Formal Approach to Dynamic SoS Design

Hela Kadri[1,2], Simon Collart-Dutilleul[1], Philippe Bon[1] and Samir Ben Ahmed[2]
[1]*IFSTTAR/ESTAS, Université Lille Nord de France, 20 rue Elisée Reclus, France*
[2]*Université de Tunis El Manar, Campus Universitaire Farhat Hached, Tunisia*

Keywords: System of Systems, Petri Nets, Discrete-event Systems, Operating Modes, Control Design, Reconfiguration, Supervisory Control Theory.

Abstract: This paper deals with the problem of System-of-Systems (SoS) modeling whose structure change due to the dynamics of its constituent systems as they are developed in different operating modes. Designed by using multi-model approach, each operating mode represents a process functioning of its system, depending on the resources to activate and requirements to enforce. Studied as a Discrete-Event System (DES), we propose a hierarchical framework for designing suitable switching control for dynamic SoS using the High Level Petri Nets (HLPN). Following a bottom up approach, we model each component first in a separate sheet and then the operating modes. Next, systems are designed by integrating a switching mechanism to commit to different operating modes when switching events occur. Finally, a mechanism for managing systems dependencies inside the SoS is proposed. Algorithms are provided to generate the different layers of HLPN permitting easy implementation of our framework and a flexible manufacturing SoS is used to illustrate our approach.

1 INTRODUCTION

In recent years, complex systems have seen an increasing interest; they became large and work together to achieve common goals. In this context, the concept of a System-of-Systems (SoS) has been proposed. It offers a high-level viewpoint encompassing the interactions between the constituent systems (Jamshidi, 2008). The SoS concept has applied in many fields including military (Huynh and Osmundson, 2006), robotics (Jamshidi, 2008), transportation (Caballini et al., 2012) and crisis management (Kadri et al., 2018). As the malfunction of a single system can have some serious consequences on the performance of the whole SoS, it's important that the design of SoSs takes into account the reliability and robustness requirements of its operation safety. Starting from this necessity, the objective of this paper is the design in a formal way of SoSs with maintaining an acceptable SoS operation despite failures occurring. To reduce complexity, we adopt a hierarchical bottom-up design approach using High Level Petri Nets (HLPN). First, we model all components of an SoS in different models. Next and to avoid to handle the whole process and the whole specification, we adopt the multi-model approach to design, for each system of the SoS, the behavior of its operating modes in separate models. In this step, we handle the problem of common components between operating modes, especially that of states. Then, we present a framework devoted to operating mode management under specific control based on supervisory control theory (Ramadge and Wonham, 1989). In this theory, systems are allowed to switch from one operating mode to another one, when exceptional events occur. Such events may be a failures, failed resource recoveries, maintenance, etc. Finally, we introduce, in an upper layer, a framework for modeling interactions and dependencies between systems in order to maintain acceptable SoS operation despite failures occurring. In the following sections, we present the developed work in its different stages and aspects.

2 PRELIMINARY

2.1 SoS Concept

The (ISO/IEC/IEEE 15288 Annex G (ISO, 2015) defines an SoS as "[...] a set of systems [brought together] for a task that none of the systems can accomplish on its own. Each constituent system keeps its own management, goals, and resources while coordinating within the SoS and adapting to meet SoS goals." However, several definitions, which are not

Kadri, H., Collart-Dutilleul, S., Bon, P. and Ben Ahmed, S.
Formal Approach to Dynamic SoS Design.
DOI: 10.5220/0007730903770384
In *Proceedings of the 14th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2019)*, pages 377-384
ISBN: 978-989-758-375-9

universally accepted, appeared before. For example, (Checkland, 1999) defines SoS as "two or more systems that are separately defined but operate together to perform a common goal". The Department of Defense American (DoD, 2008) consider an SoS as "a set or arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities." (Maier, 1998) proposes five traits, known as Maier's criteria, for distinguishing SoSs: Operational Independence of Elements, Managerial Independence of Elements, Evolutionary Development, Emergent Behavior, and Geographical Distribution of Elements.

2.2 Operating Mode Management for SoS

In this paper and to study its dynamic, SoS is studied as a Discrete-Event System (DES). There are several methods proposing safe control and able to cope with systems complexity in the DES domain. Among these methods, Supervisory Control Theory (SCT) seems to be more convenient for mode management by distinguishing process and specifications. Initiated by (Ramadge and Wonham, 1989), SCT is the theory of analyzing discrete event control systems and it allows the definition of different control strategies for each operating mode based on operating modes management. This technology aims to ensure the switching from one operating mode to another one according to the user input and safety requirements and it is the subject of several researches (Kamach et al., 2003; Kadri et al., 2013; Kadri et al., 2017).

In order to define separate behavior of each system of an SoS, we consider the multi-model approach. This approach assumes that only one operating mode is activated at a time for each system, while the others are deactivated. This allows us to define separate models for each system. Each model is a process functioning description represented by a HLPN and the process is made up of several components.

2.3 High Level Petri Net (HLPN)

HLPNs are a graphical and mathematical modeling tool. Among the types of HLPN, we use particularly the Hierarchical Prioritized Colored Petri Net (HPCPN).

2.3.1 Prioritized Colored Petri Net (CPN)

A CPN is a 7-tuple $< P,T,K,W_-,W_+,\Phi,M_0,\Pi >$ where P is a set of places; T is a set of transitions $(P \cap T = \emptyset \ , \ P \cup T \neq \emptyset)$; K is a color domain function defined from $P \cup T$ into finite and non-empty sets; W_-,W_+, defined on $P \times T \rightarrow \mathbb{N}$, are the backward and forward incidence functions, respectively, which specify the arcs connecting places and transitions; Φ is a guard function, it is a boolean expression attached to a transition $t \in T$ and by default $\Phi(t)$ is evaluated to true; M is a function defined on P describing the initial marking; Π is a priority function. It maps transitions into non-negative natural numbers representing their priority level.

2.3.2 Hierarchical Colored Petri Net (HCPN)

HPN allows to split models of large systems into manageable modules with well-defined. HCPN allows to split models of large systems into manageable submodules with well-defined. It permits to work at different abstraction levels and have the model reflect the structure of the system. It also allows to create building submodels that are used repeatedly in the HCPN model.

3 HIERARCHICAL SOS DESIGNING

3.1 Proposed Approach

The proposed approach has the purpose to provide an bottom up approach to SoS design with all its systems and with the actions of reconfiguration in a hierarchical way. The first step is Components Design in which components are modeled in separate HLPN sheets in according to their specifications. The second step is called Modes Design and it consists to study independently and separately each operating mode with applying conventionally the SCT. In fact, in any system, operating modes engage a set of components and fulfil requirements that may be very different from other ones. In the proposed approach, and through the use of hierarchical Petri nets, an operating mode model is a top layer to component models containing substitution transitions, each one is associated with a component's subnet. Furthermore, a notion of common component is presented in this step to reduce the size of the global model. The third step is Systems Design and it allows to model each system with its actions of reconfiguration. Each system model is a top layer of its operating mode models and it integrates a switching mechanism allowing to commit from one mode to another. The last step is SoS Design in which the dependencies between the systems are modeled following a request for reconfiguration. This step models of

the upper layer of the presented model.
Next subsections detail each step. But first, let denote $S = \{S_1, S_2, .., S_{|S|}\}$ the set of all systems of an SoS where $|S| > 1$, $M_i = \{M_{i,1}, M_{i,2}, .., M_{i,|M_i|}\}$ the set of operating modes of a system S_i, $i \in \{1..|S|\}$ where $|M_i| \geq 1$ and $C_i = \{C_{i,1}, C_{i,2}, .., C_{i,|C_i|}\}$ the set of components of S_i where $|C_i| \geq 1$. Let also $C_{M_{i,j}} \subseteq C_i$ be the set of components that made up the mode $M_{i,j}$, $M = \{M_1, M_2, .., M_{|S|}\}$ be the set of all operating modes and $C = \{C_1, C_2, .., C_{|S|}\}$ be the set of all components.

3.2 Components Design

Each system is made up by several components whose behavior is the same regardless of the system mode and it includes possible failures and recoveries and communication ports. At this stage, we aim to model components in separate pages. So, for each component, we associate a CPN describing its component behavior.

Definition 1. *Let S be the set systems of an SoS and let C_i be the set of components of a system S_i where $S_i \in S$. A component $C_{i,j}$ where $i \in \{1..|S|\}$ and $j \in \{1..|C_i|\}$ is modeled by a CPNs with $< P^{C_{i,j}}, T^{C_{i,j}}, K^{C_{i,j}}, W_-^{C_{i,j}}, W_+^{C_{i,j}}, \Phi^{C_{i,j}}, M_0^{C_{i,j}} >$ where:*

- $P^{C_{i,j}} = P_{\circlearrowleft}^{C_{i,j}} \cup P_{\leftrightarrows}^{C_{i,j}}$ *with* $P_{\circlearrowleft}^{C_{i,j}} \cap P_{\leftrightarrows}^{C_{i,j}} = \emptyset$. $P_{\circlearrowleft}^{C_{i,j}}$ *and* $P_{\leftrightarrows}^{C_{i,j}}$ *are, respectively, internal places and fusion (shared) places of the component $C_{i,j}$.*
- $T^{C_{i,j}} = T_{\circlearrowleft}^{C_{i,j}} \cup T_{\leftrightarrows}^{C_{i,j}}$ *with* $T_{\circlearrowleft}^{C_{i,j}} \cap T_{\leftrightarrows}^{C_{i,j}} = \emptyset$. $T_{\circlearrowleft}^{C_{i,j}}$ *and* $T_{\leftrightarrows}^{C_{i,j}}$ *are, respectively, internal transitions and switching transitions of the component $C_{i,j}$.*

3.3 Modes Design

The objective of this section is to ensure the internal behavior of the operating modes and that they are well-built and reliable according to the requirements. The commutation from one operating mode to another leads to change of the process model structure by engaging new components and by releasing others ones. Therefore, each mode has to fix the set of components necessary to perform its tasks according to the specification. It has also to differentiate between its own components, used in only one operating mode, and its common components, used in two or more operating modes of the same system. Formally,

Definition 2. *Let $S_i \in S$ and $M_{i,j} \in M_i$.*
*A component $C_{i,k}$, $k \in 1..|C_i|$, is called **own** to $M_{i,j}$ if $C_{i,k} \in C_{M_{i,j}}$ and $\forall M_{i,p} \in M_i$, such as $p \neq j$, $C_{i,k} \notin C_{M_{i,p}}$.*
*A component $C_{i,k}$, $k \in 1..|C_i|$, is called **common** to $M_{i,j}$ and $M_{i,p}$ if $C_{i,k} \in C_{M_{i,j}} \cap C_{M_{i,p}}$ with $j \neq p$.*
Let denote $C_{M_{i,j}} = C_{M_{i,j}}^{\circlearrowleft} \cup C_{M_{i,j}}^{\leftrightarrows}$ where:

- $C_{M_{i,j}}^{\circlearrowleft}$ *is the set of its own components;*
- $C_{M_{i,j}}^{\leftrightarrows}$ *is the set of its common components.*

Conceptually, we aims to model the operating modes by applying the multi-model approach, which involves designing a process model structure for each operating mode through the use of hierarchical CPN bottom-up. Each mode model is a superpage containing substitution transitions, each of which is associated to a subpage representing one model component. Superpages and subpages are connected by substitution transitions and by equating places on the two pages.
For common components, substitution transitions refer to the same subpage. So other instances of the selected component are created. Then, each place of the different instances is fused with the corresponding one to ensure an instances identical functioning representing the same common component. Accordingly, any state change made to one instance applies to all other instances as well. This keeps the behavior of common components when switching from one mode to another.
In the following Algorithm 1, we present the different steps to create mode models. First, for simplicity reasons, we confuse the notation $M_{i,j}$ of a mode identity and its associated HCPN and the notation of $C_{i,j}$ of a mode identity and its associated CPN.

Nevertheless, these mode models are not interconnected, and this is the main focus of the next section.

3.4 Systems Design

This section focuses on the design of all the systems comprising the SoS. Each system is designed and developed separately by taking the behavior that could lead to switching between modes into account. This guarantees a functioning under failure, whilst causing degraded production. The proposed framework allows to model each system as a superpage for its mode pages in which we handle the commutation problem and we add substitution transitions, each of which is associated to a subpage representing an operating mode.
In the presented framework, the switching mechanism is modeled by the $T_{\leftrightarrows}^{C_{i,k}}$ transitions where $i \in \{1..|S|\}$ and $k \in \{1..|C_i|\}$. Firing such transition must disable the transitions of their corresponding HCPL mode and enable firing the transitions of

the destination HCPN mode. To distinguish this type of transitions, we define an application noted *target_mode* whose role is to associate with each transition its destination mode.

Algorithm 1: Pseudo-algorithm generating HCPN of the operating modes.

Input: the set S of systems; the set M of modes; the set $C_{M_{i,j}}$ of $M_{i,j}$ components $(i = 1..|S|, j = 1..|M_i|)$; the set of CPN C;
Output: the set of HCPN $M_{i,j}$

foreach *system,* $i \leftarrow 1$**to** $|S|$ **do**
  foreach *mode,* $j \leftarrow 1$**to** $|M_i|$ **do**
    Create page $M_{i,j}$;
    foreach *component* $C_{i,k} \in C_{M_{i,j}}$ **do**
      Add substitution transition, $T^{\Updownarrow}_{i,j,k}$, in $M_{i,j}$ associated with $C_{i,k}$;
      foreach $P^{C_{i,k}} \in P^{C_{i,k}}_{\leftrightarrows}$ **do**
        Add $P^{C_{i,k}}$ in $M_{i,j}$;
        Add port-type tag on $C_{i,k}$;
        Add arc connecting $P^{C_{i,j}}$ to $T^{\Updownarrow}_{i,j,k}$;
      end
      if $C_{i,k} \in C^{\Leftrightarrow}_{M_{i,j}}$ **then**
        foreach $M_{i,p}$, $p \leftarrow 1$**to** j **do**
          if $C_{i,k} \in C^{\Leftrightarrow}_{M_{i,p}}$ **then**
            Fusion $P^{C_{i,k}}_{\circlearrowleft}$ in instances of $M_{i,j}$ and $M_{i,p}$;
          end
        end
      end
    end
  end
end

Definition 3. *Let $S_i \in S$, $M_{i,j} \in M_i$ and $T^{M_{i,j}}$ be the related set of transitions.*
Let target_mode : $T^{M_{i,j}} \rightarrow M_i$ be a mapping such that target_mode(t) indicates the active operating mode after firing $t, \forall t \in T^{M_{i,j}}$.

Remark 1. *Let $T^{M_{i,j}}_{\leftrightarrows} \subset T^{M_{i,j}}$ be the related set of switching transitions. $\forall t \in T^{M_{i,j}}_{\leftrightarrows}$, t corresponds to* ***a switching mechanism*** *of mode $M_{i,j}$ leading to mode target_mode(t).*

Moreover, new elements must be added to each system S_i in order to ensure the switching mechanism:

- Class color *Mode*
- Place *Mode Mgt* S_i marked by one token representing the initial mode;
- Arcs connecting *Mode Mgt* S_i to non-switching transitions and labelled by M, where M is a variable defined on *Mode*
- $\forall C_{i,k} \in C^{\Leftrightarrow}_{M_{i,j}}$ and $\forall t \in T^{C_{i,k}}_{\circlearrowleft}$, a predicate is assigned and it is satisfied as soon as the token of *Mode Mgt* S_i has the value of a mode including the component.
- $\forall C_{i,k} \in C^{\circlearrowleft}_{M_{i,j}}$ and $\forall t \in T^{C_{i,k}}_{\circlearrowleft}$, a predicate is assigned and it is satisfied only when the appropriate token is in *Mode Mgt* S_i.
- $\forall C_{i,k} \in C_{M_{i,j}}$ and $\forall t \in T^{C_{i,k}}_{\leftrightarrows}$, t is connected to *Mode Mgt* S_i with an input label M, and with an output label set to its target mode.

Algorithm 2 describes the necessary steps to create a HCPN allowing the operating mode management of each S_i.

3.5 SoS Design

In this section, we create the top-level net of the HCPN in which we guarantee a coherent behavior of the whole SoS despite failures occurring. Indeed, we aim to maintain an acceptable SoS operation when a system switches to a degraded mode. We therefore develop the following idea: if a system goes into degraded mode, a subset of systems will be affected and will have to switch off or to switch to another degraded in which a minimum operation is ensured under a revised control policy.
First, a systems decomposition of the SoS should be modeled and verified checked against the specification. Figure1 illustrates an example of systems decomposition. There are three systems: S1, S2 and S3. S1 and S3 are composed of two modes: NM for nominal mode and DM for degraded mode while S2 is composed of three modes: NM for nominal mode and DM1 and DM2 for its two degraded modes. Initially, all the three systems are in NM. When S1 (resp. S2) switches to DM (resp. DM1), S2 (resp. S3) also has to switch to DM2 (resp. DM1). And when S1 (resp. S2) reverts to the initial mode, S2 (resp. S3) also has to return to MN. If S2 is in DM, the switching of S1 from NM to DM and vice versa does not influence the SoS. However, if S1 is in DM and S2 switches to DM1, then S3 should switch to DM; and if S2 return to DM2, S1 should return to NM.

At this top layer, a dependency mechanism is ensured through a new set of high priority transitions, called set of dependency transitions and noted $T_{\rightsquigarrow}$. Each transition of the layer level SoS has the role

of associating with a subset of systems their new operating modes according to an internal change of mode in one system. We define an application noted *dependency* whose role is to ensure reconfiguration relations between systems:

Algorithm 2: Pseudo-algorithm generating HCPN of systems.

Input: the set S of systems; the set M of modes; the set of HCPN $M_{i,j}$ $(i = 1..|S|, j = 1..|M_i|)$; the set of CPN C;
Output: the set of HCPN S_i

foreach *system*, $i \leftarrow 1$**to** $|S|$ **do**
  Create page S_i;
  $K = K \cup Mode$;
  Add place $Mode\ Mgt\ S_i$;
  $M_0(Mode\ Mgt\ S_i) = M_{i,1}$;
  foreach *mode*, $j \leftarrow 1$**to** $|M_i|$ **do**
    Add substitution transition, $T_{i,j}^{\Updownarrow}$, associated with $M_{i,j}$ in S_i;
    Add arc connecting $Mode\ Mgt\ S_i$ to $T_{i,j}^{\Updownarrow}$
    Add place $Mode\ Mgt\ S_i$ in $M_{i,j}$;
    Add arc connecting $Mode\ Mgt\ S_i$ to $T_{\circlearrowleft}^{C_{i,k}}$ in $C_{i,k}$;
    Add port-type tag on $M_{i,j}$;
    foreach $C_{i,k}$, $k \leftarrow 1$**to** $|C_i|$ **do**
      Add place $Mode\ Mgt\ S_i$ in $C_{i,k}$;
      foreach $t \in T^{C_{i,k}}$ **do**
        $W_{-}^{C_{i,t}}(Mode\ Mgt\ S_i, t) = M$;
        if $target\_mode(t) = M_{i,j}$ **then**
          $W_{+}^{C_{i,t}}(Mode\ Mgt\ S_i, t) = M$
        else
          $W_{+}^{C_{i,t}}(Mode\ Mgt\ S_i, t) = target\_mode(t)$
        end
      end
    end
    foreach $t \in T_{\circlearrowleft}^{C_{i,k}}$ **do**
      $\Phi(t) = \Phi(t) \wedge [M = M_{i,j}]$
    end
  end
end

Definition 4. *Let $S' \subseteq S$ be a subset of system, $M' \subseteq M$ be its set of related operating mode and $T_{\rightsquigarrow}$ be the set of dependency transitions. Let dependency : $T_{\rightsquigarrow} \rightarrow M^{|S'|}$ be a mapping such that dependency(t) indicates the activated operating mode of each S' after firing t.*

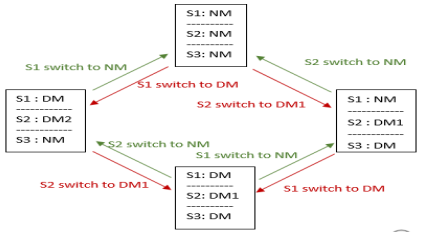


Figure 1: Example of systems decomposition inside an SoS.

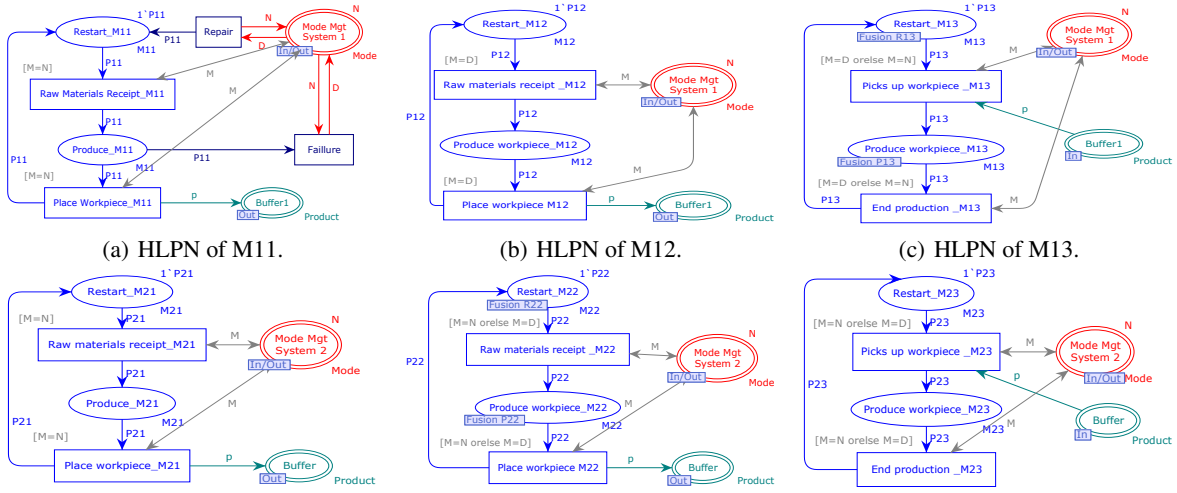
Algorithm 3: Pseudo-algorithm generating HPCPN of SoS.

Input: the set S of systems; the set M of modes; the set of HCPN $M_{i,j}$ $(i = 1..|S|, j = 1..|M_i|)$;
Output: the final HPCPN *SoS*

Create page *SoS*;
$K = K \cup ModeSoS$;
Add place *SoS Mode Mgt* ;
Add transitions $T_{\rightsquigarrow}$;
Add arcs *SoS Mode Mgt* to $T_{\rightsquigarrow}$;
$M_0(SoS\ Mode\ Mgt) = empty$;
foreach *system*, $i \leftarrow 1$**to** $|S|$ **do**
  $M_0(SoS\ Mode\ Mgt) = M_0(SoS\ Mode\ Mgt) + +(S_i, M_{i,1})$;
  Add substitution transition, $T_i^{\Updownarrow}$, associated with S_i in *SoS*;
  Add place $Mode\ Mgt\ S_i$ in *SoS*;
  Add arc connecting $Mode\ Mgt\ S_i$ to $T_i^{\Updownarrow}$;
  Add arc connecting $Mode\ Mgt\ S_i$ to $T_{\rightsquigarrow}$;
  $W_{-}(Mode\ Mgt\ S_i, t) = M_{i,1}$;
  $W_{+}(Mode\ Mgt\ S_i, t) = target\_mode(t)$;
end
foreach $t \in T_{\rightsquigarrow}$ **do**
  $W_{+}(SoS\ Mode\ Mgt, t) = M_0(SoS\ Mode\ Mgt)$;
  $W_{-}(SoS\ Mode\ Mgt, t) = dependency(t)$;
end

Some new elements also are added to this layer to ensure the dependency mechanism:

- Class color *ModeSoS* such as $ModeSoS = \cup(S_i, Mode)$;
- Place *SoS Mode Mgt* marked by as many tokens as systems in the SoS and each one contains the identity of its corresponding system and its initial mode;
- Transitions $T_{\rightsquigarrow}$;
- $\forall t \in T_{\rightsquigarrow}$, t is connected to the place *SoS Mode Mgt* with an input label set of systems current modes, and with an output label set to their new modes; t is connected to a



(a) HLPN of M11. (b) HLPN of M12. (c) HLPN of M13.

(d) HLPN of M21. (e) HLPN of M22. (f) HLPN of M23.

Figure 3: vHLPN of the SoS components.

Mode Mgt S_i with an input/output label M, where M is a variable defined on *Mode*.

Algorithm 3 creates the last layer of our framework that represents the whole management of the SoS.

4 CASE STUDY

4.1 System Description

We illustrate our contribution with a manufacturing SoS inspired from literature. This SoS features two systems, each one is composed by three components and one buffer as shown in Figure 2. *System*1

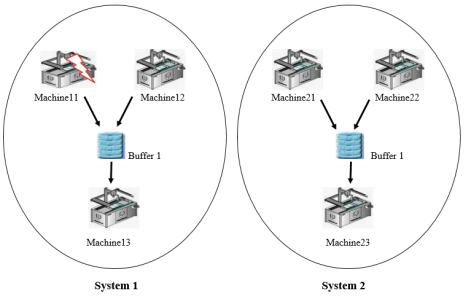


Figure 2: Manufacturing SoS.

treats workpieces type A. Thereafter, these workpieces are delivered to *System*2 in order to be used in the treatment of workpieces type B. The functioning of *System*1 is as follows: $M_{1,1}$ takes one by one workpieces from an upstream stock, achieves a treatment and then deposits it in an intermediate buffer. $M_{1,3}$takes one by one workpieces from the buffer, achieve a treatment and then deposit it in a downstream stock. Once workpieces type A arrive to *system*2, $M_{2,1}$ and $M_{2,2}$, which are quite similar, take two workpieces of which one is type type A from an upstream stock, achieves a treatment and then deposit in an intermediate buffer. Afterwards, the workpieces type B product is treated by the machine $M_{2,3}$.
It is assumed *System*1 can break down on $M_{1,1}$ and be repaired. If a failure occurs, $M_{1,2}$, which is less efficient than the defective machine, replaces it and the concerned system goes into degraded mode. A transition to a degraded mode of *System*1 influences the operation of the entire SoS: *System*2 will no longer be supplied with adequate quantities and will then go into degraded mode: $M_{2,2}$ goes into sleep mode.
The functioning of the example studied enables us to define for each system two operating modes: nominal and degraded. Formally, an $S = \{System1, System2\}$ where $System1 = \{M_{1,1}, M_{1,2}, M_{1,3}\}$ and $System2 = \{M_{2,1}, M_{2,2}, M_{2,3}\}$. The stock is not considered a component but rather as a system specification. This corresponded to a classic choice of SCT (Ramadge and Wonham, 1989) constraint because the stock is no generator events that its own.

4.2 Components Model

In our studied SoS of the Figure 2, there are 6 components spread over two systems. $C = \{M_{1,1}, M_{1,2}, M_{1,3}, M_{2,1}, M_{2,2}, M_{2,3}\}$.
For $system_1$, the functioning of $M_{1,1}$ is represented by the state machine of Figure3(a) made up of places $P^{M11} = \{StartM11, TreatmentM11\}$, transitions $T^{M11} = \{TakeWorkpieceM11, PlaceWorkpieceM11\}$, and

the related arcs labelled by the $P_1 1$ that is defined by the colour class $M11 = P11$. $M_0(StartM11) = 1`P11$ (one token $P11$). $T_{\leftrightarrows}^{M11} = \{Failure, Repair\}$, is the switching transitions. As in the following component models, place $P_{\leftrightarrows}^{M11} = \{Buffer1\}$ is a communication places, it aims to store workpieces produced; and the place *"Mode Management System S1"* is added in the system design step.

The functioning of $M_{1,2}$ (Figure 3(b)) is represented by places $P_{\circlearrowleft}^{M12} = \{StartM12, TreatmentM12\}$, transitions $T_{\circlearrowleft}^{M12} = \{TakeWorkpieceM12, PlaceWorkpieceM12\}$ and the arcs labelled by $P12$ defined on $M12 = \{P12\}$.

The behavior of $M_{2,2}$ (Figure 3(e)) is represented by places $P_{\circlearrowleft}^{M22} = \{StartM22, TreatmentM22\}$, transitions $T_{\circlearrowleft}^{M22} = \{TakeWorkpieceM22, EndTreatmentM22\}$ and arcs labelled by $P22$ defined on $M22 = P22$.

The behavior of $M_{2,3}$ (Figure3(f)) is represented by places $P_{\circlearrowleft}^{M23} = \{StartM23, AssemblageM23\}$, transitions $T_{\circlearrowleft}^{M23} = \{TakeWorkpiecesM23, PlaceWorkpieceM23\}$ and arcs labelled by $PM23$ defined on $M23 = P23$;

4.3 Modes Model

At this layer, we can notice, for $System1$ that M_{13} is engaged in its two operating modes, while M_{12} does not contribute to production in $Nominal1$ but it intervenes when a commutation to $Degraded1$ is performed. And for $System2$ that all machines are engaged in $Nominal2$, however only M_{21} and M_{23} are engaged in $Degraded2$. We then get $C_{M_{Nominal1}}^{\circlearrowleft} = M11$, $C_{M_{Degraded1}}^{\circlearrowleft} = M12$, $C_{Nominal1}^{\Leftrightarrow} = C_{Degraded1}^{\Leftrightarrow} = M13$, $C_{M_{Nominal2}}^{\circlearrowleft} = M22$ and $C_{Nominal2}^{\Leftrightarrow} = C_{Degraded2}^{\Leftrightarrow} = M21, M23$.

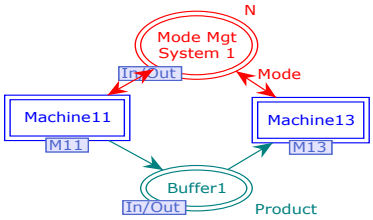
By applying Algorithm 1, we obtain four HCPNs, each one represents an operating mode of the studied SoS: Figure4(a) and Figure4(b) represent the operating modes of $System1$, while Figure4(a) and Figure4(b) represent those of $System2$.

4.4 Systems Model

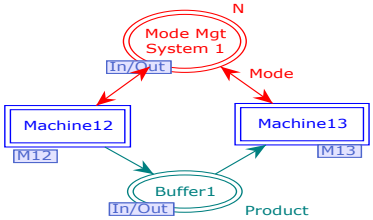
For $System1$, $Nominal1$ mode is composed of the components M_{11} and M_{13}. From this mode, a switch is possible to $Degraded1$, by the switch event *"Failure"* generated if the component M_{11} breaks down, $Degraded1$ is composed of the components M_{23}, common component with $Nominal1$, and M_{12}. When M_{11} generates the event *"recovery"*, $System1$ switches to the first mode.

In the same way, For $System2$, $Nominal1$ mode is composed of the components M_{21}, M_{22} and M_{23}. A switch is possible from $Nominal2$ to $Degraded2$ if a request for switching happens. Likewise, a switch to $Nominal2$ from $Degraded2$ can be performed following a switching request.

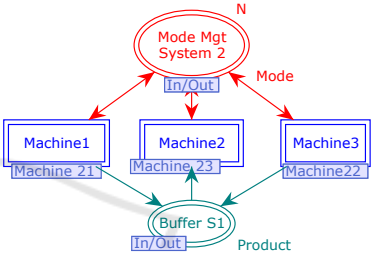
The third HCPN layers obtained is represented in Figure5(a) and 5(b).



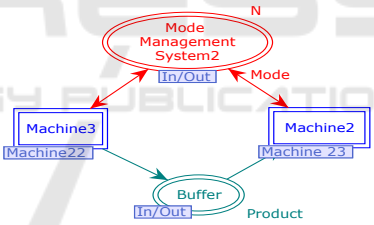
(a) HLPN of System1 nominal mode.



(b) HLPN of System1 degraded mode.



(c) HLPN of System2 nominal mode.



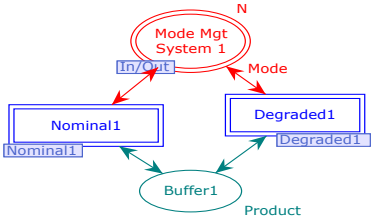
(d) HLPN of system2 degraded mode.

Figure 4: HLPN of the SoS modes.

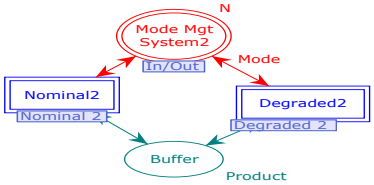
4.5 SoS Model

Figure6 shows systems dependencies inside our studied SoS: a switching of $System1$ to its degraded mode imposes on $System2$ a commutation to degraded mode and and vice versa.

The final layer of the study example is represented in Figure7 and it is obtained after generating Algorithm 3.



(a) A modal decomposition of System1.



(b) A modal decomposition of System2.

Figure 5: A modal decomposition of each system of the studied SoS, with the components used in each mode.

Figure 6: Systems decomposition.

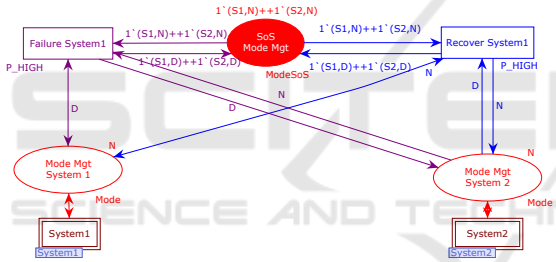


Figure 7: SoS design layer.

5 CONCLUSION

The contribution of this paper is to present a hierarchical framework using HLPN and allowing to control the dynamic inside and between systems of an SoS whose systems are modeled using multimodel approach. In fact, when an exceptional event occur in a system, it switches to another operating mode in order to maintain an acceptable functioning. This switching causes a reconfiguration inside the SoS. The proposed framework decomposes the systems of an SoS in several operating modes and each one is decompose in components. The first step of the framework is the component design where each component is designed independently. The second step is the mode design where operating modes are studied and also modeled independently on formal way. The third step focuses system design with all operating modes including its different switch dynamics by using SCT. The last step is to model the dependencies between the systems. Current research involves defining strategies when the similarities between components and modes can be noticed in order to reduce the complexity of the SoS.

REFERENCES

Caballini, C., Sacone, S., and Siri, S. (2012). The port as a system of systems: A system dynamics simulation approach. Proc. 7th Int. Conf. Syst. Syst, Genoa, Italy.

Checkland, P. B. (1999). Systems thinking, systems practice. Chichester, UK: John Wiley and Sons Ltd.

DoD (2008). System of systems engineering. In *Defense Acquisition Guidebook (DAG)*. Washington, DC: Pentagon.

Huynh, T. V. and Osmundson, J. S. (2006). A systems engineering methodology for analyzing systems of systems using the systems modeling language (sysml). Proc. 2nd Syst. Syst. Eng. Conf.

ISO/IEC/IEEE 15288 Annex G (ISO, . (2015). Systems and software engineering – system life cycle processes. In *Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers*.

Jamshidi, M. (2008). Systems of systems engineering: Principles and applications. New York, NY, USA: Taylor & Francis.

Kadri, H., Ahmed, S. B., and Collart-Dutilleul, S. (2017). Formal approach to control design of complex and dynamical systems. In *Procedia Computer Science 108C:2512-2516, pages: 2512-2516*. Elsevier B.V.

Kadri, H., Schleiner, S., Collart-Dutilleul, S., Bon, P., Ahmed, S. B., Steyer, S., Gabriel, F., and Aimé, A. O. (2018). Proposition of a formal model for crisis management in the context of high-speed train networks in border areas. In *the proceeding of Transport Research Arena 2018 (TRA 2018), 16-19th April, Vienna, Austria*.

Kadri, H., Zairi, S., and Zouari, B. (2013). Global model for the management of operating modes in discrete event systems. In *6th IFAC Conference Management and Control of Production and Logistics, Volume 6 — Part 1, Fortaleza, Brazil, September 11-13*.

Kamach, O., Pietrac, L., and Niel, E. (2003). Multimodel approach for discrete event systems: application to operating mode management. In *Multiconference Computational Engineering in Systems Applications, CESA, Lille*.

Maier, M. W. (1998). Architecting principles for systems-of-systems. In *Systems Engineering, vol. 1, no. 4, pp. 267–284*.

Ramadge, P. and Wonham, W. (1989). The control of discrete event systems. In *Proceedings IEEE, vol.77, num 1, pp. 81-98*.