Proceedings of the 14th IFAC Symposium on
Information Control Problems in Manufacturing
Bucharest, Romania, May 23-25, 2012

System Modeling Specification in SysML and Sequence Planner Language - Comparison Study

Sathyamyla Kanthabhabhajeya, Petter Falkman, Bengt Lennartson

Department of Signals and Systems, Chalmers University of Technology, Gothenburg, Sweden (e-mail: satkan@chalmers.se)

Abstract:
This paper compares two languages, Systems Modeling Language(SysML) based on Unified Modeling Language and Sequence Planner Language, where both are used for systems engineering applications. As the modern manufacturing industries pass through a challenging period in storing and exchanging huge amounts of information/data, a common platform that is helpful in putting different parts together is of major interest. This paper presents an analysis of these two languages focusing on their behavioral constructs and details their advantages and disadvantages. This paper concludes by mentioning the points that are lacking in SysML, which could be solved by combining it with Sequence Planner Language and using the combined approach for system engineering applications.

Keywords: SysML, State machine diagram, Modeling, Sequence flow diagram, Manufacturing system

1. INTRODUCTION

The technological development has been increasing exponentially for a few decades and this proportionally increases the complexity of manufacturing systems. Increased complexity, consequently complicates the solution, design, control and construction of the manufacturing system. The progress, or flow of tasks, accomplished by an industry in order to design and develop a manufacturing system is termed as work-flow. The ideal and destined work-flow for a manufacturing system is as shown in Figure 1 where the mechanical design (the left-branch)and software/system development (the right-branch) works in parallel sharing the required information. However, today there is no common platform for data exchange between mechanical and system engineers during the development of a manufacturing system.

There exist several methods like Waterfall model, V-model, Product Lifecycle Management (PLM) and so on to manage the project cycle or work-flow in software development process. In this the V-model holds a better detailed definition of individual areas of a project which could also be used for System Engineering applications, see Forsberg and Mooz (1998). The V-model supports the planning and preparing of the process but does not organize and execute the process itself. Figure 1 exemplifies the need for a common repository for specifying the requirements and a common platform for feasible exchange of information in a work-flow. Since the execution of each task in the defined work-flow use different application tools, the exchange of information and data between tools will be strenuous without a common platform. In the same area of research, languages like Systems Modeling Language (SysML), Automation Markup Language (AML), methodology like Object Process Methodology (OPM) and standard like Standard for the Exchange of Product model data (STEP) could act as a repository and platform for the project are development.

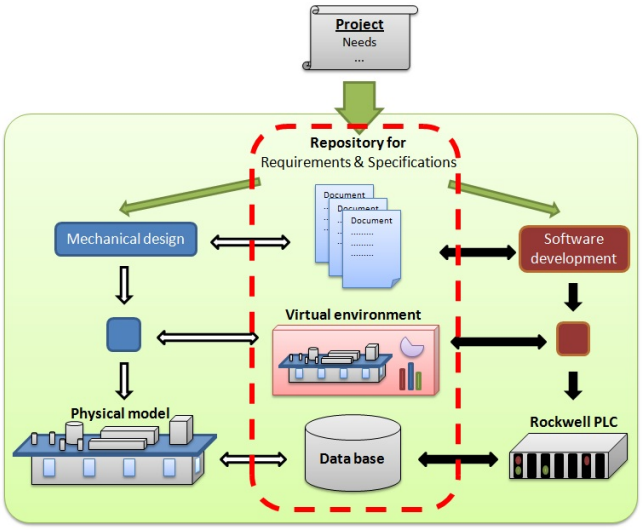


Fig. 1. Work-flow of a Manufacturing Industry

An aim of AML is to realize a common storage of information and exchange of data between engineering domains. Some work related to AML can be found in Schleipen and Drath (2009) describing the three-view-concept and in Drath et al. (2008) explaining the neutral data format of this language, which acts like a glue for seamless automation. The initial objective of STEP standard was to

978-3-902661-98-2/12/$20.00 © 2012 IFAC

10.3182/20120523-3-RO-2023.00189

represent product information which was then extended for communication and storage of process specification, see Falkman et al. (2008). Object Process Methodology (OPM) is another concept or methodology that uses graphical models for describing a system. This methodology uses only one diagram to represent the model of the system. The work related to OPM has spread in different areas e.g. exception handling, see Somekh et al. (2007) and pattern based design and so on. OPM is also compared to Unified Modeling Language (UML) and SysML by Reinhartz-Berger and Dori (2005) and Grobhshtein and Dori (2008) respectively. SysML, the language derived from UML is widely used for system engineering applications. UML has been utilized for software engineering applications, which is extended with new modeling techniques that could be used for system engineering applications in SysML, see Peak et al. (2007). Both UML and SysML are developed with an Object Oriented (OO) approach. SysML aims to combine structural and behavioral constructs of a manufacturing system and shows a better way of interchanging data. XML Metadata Interchange (XMI) format is used by SysML for exchanging data between modeling tools. SysML aims to cover all aspects of system engineering applications, but some requisites can not, however be met, see Bassi et al. (2011).

According to Linhares et al. (2007), SysML is a semi-formal language that intend to support specification, analysis, design and verification of complex systems. Some noteworthy methods for formal verifications are based on formal models of the system designed from automata theory, petri nets, state machines and so on. Linhares et al. (2007) suggests an introduction of Petri nets, for formal system behavior modeling and Temporal Logic, for formal verification, into SysML, to properly verify the design of the manufacturing system before implementation. Also Makartetskiy and Sisto (2011) approach a method to verify SysML requirements using state machine diagrams and other external tools.

In this paper, SysML is compared to Sequence Planner Language (SPL) which is a formal-graphical modeling language developed by the Automation group, Chalmers University, Sweden, see Lennartson et al. (2010). SPL is a new sequence planning approach, which uses self-contained operations to model the activities and execution constraints. SPL is also used to visualize these activities of the manufacturing system in multiple perspectives, see Bengtsson et al. (2011). The languages like Petri nets (PN) and Sequence Function Charts (SFC), use the basic notations of states and events similar to most other formal languages. SPL in more specific, include formal definition of operations and sequences of operations, see Lennartson et al. (2010). The present paper analyzes and evaluates the contributions and limitations of SysML and SPL using two examples. One of the examples is an academic model, which projects the need of resource booking and shows that modeling a resource booking in SysML is laborious compared to modeling in SPL. The second example, a case study of a Filling module in a packaging machine is useful to prove the capability of SysML and SPL models. The formal verification is also considered to be advantageous for SPL, explained in later sections.

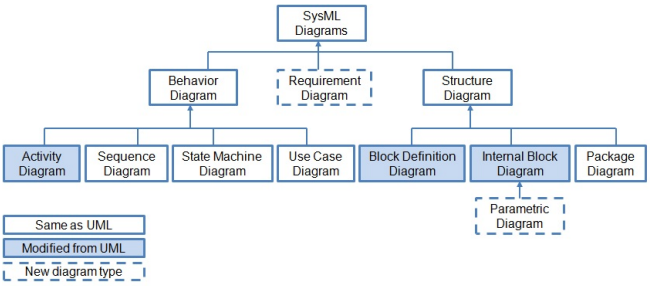


Fig. 2. Taxonomy of SysML diagrams

The following sections, 2 and 3, describe the two languages along with the academic example, followed by a comparison study in Section 4. Section 5, describes the case study, in which a filling machine from the packaging industry Tetra Pak is modeled in both SysML and SPL, followed by a conclusion and future work in Section 6.

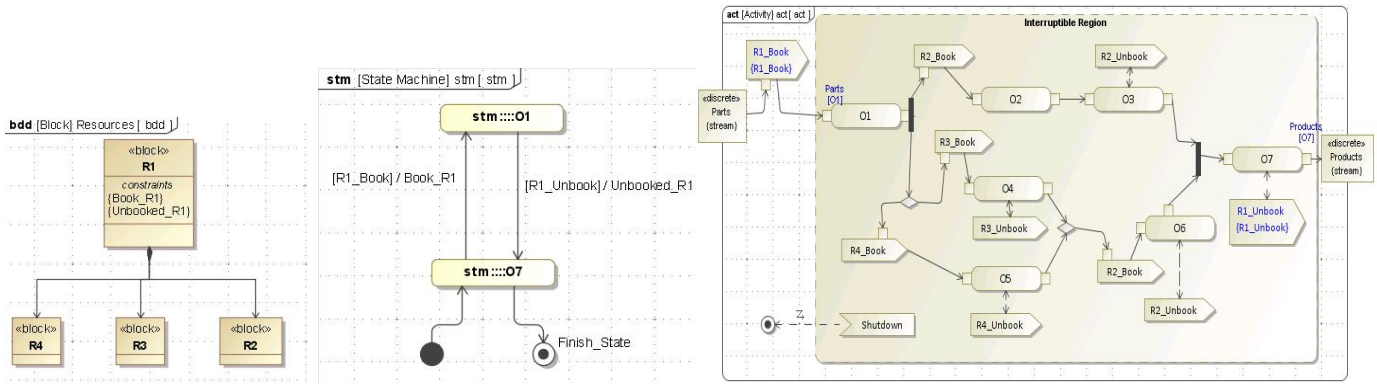
2. SYSML

The nine different diagrams that are classified in SysML language are arranged according to their purpose into three main divisions as *Behavior diagram, Requirement diagram* and *Structure diagram*. The taxonomy of the diagrams in SysML is shown in Figure 2. This section exemplifies different diagrams in SysML using an academic example, which utilizes three diagrams of SysML for its model. The tool used to model this example is MagicDraw 17.0.1 beta 2 from the vendor NoMagic MagicDraw SysML.

Example An academic example using seven operations from O_1 to O_7 is modeled using SysML in this section and using SPL in Section 3. These operations uses four different resources, R_1 to R_4. These resources can be a machine, a robot, a conveyor or a pump. An operation has to book the required resource and un-book the same resource in order to avoid a dead-lock situation. In this academic example, operation O_1 initiates the sequence and also books the resource R_1. Operation O_1 is followed by a set of parallel operations. O_2 is followed in sequence by O_3 and this sequence is in parallel to the operations O_4 or O_5 and O_6. The final operation O_7 is executed once the parallel operations are executed and also un-book the resource R_1. The operations O_2, O_3 and O_6 use a common resource R_2, while operations O_4 and O_5 uses resources R_3 and R_4 respectively. The mutual exclusion for using resource R_2 is important in this situation, as it is used by three different operations.

2.1 Structure

The Structural construct of SysML defines the physical structure of a system that is to be modeled with its four different diagrams; Package diagram, Block definition diagram (BDD), Internal block diagram (IBD) and Parametric diagram. Package diagram initializes and assigns levels of hierarchy between other diagrams and overall observations of the entire model. This diagram is primarily employed for organizing the model by grouping the model elements, see Debbabi et al. (2010). The complete system to be modeled is modularized into blocks and the



(a) Block Definition Diagram (b) State Machine Diagram (c) Activity Diagram

Fig. 3. Academic example as SysML model

relation between them is defined using the other diagram of SysML called BDD. The blocks are modular units of system description, see OMGSpecification (2010). Each block can also categorize parts, unit, value, enumeration, instance, constraint and other properties of the physical module. The BDD helps to confine the association between different blocks and define the multiplicity of inputs. The IBD is a detailed view diagram of a block, which is already defined in BDD. This diagram can include ports and shows the interconnection using connectors between the parts in a block, discussed in Section 5. The ports describe the inflow and outflow of items to the defined block of the system. As a sub-division of IBD another diagram called Parametric diagram, which is used to identify the parameters that are necessary for describing a part of a block with physical parameters is described as well. It can hold equations with variables along with its units wherever required. The BDD of the academic example is represented in Figure 3a, which shows the blocks for the resources $(R_1 - R_4)$ and the association between different resources that are used by the operations.

2.2 Behavior

The dynamic, behavioral constructs of a manufacturing system is modeled using the diagrams in this category, see OMGSpecification (2010). It contains four diagrams, namely

- Use Case diagram
- Sequence diagram
- State Machine diagram
- Activity diagram

In this category, apart from Activity diagram all the other diagrams are re-used from UML while the former is a modified version of UML.

Use Case Diagram: The relationship and the communication between the actor (user) and the system (model) are defined briefly in a Use Case diagram. This diagram uses a communication path to define the association between the actor and the system. This can also, *include, exclude, extend or generalize* the relation between the actor and the system.

Sequence Diagram: The sequence diagram is used in SysML to specify the interactions between entities. This is similar to UML's interaction diagram and one of its four diagram types is Sequence diagram. The interaction, or flow of control, between the actor and the system is characterized by a sequence diagram in SysML. The *lifeline, interactionuse, combined fragments, coregion* and *timeconstraint* are some of the elements of sequence diagram which can be used to clarify the flow of control between an actor and a system in different situations.

Activity diagram: Activity along with State Machine diagram, are dedicated to analyzing and modeling the flow of control within the system. Activity diagram models the inputs, outputs, sequences and conditions for coordinating behaviors. This diagram is flexible and able to assign and link blocks from the structural constructs to the activities of the behavioral constructs of the system. This diagram controls two different flows in a system; *control-flow* and *object-flow*. A control-flow (represented by a dotted line) is initiated with a generation of tokens by an *InitialNode* and this token follows the path until it finds a terminal with a *FlowFinal* node. An object-flow (represented by a solid line) is carrying the object or item that is defined previously in an IBD of the system. The object-flow could be associated with the item-flow in the IBD. The objects or items used in the production line could be categorized according to their type as a *continuous* flowing object or a *discrete* flowing object and/or a *stream* (an uninterrupted flow). The activity diagram also provides a way to repeat the sequence of actions using an *interuptibleRegion*.

As noticed in Figure 3c, the activity diagram of the academic example explains the sequence of operations $(O_1 - O_7)$ with modeling elements like fork, join, decision , merge and actions. The actions could be associated with either the blocks or parts of the block in the structural constructs. This activity diagram uses *sendSignalAction* (like $R1\_book, R1\_Unbook$) to make the resource booking possible between different operations. A signal $\{R1\_Book\}$ will be initiated when the object-flow token passes through and this signal is associated with a state machine diagram as shown in Figure 3b.

State Machine Diagram: State machine diagram in SysML is similar to the standard state diagram describing

the discrete behavior of the system. Transitions between states are specified along with events and guards, while the states have a specification for 'do activity'. Along with these elements, state machine diagram can refine the requirements of the system.

In Figure 3b, the authentic action after a *sendSignalAction* (Figure 3c) is defined with the help of this state diagram. A cross verification of the transition of an initiated signal from the activity diagram is made by checking the current active state of the state machine diagram. This is described in Figure 3c with $Parts[O1]$ before the action O_1 for $R1\_Book$. A two-state model similar to the one shown in Figure 3b has to be designed for each resource that has to be booked and unbooked by the operations in SysML.

2.3 Requirements

The cross-cutting constructs, or requirements define the needs and necessities of the system. This diagram specifies the capability or condition that must be satisfied by the system. This also describes the performance condition of the system and that needs to be used while constructing the model. This diagram in SysML follows a text-based requirement method where it also admits to use graphical, tabular or tree-structure formats, see OMGSpecification (2010). This requirement diagram in SysML can also have links to external file format that are standard for listing the requirements of the project, see Friedenthal et al. (2009). It also has elements like *verify, satisfy, derive, trace, refine* and so on in spite of tracking the requirements during and after modeling.

3. SEQUENCE PLANNER LANGUAGE

The formal graphical model of Sequence Planner Language (SPL) denoted as SOP in Lennartson et al. (2010) is presented in this section. SPL defines the sequence of operations that are necessary for the manufacturing system to either yield or help to yield a product. The main four associates of SPL are *Operations*, *Resources*, *Liaisons* and *Views*, for defining the groups of sequence of operations. SPL has similarities in its graphical structure with GRAFCET and SFC but it has a different semantics with three-state operations, see Lennartson et al. (2010). The same academic example, modeled in SysML is also modeled in SPL using the tool called Sequence Planner shown in Figure 5. This tool provides a bridge between other simulation languages, formal language verification tools and PLC code generation. Magnusson et al. (2011) proposes a method for generating the sequences of operation in SPL directly from a virtual environment.

3.1 Operations

An operation is an activity that takes place in the system at a particular time and under specific conditions. This operation is formally modeled as an Extended Finite Automata (EFA) by Skoldstam et al. (2007), where EFA is a generalization of automata including guards and actions, see Bengtsson et al. (2011).

Definition 1. An extended finite automata is a 7-tuple

$$E = \langle Q \times V, \Sigma, \mathcal{G}, \mathcal{A}, \rightarrow, (q_0, v_0), M \rangle$$

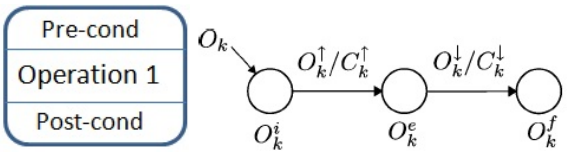


Fig. 4. Operation with pre and post condition in Sequence Planner.

where the set $Q \times V$ is the extended finite set of states, Q is a finite set of *locations* and V is the finite domain of definition of the variables, Σ is a nonempty finite set of events (the alphabet), $\mathcal{G}$ is a set of guard predicates over V, $\mathcal{A}$ is a collection of action functions, $\rightarrow \subseteq Q \times \Sigma \times \mathcal{G} \times \mathcal{A} \times Q$ is a state transition relation, $(q_0, v_0) \in Q \times V$ is the initial state, and $M \subseteq Q \times V$ is a set of marked desired states.

An operation is extended to its equivalent EFA as shown in Figure 4 with initial, executive and final locations (O_k^i, O_k^e, O_k^f) with guards and actions represented during the transition as condition on the current and updated value of the variables $(C_k \leftrightarrow \mathcal{G}/\mathcal{A})$.

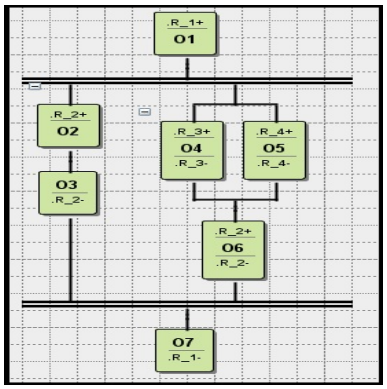
Definition 2. An operation is an EFA where the set of locations $Q_k = \{O_k^i, O_k^e, O_k^f\}$, the event set $\Sigma_k = \{O_k^{\uparrow}, O_k^{\downarrow}\}$, the set of transition conditions $C_k = \{C_k^{\uparrow}, C_k^{\downarrow}\}$, the transition relation $\rightarrow_k = \{\langle O_k^i, O_k^{\uparrow}/C_k^{\uparrow}, O_k^e\rangle, \langle O_k^e, O_k^{\downarrow}/C_k^{\downarrow}, O_k^f\rangle\}$, the initial location $q_k^i = O_k^i$, and all locations are marked, $M = Q$.

Transitions/Relations: Product or process flow is modeled in SPL using a group of operations interlinked with transitions. This transition between operations can be in a sequential way, where completion (O_k^f) of one operation initiates the next. As shown in Figure 4, the relations between consecutive operations could be an alternative, a parallel or a sequence type transition. Each type of transition, including the arbitrary order relation, follows a set of conditions when moving from one operation to another, see Lennartson et al. (2010).

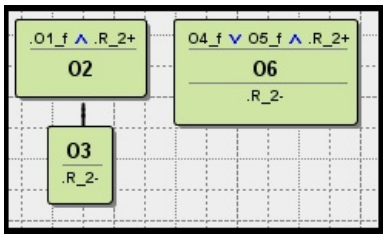
Self-contained Operations: Each operation is capable of defining the complete model without any transitions, referred to as Self-contained operations. This helps in multiple views of sequences discussed later in Section 3.3.

3.2 Resources

After specifying the sequences of operations of a product or process, SPL also allows the allocation of resources to operations. The resources available for a manufacturing system are assigned to operations, which utilize them for their respective activity. The allocation of the same resource to different operations is also feasible in SPL and hence, provides a resource booking facility to model the mutual exclusion. As in the academic example model (Figure 5a), resources $(R_1 - R_4)$ are allocated to operations and each resource is booked with a particular variable (R_1^+) in the operations pre-conditions. The same resource is unbooked (R_1^-) in the post-condition of the operation. In similar fashion, variable allocation could also be defined in SPL. Any number of variables could be created and allocated to operations. Variables also define some properties of the manufacturing system on its own, for example a counter, a



(a) View of Operation sequence



(b) Resource R_2 View

Fig. 5. SPL Model of an Academic example

switch, a timer and so on. The incrementing, decrementing and assigning of values to the variables is developed during the operation flow. Input and output signals are verified using the variables along with the model.

3.3 Liaisons and Views

This associate Liaisons act as a driver for the operation with its products. It only has two locations initial and final. This defines the entity of a process flow. Liaisons are the interface between two products is defined. For example, the interface or intersection points of two products are defined using liaisons, which are later used for modeling a welding or drilling operation, in other words liaisons are to be realized by operations. The other special technique in SPL is Visualization and this is useful for organizing the operations, resources, products or processes according to the interest of the user. As shown in Figure 5b, operations related to the resource R_2 are projected, and the other related operations are added as pre- and post- conditions to these particular operations. Relation Visualization is explained in detail by Bengtsson et al. (2011).

4. COMPARISON STUDY

SysML is a second step of UML, developed mainly for system engineering applications and it concentrates both on structural constructs and behavioral constructs of a manufacturing system. SPL in turn was developed to acheive a user-friendly formal modeling and verification technique that focuses mainly on behavioral constructs of the system.

4.1 Resource Allocation

SysML allows the user to allocate an *action* defined under a behavioral activity to a corresponding *block* defined in a structural construct of the same system, see Section 5. But, in order to utilize a common resource in a manufacturing system, it has to be booked while required and unbooked after. As can be noted in the academic example discussed in Section 2, the booking and unbooking of each resource in the model requires three different SysML diagrams and many signals, events, guards and states(see Figure 3 and 3c). Whereas in SPL, it is possible to define resources in the manufacturing system and allocate them to particular *operations*. As discussed in Section 3 with the same academic model, the resources are booked and unbooked in the pre- and post- conditions of the same operations respectively. If two or three operations uses the same resource and happens to appear at the same time, the first operation that books the resource utilizes it and the other two wait for the former to unbook the resource. This method is also used for mutual zone booking where two or more machineries(/robots) have to work in the same area.

In this stream of allocations, the *object flow* and *item flow* of behavior and structural diagrams respectively of SysML, are allocated to/from one another. This object flow allocation has made it possible to follow any instance of a product or an object that passes the system. It is necessary to track the product instance when two instances of the same product undergo different sets of operations. Following a particular instance is necessary to reduce the exception handling time. In SPL, the different instances of one product are flowing along with the sequence flow of operations in the model. It is possible to track the particular instance by defining additional conditions to the model.

4.2 Visualization

The method of *Visualization* in SPL is of greater importance, because viewing the modeled system from different perspectives make the modeling task simpler and also makes it easier to analyze the entire manufacturing system. The manufacturing system has a set of sequences of operations which describes the process flow, while another set of sequences describing the product flow with the same set of operations arranged in a different way. The projection to view the model according to the process flow or product flow or sequence of operations executed by a single resource (machine) is obtained automatically in SPL using the tool Sequence Planner (Figure 5). These graphical representations of different operation views, familiarizes the designer with the complete manufacturing system. In SysML, the projection to get and visualize the relation between actions and other blocks is defined using *Views* and *Viewpoints*. Particular blocks or actions are manually assigned to a *Viewpoint* and later the projections of these actions or blocks is obtained manually as its *Views*.

4.3 Object Flow

An important advantage of SysML is its capability of defining the objects or items as *continuous*. This stereotype helps in modeling the continuously flowing objects like fluid. The inter-dependability of the system with the object can also be analyzed during the modeling stage of the system itself. This generation of continuous products or objects is not possible with SPL. However, SPL has the

advantage of breaking down an operation into a 3-location EFA. These three locations, as explained in Section 3.1, are initial, executing and final locations of the operation. These different locations of a single operation can be used as a guard for other operations and for formal verification, see Section 4.4. For example, if an operation (O_k) has to start when another operation (O_ℓ) is running, then O_k holds a pre-condition stating O_ℓ^e which shows O_ℓ has to be executing to initialize O_k. This three-state operation is not possible to be defined in SysML's actions. Note, however, that SysML, in its state machine also has a defined three-state behavior for each state as *entry, do and exit*, which acts as a label and include only a textual expression.

4.4 Formal Verification

Comparing the methodologies of SysML and SPL, clarifies the representation of the specification of the model in each respective language. The methodology for modeling a system in SPL begins by obtaining the information from a simulation tool or a design tool, continues by forming the sequences of operations of the process, see Magnusson et al. (2011). The specification, or requirement is also defined as a model in the Sequence Planner tool, either by defining a resource and its particulars, or by defining the specification of other necessities. After modifications and defining constraints to the sequences, the automata model (Figure 4) of this original system is formally verified against its specification. This verified model is later optimized, if required. Optimization aids in designing the process path, identifying the fastest path of the system and so on.

For SysML, one of the methodologies for modeling is suggested by Bassi et al. (2011), in which, modeling in SysML begins with a requirement diagram where the specification or requirements for the system are scripted in text. Followed by classifying the diagrams using Package diagrams, then modeling the structural constructs and behavioral constructs, followed by allocating the blocks to actions. The main disadvantage of SysML is the fact that the specifications are not defined in a formal language but rather as a text. As formal verification requires a formal specification to precisely point out, for instance deadlock situations and for the purpose of further analysis, the SysML specification has to be re-modeled in some formal language and used for verification, see Section 1. To add a note, the tool Sequence Planner is still in its development stage, as all the properties of SPL are not yet completely available.

5. CASE STUDY

To view the possibilities of modeling a manufacturing system using SysML and SPL, a filling module of a packing machine (Tetra Rex TR/28) of Tetra Pak is used. Both the structure and behavior of this machine is modeled in SysML, and the sequence of operations (behavior) is modeled in SPL.

Tetra Rex, TR/28 is an extended hygiene filling machine used for filling chilled products. The lower and upper parts of the filling module is shown in Figure 6, and is an example of a complex manufacturing system. The TR/28 filling

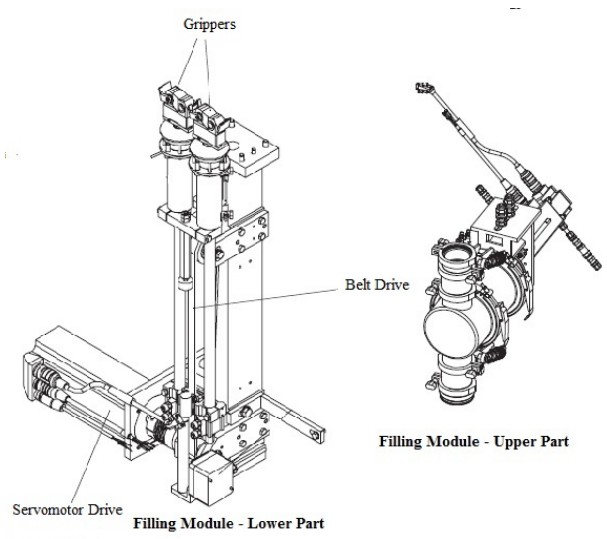


Fig. 6. Filling Module of Tetra Rex TR/28 machine, from TetraPakDocument (2007).

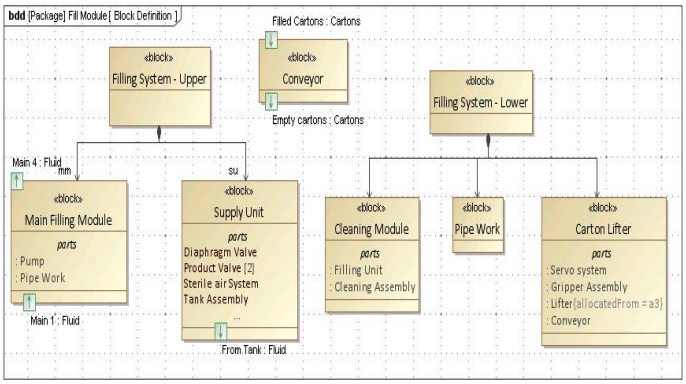


Fig. 7. BDD of Filling Module of TR/28.

machine is divided into five modules and this section will concentrate on modeling the *Filling Module*. The function of this module is divided into two sections, a lower and a upper section, where in the lower half, cartons move to the Lifter, are lifted up, and finally filled with fluid while moving down in the lift. While the upper half includes piston and valves, that initially perform the charge stroke followed by the fill stroke that fills the carton with the fluid. As this paper focuses on the study of two different languages and the methodology that could be used in future for prominent work-flow, some of the detailed analysis like modeling the complete machine, detailed model of structural constructs are not in focus here. The distinctive feature of Tetra Pak's filling and packing machines are that the entire process of the machine follows a certain *motion control profile* and performs a cyclic behavior.

5.1 SysML Model

In SysML, the filling module of TR/28 is drafted as a package using a BDD, shown in Figure 7. The filling module is structurally divided into upper, lower and conveyor portions of the system. The upper and lower portions are indeed listed with different blocks along with their own association and parts. Two more BDDs are created to describe in detail the resources available in the Upper and

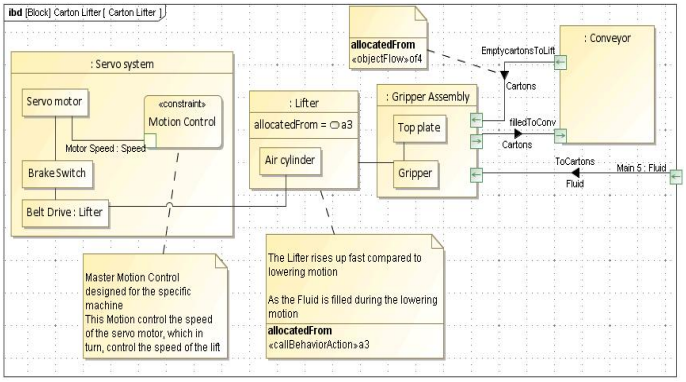


Fig. 8. IBD of Carton Lifter of TR/28.

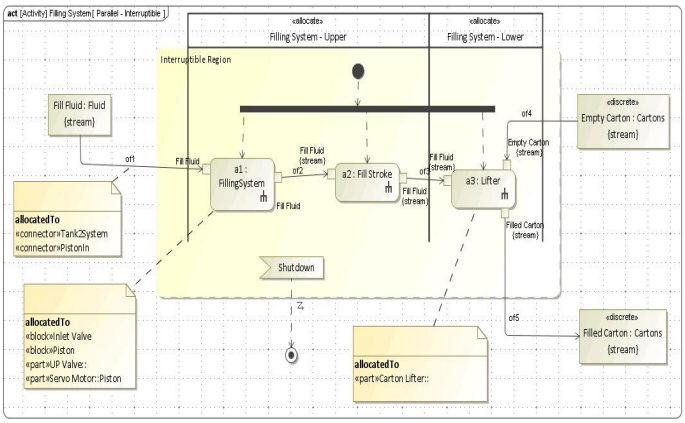


Fig. 9. Activity diagram of Filling Module of TR/28.

Lower sections and where the Objects or Items (Cartons and Fluid) that are being used in this manufacturing system are also defined. The IBD of the block *Main Filling Module* and *Carton Lifter* are to be modeled and Figure 8 depicts the IBD of *Carton Lifter*. This diagram discloses the connection and availability of ports in each blocks. These also admit the kind of flow and its direction between the blocks. Figure 8 shows in detail the in-flow and out-flow of both cartons and fluid to be filled. This diagram also displays that the *Lifter* block is *allocatedFrom* the action a_3, where the action a_3 is defined in an activity diagram, as shown in Figure 9.

Similarly, the in-flow of cartons from the conveyor to the gripper assembly is related to the object-flow of_4 mentioned in the activity diagram. With the help of the object-flow relation, it is possible to track the particular instance of the material in-flow. The other important factor in Tetra Pak machines, as mentioned earlier, is the *Motion Control Profile*. This profile controls the speed of the motor, which in turn controls the movement and flow of material in the system. The complete system is synchronized to this motion control. In order to explicitly include this profile, a *Motion Control* $\langle\langle constraint \rangle\rangle$ block is introduced along with a *Servo motor* block, as shown in Figure 8. This constraint block carries the reference file for the motion control. As could be noted, the speed of the servo motor is linked as input and output to the constraint block controlling this block in accordance with the motion control profile.

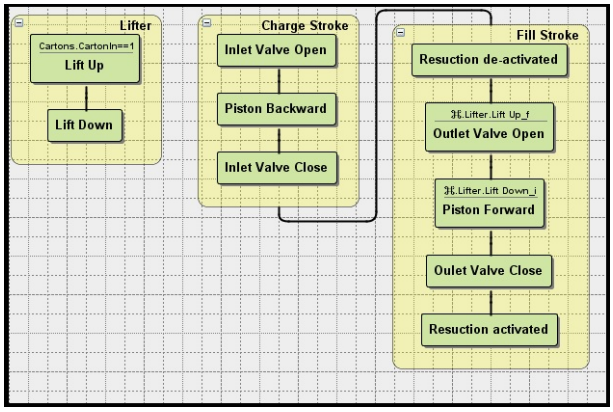


Fig. 10. Sequence Planner model of Filling module of TR/28.

The control-flow describing the sequence in which the actions to take place and the object-flow of the material to and from the actions are depicted in the activity diagram as shown in Figure 9 (dotted and solid line respectively). The interruptible region is modeled for repeating the procedure and is only stopped by a signal action called *Shutdown*. These actions are associated with the structural parts of the system. Each action (a_3) is also separately *'allocatedTo'* particular part or block (*Carton Lifter*) of the system. The structure and behavior models are linked and it is possible to view the entire model of the system both mechanically and logically.

5.2 Sequence Planner Model

The sequence planner model of the filling module of TR/28 machine is shown in Figure 10. The activities of the machine are grouped and each group's activities are modeled within the same group as operations. This type of relation (operation within operation) is called Hierarchical type. As seen from the example shown in Figure 10, the operation set *Charge Stroke* has to be finished completely in order to proceed to the next set of operations called *Fill Stroke*. The *Lifter* operation set will be initiated in parallel to the *Charge Stroke* operation itself. The pre- and post- conditions of these self-contained operations are also defined. By default, the transition between operations specifies that the preceding operation has to be finished before initiating the consecutive operation. The other pre-conditions could be included algebraically.

In this case, the outlet valve can open only when the lifter is in its up position and hence $Liftup\_f$ is assigned as a pre-condition for the operation *Outlet Valve Open* where $Liftup\_f$ is the final location of operation Lift up. Similarly, the lifter can move up only when there is carton available, and *CartonIn* is a variable assigned for denoting the signal received from the machine when a carton is sensed in the lift. This signal condition is added as a pre-condition for the operation *Lift Up*. The resources *Lift* and *Piston* are allocated to their respective operations. In this case, resource booking is not necessary as a single operation uses the single resource. Variables are also created to represent other sensor signals like *inValveOpen, inValveClose, liftInHome, liftInFillPos* and so on, while the sequences are modeled for the module.

This case study of Filling Module of a filling machine in a packaging industry, projects the nature of modeling between the two languages (SysML and SPL). SysML, gives importance to the detailed description of the system in a semi-formal modeling methodology. On the other hand, SPL concentrates on the behavioral modeling of the system in a formal way, also including a simple methodology.

6. CONCLUSIONS AND FUTURE WORK

Two significant languages for modeling manufacturing systems are compared and their pros and cons are analyzed. SysML, the language from UML which has emerged into the system engineering line is huge and able to structurally and behaviorally construct the system while, SPL is a small language concentrating on the behavioral aspects or work-flow, of the manufacturing system trying to solve the problems that arise. As analyzed in the comparison study of these languages, when considering the resource booking, SPL has defined a simpler way compared to the method of SysML. SPL also has a firm method to give projections of operation, which could be included to SysML that already has a structure for visualization, but not a method. Similarly, in other cases as well, SPL accommodate features that are not available in SysML but useful for System engineering applications. In comparing these two languages focusing on behavioral constructs, Sequence Planner language's method can be combined with SysML's behavioral constructs, which in turn helps SysML to widen the scope of formal modeling and include process planning along with verification and validation. Hence, utilizing SPL's benefits on behavioral model along with SysML would be a research area of interest in the future in order to achieve a better model description for system engineering applications. The behavioral construct of the complete system model in SysML could be imported to SPL and after analyzing and modifying the model, it could be synchronized with the existing model in SysML.

ACKNOWLEDGEMENTS

This work was carried out in collaboration with and support from Tetra Pak as well as the Wingquist Laboratory VINN Excellence Center within the Area of Advance Production at Chalmers, supported by the Swedish Governmental Agency for Innovation Systems (VINNOVA). The support is gratefully acknowledged.

REFERENCES

Bassi, L., Secchi, C., and Bonfe, M. (2011). A SysML-Based methodology for manufacturing machinery modeling and design. *IEEE/ASME Transactions on Mechatronics*, 16, 1049–1062.

Bengtsson, K., Bergagrd, P., Thorstensson, C., Lennartson, B., Akesson, K., Yuan, C., Miremadi, S., and Falkman, P. (2011). Sequence planning using multiple and coordinated sequences of operations. *Accepted for publication in IEEE Transactions on Automation Science and Engineering*.

Debbabi, M., Hassaine, F., Jarraya, Y., Soeanu, A., and Alawneh, L. (2010). *Verification and Validation in Systems Engineering*. Springer, Berlin.

Drath, R., Luder, A., J-Peschke, and Hundt, L. (2008). AutomationML - the glue for seamless automation engineering. *IEEE International Conference on Emerging Technologies and Factory Automation. ETFA*, 616–623.

Falkman, P., Nielsen, J., Lennartson, B., and von Euler-Chelpin, A. (2008). Generation of STEP AP214 models from discrete event systems for process planning and control. *IEEE Transactions on Automation Science and Engineering*, 5, 113–126.

Forsberg, K. and Mooz, H. (1998). System engineering for faster, cheaper, better. *Center for Systems Management*.

Friedenthal, S., Moore, A., and Steiner, R. (2009). *A Practical Guide to SysML*. Morgan Kaufmann OMG Press, Burlington, MA, USA.

Grobhshtein, Y. and Dori, D. (2008). Evaluating aspects of systems modeling languages by example: SysML and OPM. *INCOSE*.

Lennartson, B., Bengtsson, K., Yuan, C., Andersson, K., Fabian, M., Falkman, P., and Akesson, K. (2010). Sequence planning for integrated product, process and automation design. *IEEE Transactions on Automatic Science and Engineering*, 7, 791–802.

Linhares, M.V., de Oliveria, R.S., Farines, J., and Vernadt, F. (2007). Introducing the modeling and verification process in SysML. *12th IEEE Internation Conference on Emerging Technologies and Factory Automation*, 344–351.

Magnusson, P., Sundstrom, N., Bengtsson, K., Lennartson, B., Falkman, P., and Fabian, M. (2011). Planning transport sequences for flexible manufacturing system. *Preprints of 18th World Congress of the International Federation of Automatic Control. IFAC*.

Makartetskiy, D. and Sisto, R. (2011). An approach to refinement checking of SysML requirements. *IEEE 16th International Conference on Emerging Technologies and Factory Automation.ETFA*, 1–4.

OMGSpecification (2010). OMG Systems Modeling Language (OMG SysML). URL `http://www.omg.org/spec/SysML/1.2/`.

Peak, R., Burkhart, R., Friedenthal, S., Wilson, M., Bajaj, M., and Kim, I. (2007). Simulation-based design using SysML, Part 1: A parametrics primer. *INCOSE*, LNCS 2102.

Reinhartz-Berger, I. and Dori, D. (2005). OPM vs. UML - experimenting with comprehension and construction of web application models. *Empirical Software Engineering, Netherlands*, 10.

Schleipen, M. and Drath, R. (2009). Three-view-concept for modeling process or manufacturing plants with automationML. *IEEE 14th International Conference on Emerging Technologies and Factory Automation. ETFA*, 1–4.

Skoldstam, M., Akesson, K., and Fabian, M. (2007). Modelling of discrete event systems using finite automata with variables. *In Proc. 46th IEEE Conference on Decision and Control*.

Somekh, Y., M.Peleg, and Dori, D. (2007). Classifying and modeling exceptions through Object Process Methodology. *International Conference on Systems Engineering and Modeling*, 60–70.

TetraPakDocument (2007). OM operation manual, Tetra Rex TR/28 XH.