \text{Automaton Type} = \textit{specification}; \\
& X = \{0, 1, 2, \dots, n\}, n \text{ is the capacity}; \\
& \Sigma = \{\text{Set of InMachine.OutEvent}\} \\
& \quad \cup \{\text{Set of OutMachine.InEvent}\}; \\
& \delta = \{(n, p, n+1) \mid p = \text{InMachine.OutEvent}, n = 0, 1, \dots, n-1\} \\
& \quad \cup \{(n+1, q, n) \mid q = \text{OutMachine.InEvent}, n = 0, 1, \dots, n-1\}; \\
& x_0 = \{0\}, X_m = \{0\}.
\end{aligned}$$

The mapping should make sure that  $\mathcal{A}_s$  can be interpreted into  $A_{RW}$  correctly whatever the instance is, or the model should be rebuilt. The model must have the characteristic of universality.

In the same way, the  $\mathcal{S}$  of a robot can be defined as follows:

$$\begin{aligned}
& \text{Automaton Type} = \textit{plant}; \\
& X = \{0, 1, 2, 3, \dots, n\}, n \text{ is the number of buffer it move to}; \\
& \Sigma = \{\text{Set of MoveToEvent}\} \\
& \quad \cup \{\text{Set of EndMoveEvent}\}; \\
& \delta = \{(0, pin, n) \mid pin \in \text{MoveToEvent}, n = 1, 2, \dots, n, \\
& \quad i = 1, 2, \dots, i-1, i+1, \dots, n\} \\
& \quad \cup \{(n, qn, 0) \mid qn \in \text{EndMoveEvent}, n = 1, 2, \dots, n\}; \\
& x_0 = \{0\}, X_m = \{0\}.
\end{aligned}$$

The developing of processing procedure is a little more complex that machine and buffer. The activity diagrams are the specific requirements of processing procedure and block ‘‘PS’’ is to gather all the necessary information in activity diagrams and process them to specifications. Based on the block PS, two kinds of specification should be transformed: coordinating specification of buffer behavior and transportation system behavior constraint.

#### (1) Coordinating specification of buffer behavior

The modeling of basic buffer behavior is based on the prototype of buffer shown in Fig. 3. However, the automaton cannot tell the state of buffer when different part type is stored in it. The block PS can tell the fork and join buffer in activity

diagrams. If the buffer is a fork buffer or join buffer, it should split the states from basic buffer and form a specification.

#### (2) The transportation system behavior constraint

The specification restrains the behavior of the robot. The state number of the transformed automata is the number of buffer. There are two kinds of transition: the transition according to the processing procedure; and the transition from in-buffer to out-buffer. All the states are marked states. The PS block pre-processes the procedure information in activity diagram such as the trajectory of the robot and the decision of InMachineBuffer or InMachineBuffer. The  $\mathcal{S}$  of the constrain of transport is:

$$\begin{aligned}
& \text{Automaton Type} = \textit{specification}; \\
& X = \{0, 1, 2, 3, \dots, n-1\}, n \text{ is the number of buffer it moves} \\
& \quad \text{to}; \\
& \Sigma = \{\text{Set of active MoveToEvent}\}; \\
& \delta = \{(m, pmn, n) \mid p = \text{MoveToEvent}, m = \text{OutMachineBuffer}, \\
& \quad m = \text{OutMachineBuffer}, n = \text{InMachineBuffer}, \\
& \quad pmn \in \textit{Trajectory}\} \\
& \quad \cup \{(m, pmn, n) \mid q = \text{MoveToEvent}, m = \text{InMachineBuffer}, \\
& \quad n = \text{OutMachineBuffer}\}; \\
& x_0 = \{0\}, X_m = \{0, 1, 2, \dots, n\}. \\
& \textit{Forbidden event} = \{\text{Set of MoveToEvent}\} \\
& \quad \setminus \{\text{Set of active MoveToEvent}\}.
\end{aligned}$$

#### c) Step 3

The  $A_{RW}$  can be considered as the formalism of SCT model. According to the R-W theory, we can transform it to the 5-tuple into automata. When all the  $A_{RW}$  models have been transformed from SysML models, we can perform the computation of supervisor synthesis in a traditional way. The algorithms are based on the R-W supervisory control theory, which are mentioned in section I and section II.

In conclusion, the model template developing process is to create front end SysML templates with attributes based on the real components, to construct the transformation rules for the corresponding interface model and finally to transform the model into automata in SCT.

## VI. CASE STUDY OF THREE-LEVEL MODELING APPLICATION

In order to illustrate three-level modeling method, an example for supervisory control problem is presented. The case study includes the modeling of plant and specifications and the synthesis of supervisors. The example is cited from [14].

### A. Description of the studied system

The system to be studied consists of a transfer line with three industrial workstations M1, M2 and M3 with two buffers of capacity of only one part respectively, as shown in Fig. 9. The system can process two types of part: Type A and Type B. The process plan defines that part Type A shall have an operation sequence by workstation M1, M2 and M3 in the order; Part Type B is processed by M1 and M2 in the order. A

robot can freely move from one buffer to another without specifications. In this example we don't concern about the behavior of inlet and outlet conveyors of the system.

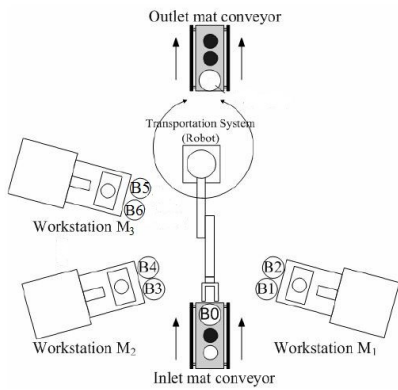


Fig. 9. Schematic of the production line to be studied.

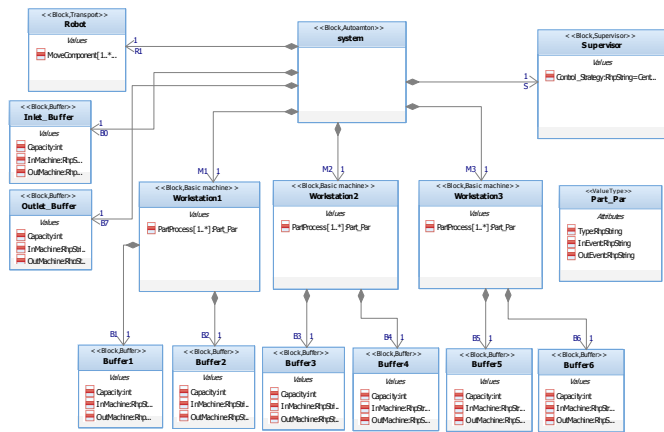
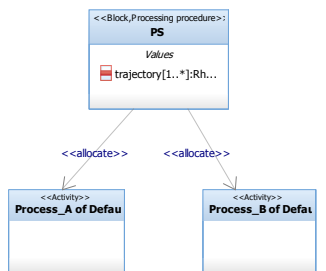
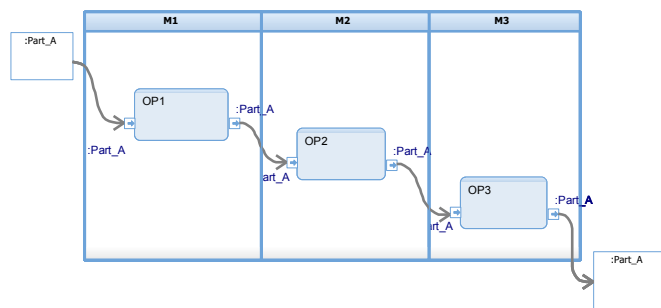


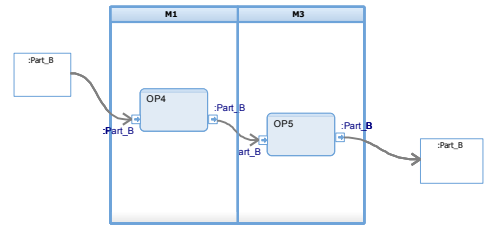
Fig. 10. System Structure.



(a) Definition of Processing procedure



(b) Processing\_A



(c) Processing\_B

Fig. 11. Processing procedure.

### B. Modeling the system by SysML model template

A BDD is used to give the system definition. Each of the components is defined by an individual model template, as shown in Fig. 10. Three machines, eight buffers, a robot and a supervisor represent the real components in the system.

Another BDD and two activity diagrams are used to define the processing procedures of part type A and type B, shown in Fig. 11. The parameters in the model are set according to the real system. If the user do not concern about the name of event, all the event parameters are set default values. The capacity of Buffer is set 1. Control Strategy is set centralized control.

The above modeling process should be done by the model template user, and the following process is performed automatically by the model itself.

### C. Model transformation

The SCT models are obtained by  $A_{RW}$ . According to the algorithms of three-level model, we can transform models to automata representing DES formalism and the result of computation is shown from Fig. 12 to Fig. 16.

The computation of supervisor synthesis can be performed by Supremica. In this case we use centralized control strategy. We set the parameter of supervisor as "centralized control".

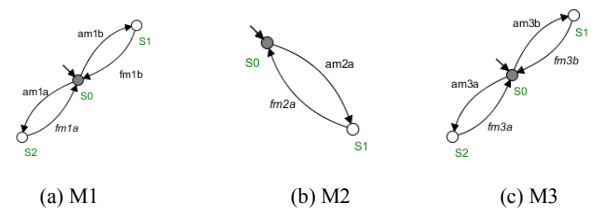


Fig. 12. Automata of Machine.

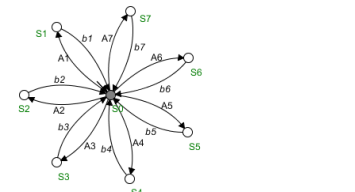
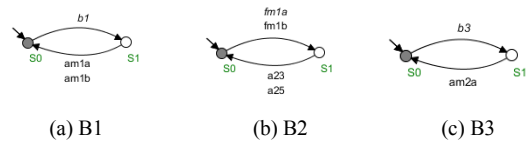


Fig. 13. Automaton of robot.



(a) B1

(b) B2

(c) B3

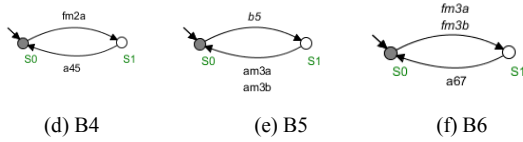


Fig. 14. Automata of buffer.

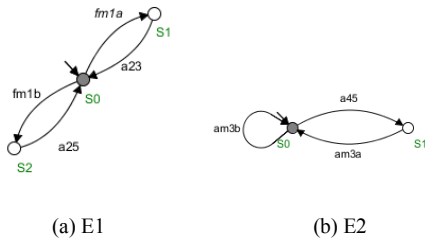


Fig. 15. Automata of processing specification.

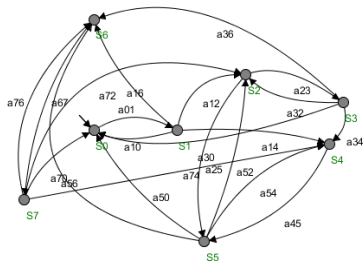


Fig. 16. Automata of robot behavior constraint, E3.

Above is the basic principle of modeling process by three-level modeling technology. It is obvious that the only thing that the user does is making BDDs and activity diagrams which include all the necessary instances of template and setting the parameter according to the real system. In Step C, all the computations of SCT are done by inner computation core based on SCT. Therefore, the user can just concern about the system and achieve a rapid modeling.

## VII. CONCLUSIONS AND FUTER WORK

In this paper, a new modeling approach is proposed. The three-level modeling methodology, based on the encapsulation of SCT model, provides model templates in SysML so that the system designers can achieve the rapid modeling. The system designers use the templates, define the values of them and organize them in SysML diagrams. The models will be automatically transformed and computed by the inner computation core of the model. By this method, the model users can focus on the system attributes and they do not have to take a lot of time in system abstraction. A traditional manufacturing system modeling process is also put forward. We make use of a simple example to illustrate the principle of the rapid modeling process.

In the future work, we will extend the model template library. More templates will be designed so that it can deal with more complex manufacturing system. What's more, the parameters of each template can also be extended. By the generalization of stereotype in SysML, a basic model can be

generalized to different types so that the models have better applicability.

## REFERENCES

- [1] P. J. Ramadge and W. M. Wonham, "Supervision of discrete event processes," in Proc. 21st IEEE Conference on Decision and Control, Orlando, FL, USA, September 1982, pp. 1228-1229.
- [2] F. Lin and W. M. Wonham, "On observability of discrete event systems. Information Sciences," vol. 44, no. 3, pp. 173-198, 1988.
- [3] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," SIAM J. Control and Optimization, vol. 25, no. 1, pp. 206-230, Jan. 1987.
- [4] P. J. Ramadge, and W. M. Wonham, "The control of discrete event systems," Proc. of IEEE, vol. 77, no. 1, pp. 81-98, 1989.
- [5] R. Cieslak, C. Desclaux, A. Fawaz and P. Varaiya, "Supervisory control of discrete-event processes with partial observations," IEEE Transactions on Automatic Control, vol. 33, no. 3, pp. 249-260, 1988.
- [6] R. D. Brandt, V. Garg, R. Kumar, F. Lin, S. I. Marcus, and W. M. Wonham, "Formulas for calculating supremal controllable and normal sublanguages," Systems & Control Letters, vol. 15, no.2, pp. 111-117, 1990.
- [7] S. Lafortune and E. Chen, "The infimal closed controllable superlanguage and its application in supervisory control," IEEE Transactions on Automatic Control, vol. 35, no. 4, pp. 398-405, 1990.
- [8] R. G. Qiu and S. B. Joshi, "A structured adaptive supervisory control methodology for modeling the control of a discrete event manufacturing system," IEEE Transactions on Systems, Man, and Cybernetics-Part A: System and Humans, vol. 29, no. 6, pp. 573-586, Nov. 1999.
- [9] W. M. Wonham and P. J. Ramadge, "Modular supervisory control of discrete event systems," Maths. Control, Signals Syst., vol. 1, no. 1, pp. 13-30, Feb. 1988.
- [10] M. H. de Queiroz and J. E. R. Cury, "Modular supervisory control of composed system," Proc. 19th Amer. Control Conf., pp. 4051-4055, Jun. 2000.
- [11] R. R. Su, J. H. van Schuppen and J. E. Rooda, "Model Abstraction of Nondeterministic Finite-State Automata in Supervisor Synthesis," in IEEE Transactions on Automatic Control, vol. 55, no. 11, pp. 2527-2541, Nov. 2010.
- [12] R. Su, J. H. van Schuppen and J. E. Rooda, "Aggregative Synthesis of Distributed Supervisors Based on Automaton Abstraction," in IEEE Transactions on Automatic Control, vol. 55, no. 7, pp. 1627-1640, July 2010.
- [13] M. Fabian and A. Hellgren, "PLC-based implementation of supervisory control for discrete event systems," in Proc. of the 37th IEEE Conference on Decision and Control, Tampa, FL, 1998, vol.3, pp. 3305-3310.
- [14] M. H. de Queiroz and J. E. R. Cury, "Synthesis and implementation of local modular supervisory control for a manufacturing cell," Proc. 6th Workshop Discrete Event Syst., pp. 377-382, 2002.
- [15] D. B. Silva, A. D. Vieira, E. A. P. Santos and M. A. B. de Paula, "Application of the supervisory control theory to automated systems of multi-product manufacturing," 2007 IEEE Conference on Emerging Technologies and Factory Automation (EFTA 2007), Patras, 2007, pp. 689-696.
- [16] A. B. Leal, D. L. L. da Cruz and M. S. Hounsell, "PLC-based implementation of local modular supervisory control for manufacturing systems," Faieza Abdul Aziz (Ed.), Manufacturing System 1ed., 2012, pp. 159-182.
- [17] S. Puroo and V. Vaishnavi, "Product metrics for object-oriented systems," ACM Computing Surveys, vol. 35, no. 2, pp. 191-221, 2003.
- [18] S. R. Chidamber and C. F. Kemerer, "A metrics for object oriented design," IEEE transactions on software engineering, vol. 20, no. 6, pp.476-493, June 1994.
- [19] E. Huang, R. Ramamurthy and L. F. McGinnis, "System and simulation modeling using SYSML," 2007 Winter Simulation Conference, Washington, DC, 2007, pp. 796-803.



- [20] C. Cassandras and S. Lafortune, "Introduction to Discrete Event Systems," Springer, 2nd edition, 2007.
- [21] S. C. Spangelo et al., "Applying Model Based Systems Engineering (MBSE) to a standard CubeSat," 2012 IEEE Aerospace Conference, Big Sky, MT, 2012, pp. 1-20.
- [22] A. Giorgetti, A. Hammad and B. Tatibouët, "Using SysML for Smart Surface Modeling," 2010 First Workshop on Hardware and Software Implementation and Control of Distributed MEMS, Besan, TBD, France, 2010, pp. 100-107.
- [23] E. A. P. Santos, V. J. D. Negri, and J. E. R. Cury, "A computational model for supporting conceptual design of automatic systems," In Proc. of 13th International Conference on Engineering Design, pp. 517-524, Glasgow, UK, August 2001.
- [24] L. Grigorov, J. E. R. Cury and K. Rudie, "Design of discrete-event systems using templates," in Proc. of 2008 American Control Conference, Seattle, WA, 2008, pp. 499-504.
- [25] L. Grigorov and K. Rudie, "Techniques for the parametrization of discrete-event system templates," In: Proc. 10th Int. Workshop on Discrete Event Systems, WODES '10, Berlin, Germany, 2010, pp. 380-385.
- [26] M. H. de Queiroz, J. E. R. Cury, "Modular Supervisory Control of Large Scale DiscreteEvent Systems," Proc. Int. Workshop Discrete Event Syst. Anal. Control, pp. 103-110, 2000.
- [27] Y. G. Silva. "Formal synthesis, simulation and automatic code generation of supervisory control for a manufacturing cell," Proc. of the 20th International Congress of Mechanical Engineering, Gramado, Brazil, vol. 4, pp.418-426, 2009.