A transformation method of OPM Model to CPN Model for System Concept Development

Wenlu Zhou[1,a],Feng Yang[1,b],Yifan Zhu[1,c]
[1]College of Information System and Management
National University of Defense Technology
Changsha, Hunan,410072,China
[a]hbyczwl@163.com,[b]yf.nudt@gmail.com,[c]yfzhu@nudt.edu.cn

***Abstract*—Modeling languages for concept development in System Engineering usually provide a static model and lack computational capability. We introduce and implement a method combing the Object Process Methodology (OPM), a holistic modeling language well suited to describe the concept model, and Coloured Petri Net (CPN), an executable modeling language supporting elaborate simulation and analysis to make the process of System Engineering more continuous. Not only the basic entities and links, but also the hierarchical properties are converted from OPM to CPN according to the rules we proposed. Application in a simple air defense system demonstrates the process develop a concept model of OPM to a preliminary simulation model of CPN by using this method.**

Keywords—Object Process Methodology; Coloured Petri Nets; transformation; System Engineering

I. INTRODUCTION

Concept development is the primary and important phase in System Engineering, since the change of it costs less and affects more. A lot of modeling languages were introduced to help understanding structures and behaviors of a system and developing a conceptual model in this phase, such as Unified Modeling Language (UML)/System Modeling Language (SysML), Object Process Methodology (OPM) and so on. Despite these languages trying to describe the dynamic behaviors of a system, they are still providing a static model and cannot fully describe the quantitative aspects. It may need simulation of the system model to explore the behaviors of the system and validate the conceptual model as well.

Viewing system as a whole, OPM is more consistent with the ideal of System Engineering. It provides a holistic and hierarchical model to describe a system while UML/SysML presents different aspects of a system in separated diagrams. Although the dynamic logic of an OPM model can be checked by animation, it still cannot deal with computational behavior, which is needed in many cases.

There are a lot of researches aimed at addressing this issue. S. Bolshchikov etc.[1] propose two concepts: Vivid OPM and OPM Matlab Layer, the second of which can creates Matlab code from an OPM model added a numerical computational layer and make it possible to simulate system's behavior quantitatively. F. Simon etc.[2] suggest the possibility of combining the executable meta-language called Object-Process Network (OPN) with modeling languages including OPM. Rengzhong Wang[3] proposed a holistic modeling method for architecture development by combining the capabilities of OPM, Coloured Petri Net (CPN) and feature model. Additional information defined with CPN semantics is extended in OPM, following by mapping this model to a CPN model according rules proposed. It is a significant exploration; however, there were some shortcomings could be improved. The additional information made it more difficult for architects to build a correct OPM model and it did not mention how to convert a hierarchical OPM model to a CPN model with subnets.

In this paper, a method transforming an OPM model to a CPN model by mapping OPM notations to CPN is developed and implemented. Section 2 briefly introduces the theory of OPM and CPN and points out the significance of the transformation from OPM to CPN. We introduce the transformation method in section 3, which mainly concerns the logical relationship between the different elements and procedure links in the OPM and does not need any additional information for the convert. It can also turn a hierarchical OPM model into a CPN model with subnets and keep the capability of describing complex systems of OPM to some degree. Section 4 presents an example to explain the application of the convert. Section 5 contains the conclusions and future work.

II. OPM AND CPN

A. Object-Process Methodology

Object-Process Methodology (OPM) is a holistic modeling language for understanding and developing systems developed by Dori [4]. It combines the object-oriented and process-oriented concepts and describes structure and behavior aspects of a system in a holistic model.

Entities and links are the main building blocks of OPM. Entities include states and things (Objects and Processes). Objects are existing things, and processes are things that transform the objects by generating, consuming, or affecting them. States are situations at which the objects can exist, and belong to the objects. There are two types of links: structural links and procedural links. Structure links express the static,

persistent relationship among objects or processes, while procedural links express the dynamics behavior of a system.

OPM adopts detail decomposition rather than aspect decomposition to manage the systems' complexity resulting in a holistic hierarchical model. OPM contains two representation modes, the graphic and textual which are semantically equivalent. Object-Process CASE tool (OPCAT) is a software environment supporting system development and lifecycle using OPM [5]

B. Coloured Petri Nets

Coloured Petri Nets (CP-nets or CPN) is a graphical oriented language for design, specification, simulation and verification of discrete event systems [6]. It combines the capabilities of Petri nets with the programming languages and has the ability to establish a hierarchical model.

The building blocks of CPN are places, transitions, tokens and arcs. Places describe the state of the system and transitions describe the actions. Arcs indicate how the state changes when the transitions occur and are presented with arc expressions. Each place contains a set of marks called tokens carrying data values which belongs to a given type corresponding to the place. The types of data values are referred as colour sets which make the tokens distinguishable from each other.

CPN Tools is a mature tool supporting editing, simulation and analysis of CPN. The inscription language is Standard ML. It has different simulation modes. Monitors can be used to observe, inspect, control or modify the simulation [7]. As for analysis, CPN Tools supports state space analysis and performance analysis.

C. Strengths and Weaknesses

OPM and CPN have their own strengths and weaknesses. OPM is well suited to describe the concept model in system development since it provides various general semantics to describe different systems and makes them easy to understand. The view taking a system as a whole is the nature of System Engineering. However, it cannot provide enough numeric analysis which is necessary in the following assessment and validation. CPN, on the other hand, is able to support elaborate simulation and analysis especially for concurrency, however it is difficult to build a completed and correct CPN model form the very beginning. Since the two modeling languages have their own strengths in system development and neither of them can demand the needs of system design and validation alone, it is a nature thought to combine them together to complement each other. The capability of establishing a hierarchical model ensures the possibility to describe complicated systems and is crucial for the transformation.

III. METHOD

According to the introduction above, there is a nature relationship between OPM and CPN. They both have a graphic representation. The essence of CPN is a state machine and OPM also describe the states of a system and their transformations through processes. As OPM does not have a precise mathematically definition like CPN, it is hard to give a logical representation of the covert method. One possible way is mapping the building blocks between OPM and CPN, so that models constructed by them can transform from OPM to CPN. The method is implemented by transforming the xml files.

A. The Conversion of Entities and Links

The entities in OPM are objects, processes and states. Processes are converted to transitions in CPN, since they both describe the behavior of change in a system. States are mapped to places, with the name in the format of "O_S", where "O" stands for the name of the object the state belongs to and "S" represents the name of the state. Objects that are connected to the process without states are mapped to places. If an object connected to the process has one or more states, it does not need to be converted to a CPN place, because the procedural links will change the end to its states. Objects do not have a procedural link will not be mapped to CPN.

As CPN is mainly concerned the behavior aspect of a system, structural links are not mapped to CPN. Those procedural links are divided into four types for the convert to CPN: (1) consumption links, (2) instrument links, (3) result links and (4) process links.

Consumption links include consumption link and consumption event link which link from an object or a state to a process. These links are mapped to arcs from places to transitions. If the source of the link is an object with states, then build a set of arcs the sources of which are places converted from every single state of the object and add a XOR relationship among the arcs. Instead of an OR relationship, a XOR relationship fixes the number of tokens consumed in the transition. The conversion of links in following part will use a XOR relationship instead of an OR relationship as well. The XOR relationship is represented by the structure of CPN. The destinations of the set of arcs are different new transitions with the name of "P_Transition", where "P" is the name of the source place. Then link those transitions to a new place named with the original object and then connect this new place to the transition converted from the original process as shown in the Table 1. If there is only one state of the source object, just change the source of the link to the place converted from the state.

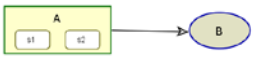
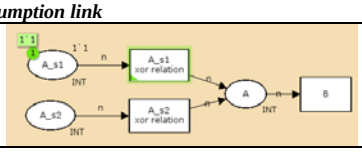
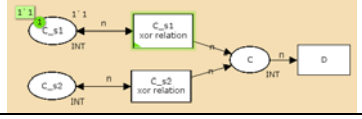
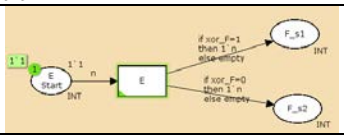
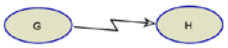
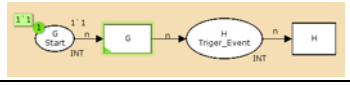
Instrument links include agent link, instrument link, effect link, instrument event link and condition link. Agent link and effect link connect an object to a process, and the rest link from an object or a state to a process. Those objects and state can trigger the process without being transformed, so the links are mapped to bidirectional arcs between places and transitions. If the source is object with states, it is similar as the consumption links except that the set of arcs are bidirectional. See Table 1.

Result link which connect a process to an object or a state, are mapped to arcs from transitions to places. If the destination of the link is an object with states, then build a set of arcs the destination of which are places converted from very state of the object and the source of which is the transition converted from the process. There is a XOR relationship among the arcs which is represented by the arc expression. To build a XOR

relationship, a new colour set will be declared with the name of “Xor_O”, where “O” is the name of the object. The colour set is a type of integer with the range of 0 to m-1, where m stands for the number of the states in the object. Then there will be a variable named “xor_O” which is a type of “Xor_O” and the value of it indicating a particular state. The arc expression is “if xor_O = i then 1`n else empty” where i is the number representing each states. Once the transition fires, the variable “xor_O” will be given a random value and if it equals the number representing the state, it will pass the token to the place transformed from the state, else it will pass nothing. Table 1 shows the case.

Process links include invocation link and exception link, which connect two processes. A new place will be added between the two transitions converted from the processes with the name of “T1 Trigger_Event” or “T2_Exception”, where “T1” and “T2” represents the name of destination process and source process respectively. A process link is mapped to a link from the source transition to the addition place and a link from the place to the destination transition. See Table 1.

TABLE I. SPECIAL CASES IN CONVERSION OF LINKS

OPM	CPN
consumption link	
 A can be s1 or s2. B consumes A.	
instrument link	
 C can be s1 or s2. D requires C.	
result link	
 F can be s1 or s2. E yields F.	
invocation link	
 G invokes H.	

B. *The Conversion of Hierarchy*

In OPM, the main mechanism to manage systems’ complexity and to build a hierarchical model is in-zooming/out-zooming makes a set of lower-level things enclosed with a thing visible /invisible. There will be a new OPD for each thing that is zoomed in and the set of OPDs will be connected by the zoomed in things. In CPN, the main mechanism is substitution transitions. A substitution transition is a transition that stands for a whole page of net structure. The places related to the substitution transition are socket places. The places in subpages functioning as the interface through which subpages communicate with their parent pages are called port places. Port places are marked with an In-tag, Out-tag or I/O-tag representing an input port, output port or Input/output port respectively. Each port place of the subpage is assigned to a socket place of the substitution transition and they are functionally identical.

Since in-zooming/out-zooming is prior for processes, those processes zoomed in will be mapped to substitution transition and objects zoomed in will not be mapped to CPN. The rules for turning a hierarchical OPM model to a CPN model with subnets are as follows.

- A process which is zoomed in will be mapped to a substitution transition.
- If the main process i.e. zoomed in process is enclosed with one or more sub-processes, the subpage will not contain the transition converted from the main process, and only presented the sub-transitions.
- If the main process is not contained in the subpage, those links connected to it will be change to the sub-process. If the main process is the destination of a link which is a type of consumption links, instrument links and process links, then change the destination of the link to the first sub-process. If the main process is the source of a link, which is a type of result links and process links, then change the source of the link to the last sub-process. The order of the sub-processes is defined by their locations as the timeline in an OPD flows from the top to the bottom. The first sub-process is the one at the top and the last sub-process is the one at the bottom within the main process.
- If an object appears in both parent OPD and zoomed in OPD and satisfies the condition of transforming to a place, the object in parent OPD will be a socket place and the object in zoomed in OPD will be a port place. The tag of the place is defined according to the relationship between the object and the main process. If there is only a type of consumption links, it will be an In-tag marked with the place and if there is only a type of result links, it will be an Out-tag. If there is a type of instrument links, it will be an I/O-tag. It is the same with the states within an object which appears in both parent OPD and zoomed in OPD.
- The newly added places in the transformation of consumption links and process links will also considered in the conversion of hierarchy if the links appears in both parent OPD and zoomed in OPD.

There are some additional rules for completing the CPN model.

- The default declaration of the colour sets and variables is defined to ensure the integrity of model logic and make it possible to run. The default colour set is “INT” which is a type of integer and a default variable is “n” which is a type of “INT”. All the places are type INT and all the arc expressions are “n” which means each token carrying an integer as its data value. User-defined declarations and arc expressions can also be used to solve the specific problems.
- Places without any arc from a transition to itself will have initial tokens. The default initial mark is “1`1”,

which means there is one token the data value of which is one. Only one of those places transformed from a set of states in an object will have an initial token indicating the object state at the beginning.

- If a process has no input which means it is not the destination of any links, there will be an additional place named “P Start”, where “P” stands for the name of the process and the place will connect to the transition converted from the process.

IV. Application in a Simple Air Defense System

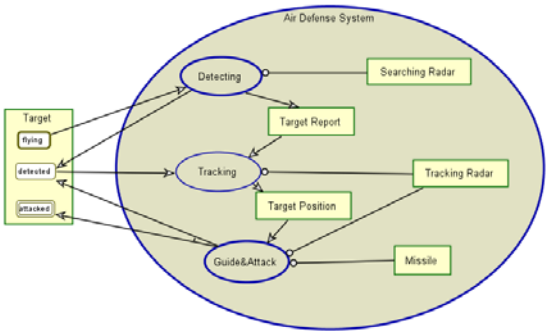


Fig. 1. *OPD of Air Defense System*

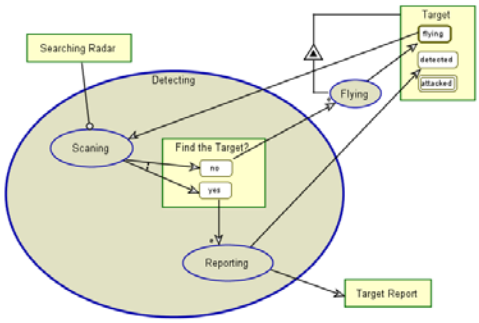


Fig. 2. *OPD of the in-zoomed Detecting process*

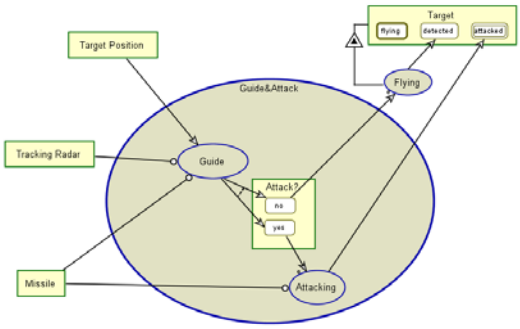


Fig. 3. *OPD of the in-zoomed Guide&Attack process*

An air defense system consists of searching radar, tracking radar, missile and command centre. Searching radar is respond for detecting the target and reporting it to the command centre, tracking radar is respond for tracking the target accurately and guiding the missile to attack the target. Missile is the weapon to attack and command centre is respond for communicating and control. We simplify the model and ignore the command centre assuming that the radars and missile can communicate well with each other. The OPM model of the simple air defense system is shown in Figure 1. The detecting process and guide and attack process can be zoomed into a new OPD and details shown in Figure 2 and Figure 3.

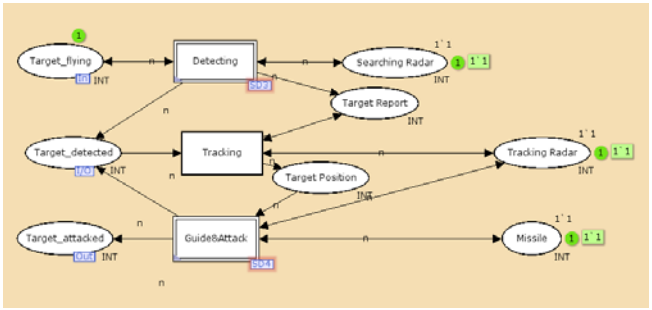


Fig. 4. *CPN of Air Defense System*

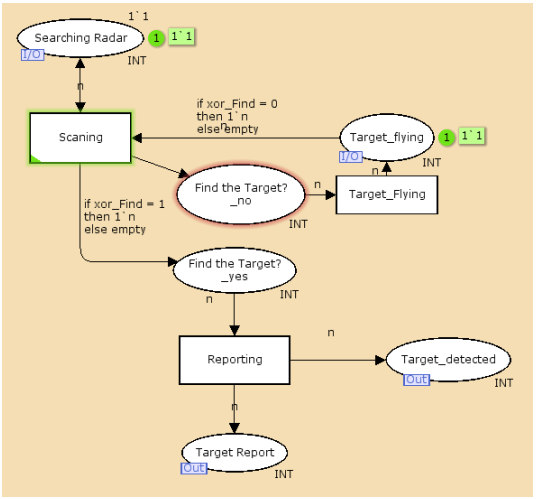


Fig. 5. *Subnet of the Detecting process*

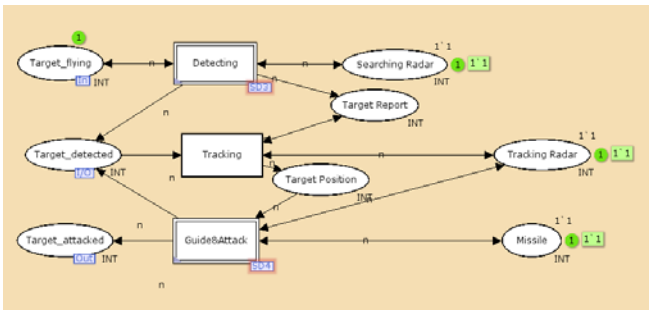


Fig. 6. *Subnet of the Guide&Attack process*

After the transformation, we can get corresponding CPN shown in Figure 4, 5 and 6.It can be simulated and validated the correctness of its logic. Additional formulations and possibilities can be added into the arc expressions and the model can calculate the measurement concerned during the

simulation, like the position and speed of the target and the missile, how different possibilities of detecting affect the time of finding the target and so on.

V. Conclusion and Further Work

In this paper, we introduce and implement a method transforming an OPM model to a CPN model by mapping OPM notations to CPN. Not only the basic entities and links, but also the hierarchical properties are converted from OPM to CPN according to the rules we proposed. Combing the capabilities of describing concepts of OPM and the capabilities of simulation and analysis of CPN, it can help develop a model from concepts to details in System Engineering. Application in a simple air defense system demonstrates the process develop a concept model of OPM to a preliminary simulation model of CPN by using this method. It can make the process of concept development more continuous to some degree and make it easier to develop, analysis and validate the model.

Further work based on this research might include following topics:

- Transformation of the concept of time in OPM to build a timed CPN model.
- Transformation based on meta-model and formulation definition of OPM and CPN.

Acknowledgment

This research was supported by Natural Science Foundation of China (61074107,91324014).

References

[1] Bolshchikov, S., Renick, A., Somekh, J., & Dori, D. OPM Model-Driven Animated Simulation with Computational Interface to Matlab-Simulink.

[2] Simona, F., Pinheiro, G., & Loureiro, G. (2007). Towards Automatic Systems Architecting. In Complex Systems Concurrent Engineering (pp. 117-130). Springer London.

[3] Wang, R. (2012). Search-based system architecture development using a holistic modeling approach.

[4] Dori, D. (2002). Object-Process Methodology: A Holistic Systems Paradigm; with CD-ROM (Vol. 1).

[5] Dori, D., Reinhartz-Berger, I., & Sturm, A. (2003, April). OPCAT-A Bimodal Case Tool for Object-Process Based System Development. In ICEIS (3) (pp. 286-291).

[6] Jensen, K. (1997). A brief introduction to coloured petri nets. In Tools and Algorithms for the Construction and Analysis of Systems (pp. 203-208). Springer Berlin Heidelberg.

[7] Wells, L. (2006, October). Performance analysis using CPN tools. InProceedings of the 1st international conference on Performance evaluation methodolgies and tools (p. 59). ACM.