Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/090438

Modeling of Adaptive Cyber Physical Systems using Aspect-oriented Approach

Zhilin Qian[1,2], *Huiqun Yu*[1,*] *and Guisheng Fan*[1]

[1] Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai 200237, China
[2] Shanghai Key Laboratory of Computer Software Evaluating and Testing, Shanghai 201112, China

Received: 8 Nov. 2014, Revised: 8 Feb. 2015, Accepted: 9 Feb. 2015
Published online: 1 Jul. 2015

Abstract: This paper proposes an aspect-oriented approach to modeling adaptive cyber physical system (CPS) using Petri nets. The core concerns of CPSs are described as device model and task model, and dynamic variations of system behaviors or environment conditions are extracted as crosscutting concerns. The models of runtime inspection as well as device adaptation and task adaptation are designed as aspects nets. For the device adaptation strategy, fault types are analyzed and the control loop concept is integrated to form the adaptation aspect model. For the task adaptation, a rescheduling method using PSO-Pareto algorithm to find the best solution of the backup devices is proposed. Via well-defined rules, these aspect nets can be weaved with the core concern nets into a comprehensive adaptive CPS model. By theoretical analysis and a case study, we show the modeling approach is feasible and flexible, which simplifies the design of adaptive CPSs.

Keywords: CPS, aspect-oriented method, Petri net, modeling, self-adaptation

1 Introduction

Cyber physical system is a system consists of computational elements and physical entities and their interactions. The physical entities include sensors, actuators and the interlink network. With its characteristics related but not restricted to those knowledge areas such as distributed system, real-time system, embedded system, wireless network sensors, ubiquitous computing, control theory and etc., CPS is definition-complicated, behavior-intricate and architecture-heterogeneous. As combination of computation, controlling and communication, CPS is not only sensitive to the system itself but also the environment surrounded[1]. As a CPS might be extended to ultra large in scale, the design method desires the characteristics of flexibility, versatility, resilience and dependency. Therefore, autonomous adaptation, also known as self-adaptation, needs to be considered in the construction of CPS as a runtime reconfiguration of the system according to the perceived behavior changes, quality of service changes and environment changes.

Petri net as a graphical description tool can strongly describe the system behaviors [2]. With formal specification associated, system behaviors can be analyzed, verified and validated. Due to its clear visual illustration of both structure and behaviors of the targeted system, Petri net can not only handle the concurrent events, but also well express the details of the system. Besides, Petri net provides an easier way to transfer the high-level system design to the low-level implementation. From the modeling perspective, the traditional Petri net is inadequate to deal with runtime adaptation of CPSs. We need to extend the Petri net theory and seek a novel way to demonstrate the CPS behaviors and runtime adaptation. The approach and the model should both be flexible and easy to be adjusted or expanded without interfering with the ultimate system goals.

Aspect-oriented programming (AOP) [3] is first presented as a modularized programming paradigm. It separates the concerns from the business descriptions. Traditional AOP methods and tools are concerned with program elements at much lower level of source codes. The concerns can be separated as core concern and crosscutting concern. The latter are usually relevant to non-functional properties that may crosscut multiple business abstractions. Elevating the idea to software engineering life-cycle, well-defined concerns can be any

* Corresponding author e-mail: yhq@ecust.edu.cn

distinct facets in the system logic. This concept motivates us to develop a modularized design approach with AOP and Petri net for adaptive modeling. In our former work, we propose a time-constrained aspect-oriented Petri net to model the dynamics of CPSs. We show that the method is feasible for capturing hybrid characteristics of CPSs with discrete events and continuous activities. In addition, the method supports separation of concerns in the design of CPSs.

Different from software entities, the physical processes are reflected in the Petri net model from two aspects. Firstly, the quantitative capacities of physical entities in CPSs are represented as markings of tokens in Petri nets. In contrast, software entities could be simultaneously invoked by several services anytime anywhere with no quantity limitation. Secondly, similar to the hybrid system theory presented in [4], we model the physical entities as tokens of place in the Petri net in a high abstract level. The tokens are generalized as continuous variables. The discrete actions of software entities are modeled as their value changes instantaneously and then trigger the next step of the model evolvement. Whilst, the continuous activities are modeled by real-valued variables whose values change continuously over time collapsing according to the laws of physics such as the differential equations, etc. Well-defined triggering rules will govern the model execution.

In our CPS design, the bottom of the architecture is described as component-based structure. To be more specific, physical devices are the major entities in the infrastructure and business processes are described as tasks. We first present the core concern models as the devices and tasks. Then we focus on the model of runtime monitoring and adaptation for fault-tolerance. According to fault types, we propose different adaptation strategies that in turns are modeled as aspects. All these models are presented using Petri nets. For the device adaptation, we analyze the device fault types and combine the control loop concept to form the adaptation aspect model. For the task adaptation, we propose a rescheduling method with PSO-Pareto algorithm to seek the best solution of the backup devices. With the well-defined crosscutting concern and pointcuts, we can weave the adaptation aspects with the core concerns, which results in a comprehensive CPS model. Theories of Petri nets help prove the correctness of the integrated CPS model.

2 Related Work

In research areas of software engineering, adaptation is not a brand new topic to be discussed. Its ideas and technologies have already been applied in various areas such as autonomic computing, autonomous robotics, multi-agent systems, machine learning and even the nowadays flourishing and prosperous areas like Cloud computing, Big data and the Internet of Things. Also some inspirations from the natural and biological systems have enlightened the adaptive implementation in software systems[5]. Since adaptation has partially been realized in some specific domains and its importance keeps increasing due to the current application situation, introducing self-adaptation idea into software systems is necessarily worthwhile. Cheng [5] and Kramer [6] et al. present a general self-managing three-layer architecture to implement adaptation. Dalpiaz et al. [7] present a self-reconfiguring method to deal with the multiple runtime needs. Salifu et al. [8] focus on the system self-healing method for system maintenance. Chung [9] focuses on the security aspect of the system, and proposes a self-protecting method to deal with attacks. However, the establishment of self-adaptation for CPS is little depicted and endeavored in a systematical way. Zhang et al. [10] present a model of fault diagnosis and using a sensor activation decision to configure the system to achieve the best quality satisfaction. Phan et al. [11] propose a multi-model framework for facility adaptation. Zhang et al. [12] present an agent based model and a cooperation mechanism to achieve multiple adaptive control. Other adaptation researches about CPS are mainly focus on specific application areas. Though efforts have been made on the researches of adaptive CPS, the appropriate methods of modeling and realization of CPS with the runtime adaptation still remain challenging.

As the CPS's intrinsic characteristic with the mutual interactions between the software and environment which also been called as the cyber world and the physical world, some of its behaviors are not predictable at the design and development phase [1]. Also, at the execution phase, the failures of hardware and software, and the degradations of the quality of services are inevitable either. So we need a runtime configuration and solution to deal with the adaptation needs. Therefore, the modeling methods should be flexible with the concerns of adaptations. Former researchers demonstrated four modeling dimensions as *goals*, *changes*, *mechanisms* and *effects* [13]. In adaptive CPS modeling, considering above aspects, both functional and non-functional goals will evolve and remain dynamic in a long-term duration and multiple goals may interfere with each other. The cause of CPS behavior changes can be both external and internal which accordingly are represented as the impacts from environment and the behavior changes of software itself. Changes can occur frequently, and some of them are unforeseeable. We should consider the autonomous adjustments and make it the best efforts to fulfill the system goals. Considering most of the adaptations are within short-term duration and event triggered, one applicable structure is centralized analysis and distributed adaptation. Some of the adaptations are safety-critical and others may be mission-critical, with non-deterministic predictability. The execution of adaptations should not affect the performance of system. One bad scenario is that the whole system is busy adapting and adjusting while the desired normal services are ignored or neglected.

Therefore, the effects of the adaptation should be light-weighted.

Comparing to the low-level implementation as programming, high-level model expression is much easier to be analyzed and manipulated. Modeling CPS through a top-down method from business processes to code generation following Model Driven Architecture (MDA) approach is applicable. Communications between components can be constructed based on Internet network or Ad-Hoc, ZigBee, Bluetooth, DSRC technologies which all of them are not the major focuses in this paper.

3 Architecture and Method

3.1 Design roadmap

In CPS, devices as entities can be used in different tasks and provide certain services. These services may have faults or failures that need a runtime adaptation. Unlike the implementation of the pure software system, the participation of hardware devices may change the whole layout and structure of the system. So we need a flexible and reusable approach to deal with the adaptation and reconfiguration.

At the design phase, as we apply the MDA approach to developing CPSs, the first step is requirement analysis. Then the basic model with all the functional and non-functional requirements is constructed according to the business processes. At this point we use Petri net as the modeling language to describe the basic model as core concern model. Then we aim at the possible runtime faults and failures in the CPS. Fault types are analyzed and respective adaptation strategies are constructed for faults recovery. These adaptation strategies are separated as aspects and described using Petri nets too. Based on the crosscutting concerns, pointcuts and the weaving rules, the aspects can be weaved into the basic model to form the final adaptive model. Respectively, the system model will be presented as a composition of basic nets and aspect nets as aspect-oriented Petri nets (AOPN)[14, 15]. At last we use theories of Petri nets to analyze the model and to make sure its consistence, correctness and compatibility. The design map is presented in Figure 1.

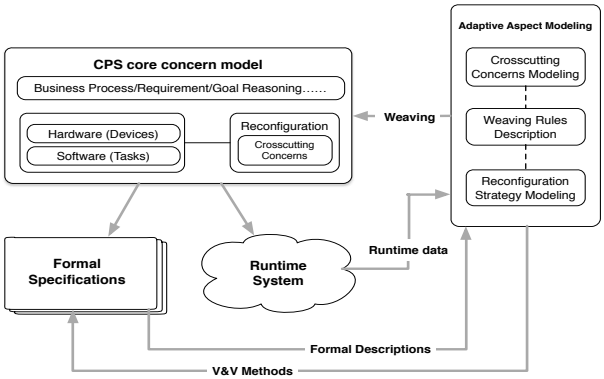

Figure 1. The design roadmap of the adaptive CPS modeling

3.2 The AOPN modeling approach description

The structure of the CPS contains the hardware as physical devices, software as tasks and the assigned relationships between them. Several definitions of the model and approach are given below.

Definition 1. (Requirement model) The requirement model of the CPS is $\Psi =< De, Tk, DC, RD >$:
(1) $De = (Sen, Act, Nod)$ is a finite set of devices includes sensors, actuators and network nodes and their backups.
(2) $Tk = (SubTk, R, N, P, Re)$ is a finite set of tasks. $SubTk$ is the finite set of sub tasks, R is the finite set of resources, N is the number of resources involved in certain task. P is the properties of the task. $\forall tk_i \in Tk$, $P(tk_i) = (In, Out, V, T)$ denotes the input, output and the executive speed and time constraint of the task. $Re = \{\succ, ||, +, \neq\}$ is the relationship between tasks as sequential, parallel, selective, and mutual exclusion.
(3) $DC : Tk \to De$ is the function that assigns the devices for each task.
(4) $RD : De \to rv$ is the threshold value of each device for reference.

Definition 2. (Composition model) The composition model is $AOPN =< N_b, N_a, IO, Pt, Ad, Ia, t_p, v, M_0 >$:
(1) $N_b = (P_b, T_b; F_b)$ is basic net denoted by Petri net structure.
$P_b = \{p_1, p_2, ..., p_n\}$ is a finite set of places.
$T_b = \{t_1, t_2, ..., t_n\}$is a finite set of transitions.
$P_b \cap T_b = \emptyset$ and $P_b \cup T_b \neq \emptyset$.
$F_b \subset (P_b \times T_b) \cup (T_b \times P_b)$ is a finite set of arcs.
(2) $N_a = (P_a, T_a; F_a)$ shares the same structure definition with N_b.
(3) $IO \subset P_a \cup T_a \cup F_a$ is the set of start points and the end points of the aspect net. The start point or the end point can be a place or a transition or an arc and they are the conjunction points between the basic net and the aspect net.
(4) $Pt \subset N_b$ is a finite set of pointcuts in the basic net N_b. It can be any elements in the basic net such as places, transitions or arcs.
(5) $Ad = \{before, after, around\}$ is the type of advice net.
(6) $Ia \subset P \times T \wedge Ia \cap F = \emptyset$ is the set of inhabit arcs. It is non-null if the advice is *around* type.
(7) $t_p \subset T$ represents an abstract transition of certain pointcuts.
(8) $v = v_i(\tau) \to (0, R^+)$ is the firing speed of certain transition t_i at time τ. If the transition is discrete then the speed is defined as 0 and if continuous then the speed is a real number defined forehead according to the requirement.
(9) M_0 is the initial marking of the model.

Denote $Pre, Post \in N^{|P|\times|T|}$ as the input and output matrix of the net. For $t_i \in T$, ${}^\bullet t_i = \{p_j \in P | Pre_{ji} \in F > 0\}$

is the pre place set of transition t_i, $t_i^\bullet = \{p_j \in P | Post_{ij} \in F > 0\}$ is the post place set of transition t_i. Similarly, there is $p_i \in P$, ${}^\bullet p_i = \{t_j \in T | Pre_{ji} \in F > 0\}$ as the pre transition set of place p_i and $p_i^\bullet = \{t_j \in T | Post_{ij} \in F > 0\}$ as the post transition set of place p_i.

Definition 3. (System state) Define the system state at time σ as $S = (M, TS)$.

Marking M denotes the resource distribution in system. TS is the waiting time of marking M at time σ. The initial state is $S_0 = (M_0, TS_0)$, TS_0 means all the tokens under initial state is available. Then we know that as the time elapses ω unit, the tokens in places will change and lead the system to a new state. Denote as $\sigma + \omega(\omega > 0)$, $S[\omega > S'$.

Assume at time σ, the model is under marking M. For $\forall p_i \in P$, there's j tokens in p_i, and $d_{ik}(0 < k < j)$ is the kth token. ct_k is the create time of d_{ik} and dt_i is the delay time of place p_i. Then $TS(p_i) = (TS_{i1}, TS_{i2}, ..., TS_{ij})$, where $TS(d_{ik}) = max\{dt_i - (\sigma - ct_i), 0\}$. We call $TS(d_{ik})$ as the waiting time interval, which means the system needs to wait this time unit to use the entity d_{ik}. If it is zero, means this entity is available. To diminish the complexity, here we only discuss the situation that in place p_i, $dt_i > 0$ and there exists only one type of entities. Define $TS(M, \sigma)$ as the set of waiting time under marking M at time σ. M^a and M^u are the available and unavailable entity distribution in marking M. $|M^a(p_i)| = k$ and $|M^u(p_i)| = k$ means there is k entities available and unavailable in place p_i.

Definition 4. (Enabled transition set) Let $ET(S)$ be the set of all the enabled transitions under state S. For transition $t_j \in T$, t_j is enabled under marking $m \in M$, if and only of $\forall p_i \in {}^\bullet t_j$, $|M^a(p_i)| = k > 0$. That means if t is enabled, p as the pre set of t, needs to have at least one token available.

Definition 5. (The biggest triggering transition set) Let $VT(S)$ be the set of all the valid triggering transitions under state S. Let $CT(S)$ be the biggest set of the concurrent transitions under state S. Then the biggest triggering transitions set under state S is defined as $H(S) = \{t_i | t_i \in CT(S) \cup VT(S)\}$.

For $S = (M, TS)$ as one state in system model, $\delta = 1/v$ is the time threshold value. For transitions $t_i, t_j \in ET(S)$, if $\delta_i \leq min(\delta_j)$ then we say that the triggering of transition t_i under state S is valid.

For $\forall t_i, t_j \in VT(S)$, if ${}^\bullet t_i \cap {}^\bullet t_j = \emptyset$, the transitions t_i, t_j under state S are concurrent. Otherwise they conflict. If the relationship between two transitions is concurrent, then the triggering of one transition will not affect the other one.

Definition 6. (Reachability) $S = (M, TS)$ is one state in system model. The system evolves to a new state S' through all the valid triggering of transitions in $H(S)$. Denote as $S[H(S) > S'$, we call S' is the reachable state of state S. For $\forall t_i \in H(S)$, $\forall p_i \in {}^\bullet t_j \cap t_j^\bullet$, the marking will evolve as $M'(p_i) = M(p_i) - F(p_i, t_j) + F(t_j, p_i)$. Define $H_1, H_2, ..., H_m$ and $S_1, S_2, ..., S_m$ as the trigger-able sequence and the state sequence of the system model, then $S[H_1(S) > S_1[H_2(S_1) > S_2 ... S_{m-1}[H_m(S_{m-1}) > S_m$, then we call the state S_m is reachable. The set of all the reachable markings from S is denoted as $R(S)$ and $S \in R(S)$. According to the triggering rules discussed above, we can construct the reachable state from the initial state S_0.

Definition 7. (Advice operations)
(1) *before*: $N_a \succ N_b$ represents that aspect net N_a will execute before the basic net N_b proceeded when the process flow come to the joint point. The computation of N_b will suspend and N_a must be completed before N_b can be continued.
(2) *after*: $N_a \prec N_b$ represents that aspect net N_a will execute after the basic net N_b proceeded when the process flow come to the joint point. The computation at the joint point in N_b is completed first then N_a can be started.
(3) *around*: $N_a \Xi N_b$ represents that aspect net N_a will execute instead of the basic net N_b when the process come to the joint point with the conditions satisfied. The computation of the joint point in N_b will no longer be executed in this case.

Graphical notations of AOPN have been elaborated in our former work and will not be discussed in this paper due to the limited space.

Definition 8. (Composition operation) Composed aspect-oriented Petri net model is $CN = (N_b, N_a, OP)$:
(1) Basic net $N_b = \{P_b, T_b; F_b, I_b, O_b, Pt\}$ shares the same definition with Petri net.
(2) Aspect net $N_a = \{P_a, T_a; F_a, I_a, O_a\}$ is mainly the definition of the Introduction net.
(3) Operations $OP = \{N_a \succ N_b, N_a \prec N_b, N_a \Xi N_b\}$.

Pt is the set of crosscutting concerns denoted by pointcuts of joint points in the basic net. It has three types such as *place-pointcut*, *transition-pointcut*, and *arc-pointcut*.

A composed net $CN = N' = N_b \Theta N_a$ is a combination of basic net N_b and aspect net N_a. An aspect A contains a set of pointcuts Pt, a set of advices denoted as $A.V$ and a set of introductions denoted as $A.I$.

Here we use a transition type of pointcut to discuss the formal syntax of three types advices. $t_{ai} \in I_a \subset I$ is the start point of the aspect net. Also means the entry point for an aspect. Here the start point is a transition in the introduction net of the aspect. Similarly, $t_{ao} \in O_a \subset O$ is the end point of the aspect net also means the exit point for an aspect.
(1) $N_a \succ N_b$
$N' = N_b \Theta N_a$, where
$P' = P_b \cup P_a$
$T' = T_b \cup T_a$
$F' = F_b \cup F_a \cup \{(p_a, t) | (t \in Pt \in T_b) \wedge (p_a \in {}^\bullet t_{ao})\}$
(2) $N_a \prec N_b$
$N' = N_b \Theta N_a$, where
$P' = P_b \cup P_a$

$T' = T_b \cup T_a$
$F' = F_b \cup F_a \cup \{(t, p_a) | (t \in Pt \in T_b) \wedge (p_a \in t_{ai}{}^{\bullet})\}$
(3) $N_a \Xi N_b$
$N' = N_b \Theta N_a$, where
$P' = P_b \cup P_a$
$T' = T_b \cup T_a - \{t | t \in Pt \in T_b\}$
$F' = F_b \cup F_a \cup \{(p_b, t_{ai}) | (p_b \in {}^{\bullet}t \in Pt) \wedge (t_{ai} \in T_a)\} \cup \{(t_{ao}, p_b{}') | (t_{ao} \in T_a) \wedge (p_b{}' \in t^{\bullet})\} - \{\{(p_b, t) | (p_b \in {}^{\bullet}t) \wedge (t \in Pt)\} \cup \{(t, p_b{}') | (p_b{}' \in t^{\bullet}) \wedge (t \in Pt)\}\}$

The description of the weaving mechanism that weaves the aspect of *transition-pointcut* with *around* advice named *AroundTransition* into the basic Petri net is described as following[16].

CN weaving AroundTransition (*BasicNet* N_b, *Aspect* A)
{
For $(\forall p_i \in {}^{\bullet}tcut) \wedge (p_i \in N)$
$\{N_b.A = N_b.A \cup \{(p_i, ({}^{\bullet}tcut)^{\bullet})\} \wedge (({}^{\bullet}tcut)^{\bullet} \in A.I)\}$;
For $(\forall p_j \in tcut^{\bullet}) \wedge (p_j \in N)$
$\{N_b.A = N_b.A \cup \{({}^{\bullet}(tcut^{\bullet}), p_j\} \wedge ({}^{\bullet}(tcut^{\bullet}) \in A.I)\}$;
$N_b.A = N_b.A - \{(p_i, tcut)\} - \{(tcut, p_j)\}$;
$N_b.Pt = N_b.Pt - \{tcut\}$;
Return N_b;
}

For the verification of the final adaptation model weaved with aspects, we use the composition verification approach referred to the method presented in [17]. The graphical formal software architecture description model (SAM) is a general framework based on two complementary formalisms: Petri nets and temporal logic. Petri nets are used to visualize and model both structure and behaviors. And temporal logic is used to specify the required properties of system. Each module in the software architecture can be regarded as a component. Here the component can either be the physical entity or the software entity as well as other elements in the system such as the adaptation aspects. And each component is defined using a Petri net to visualize its internal logical structure. Then the whole software architecture is defined as a hierarchical model supports compositionality in both software design and analysis.

Definition 9. (Verification of composition correctness)
The system can be defined by a set of compositions as $C = \{C_1, C_2, ..., C_k\}$. Each composition denoted as $C_i = \{C_{ij}, Cs_i\}$ corresponds to a design level or the concept of sub-architecture and consists of a set of components C_{ij} and a set of composition constrains Cs_i. Each component $C_{ij} = (S_{ij}, B_{ij})$ in a composition C_i has a temporal logic formula specified property specification S_{ij} and a Petri net described behavior model B_{ij}. A composition constraint is defined as a property specification often contains the connections of multiple components. Then the correctness of the system specification is defined in two levels.

First is the component correctness. The property specification S_{ij} holds in the corresponding execution sequences of the behavior model B_{ij} as $B_{ij}| = S_{ij}$.

Second is the composition correctness. The conjunction of all constraints in Cs_i of C_i is implied by the conjunction of all the property specifications S_{ij} of C_{ij} as $\wedge S_{ij}| - \wedge Cs_i$. Then we can say the conjunction of all constraints in Cs_i holds in the integrated behavior model B_i of composition C_i as $B_i| = \wedge Cs_i$.

4 The Adaptation Model

From the perspective of runtime adaptation, we divide the runtime adaptive requirements into the hardware facet and software facet. The hardware facet is mainly about the physical devices. The software facet is mainly about the tasks. First of all we present the core concern models of devices and tasks. Then the runtime inspection model for fault discovery is given. After the analysis of different types of faults, the respective adaptation models are proposed with adaptation strategies.

4.1 The core concern model

The basic model of the CPS is constructed as the device models. Device model described in Petri net is shown in Figure 2. If there is an executive request then the device will be invoked and initiated to a standby state, which is denoted as a token in place p_r of the model. According to different roles of the devices, the request can be collecting data or executing for services or adjusting to normal state. The place p_e will store the flag to keep the information that the service execution or data collecting procedure is finished.

For the modeling of tasks, we can abstract the task into three states: standby, execution, and output. Then according to the defined relationships as sequential, parallel, selective and mutual exclusion between tasks, the tasks will schematize the device resources and connect them for scheduling. The composed task model CM_i will not be graphically elaborated in this paper.

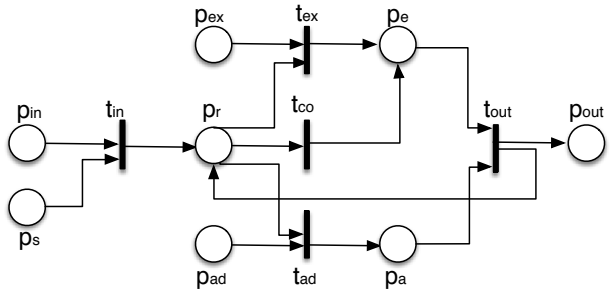

Figure 2. The Petri net model of device De_iM

4.2 System inspection and adaptation

To support the runtime adaptation, we introduce the reflection mechanism into the system design. The self-representation capability of the reflection can make the system correspondently response and react to its own problem using two steps: inspection and adaptation. Through weaving an inspection aspect module into the system, the systems runtime surveillance can be achieved. If adaptation needed, the well-designed strategic adaptation aspects can be weaved into the system model to adjust or replace certain behaviors or characteristic, then bring the system back to normal.

Here we consider two situations. One is the value deviation of devices. Runtime data of these devices will be compared with the defined reference values, if the deviation is beyond the threshold, the fault will be outputted. This situation is most likely happened to sensors. The other situation is the unavailability of device. This situation is easier to deal with because lack of the requested device will lead the task to suspend and throw out an exception. The model is shown in Figure 3.

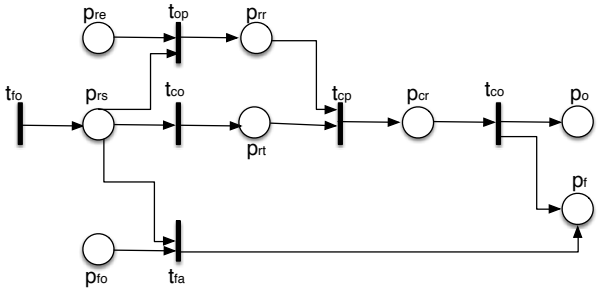

Figure 3. The modeling of runtime inspection and adaptation

In this model we define two crosscutting concerns pointcuts t_{fo} and p_{fo}. Pointcut $t_{fo} = \{De_iM.t_{in}\}$ is a transition that leads the model to a process of checking the error deviation of the device. And $p_{fo} = \{De_iM.p_r\} \cup \{CM_i.p\}$ will trigger an exception when the device has error or is unavailable or the task is suspended. When the place p_f has token, it requires adaptation mechanism to resolve the fault. We need to weave the respective adaptation strategy into the core concern model. The adaptation aspects will be discussed in the following sections.

4.3 Device adaptation

According to the situations discussed in Section 4.2, we need to consider two scenarios for device reconfiguration respectively. Targeting at the abnormal execution of the device, we can adjust the properties of the device as its volt or parameters to make sure it back to track. Here we use PID controller with feedback loop theory to adjust the device. If the device is not available, we need to reboot it. If the device has finished its service already and is currently not occupied by any task, it needs to hibernate so as to save energy. The aspect net is mainly about the introduction net of aspect as shown in Figure 4. p_{dw} will store the state information of devices.

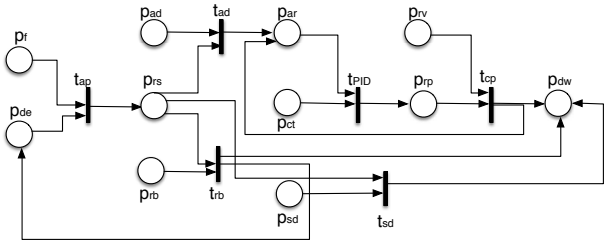

Figure 4. The modeling of device fault adaptation

4.4 Task adaptation

In our proposed CPS model, we assume that most tasks as software procedures are well designed and will barely have internal breakdown. We consider the situation of task breakdown that mainly triggered by the fault of devices or the unavailability of devices. We apply the re-scheduling scheme for re-planning the resources that required by tasks, to make sure the task continue fulfilling the ultimate target. When the task is initiated, system starts to execute and send the invoke requests to all the required devices. At runtime, if some of the devices are malfunctioned and start the device reconfiguration processes as self-adaptation, there is no need to halt the task to wait for the faulty device back to normal. And we don't know when the device will be reconfigured or the effect of the device reconfiguration such as successfully recovered or not. The implementation and deployment of system like CPS must have redundant devices in various types. Those idle devices can become the backups of malfunctioned devices. So rescheduling is a feasible approach to deal with the task breakdown. The model is shown in Figure 5.

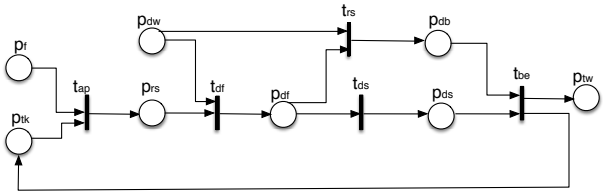

Figure 5. The modeling of task adaptation

Resources in CPSs are unlike the resources in Web and Cloud, which the latter two are mainly about virtual resources. That means one resource can be shared by different tasks at the same time. Here in CPSs, resources

are mostly devices and they provide certain services in one task. So once the task is initiated and resources are planned, these resources will be occupied till the task is terminated. So one resource can only be invoked in one task at the same time. Here the rescheduling in our paper is not a start-over re-planning of the resources scheduling. We keep a flag at the breakdown point of the runtime state. All the results before the flag are reserved. There is one situation we must consider. Before the task initiate, all resources are available for scheduling and we can make the best effort for the resource planning. But at runtime, there may exist multiple tasks and some of the resources are already occupied. How to find the most applicable backup device for the rescheduling is our main concern. We propose a PSO (Particle Swarm Optimization) based Pareto optimization algorithm to find the best solution.

In CPS, services of physical devices are usually location-aware, energy-aware and time-critical. We consider these criteria in the evaluation process of finding the best solution. They are the complexity of the reschedule path, the energy cost of the rescheduled device, the time delay of the rescheduled device and the service utility the device can provide.

$$B = \sum_{i=1}^{m} p_i * l_i \tag{1}$$

$$E = E_{initial} + E_{path} + E_{execution} \tag{2}$$

$$T = t_{response} + t_{delay} \tag{3}$$

$$Q = E^{-e} \tag{4}$$

The complexity of the reschedule path is calculated as Formula (1). Assume there exist m possible paths in the system to reschedule the device. p_i is the possibility of path i be chosen. And l_i is the length of the path i.

The energy cost of the rescheduled device is calculated using Formula (2). If the device will be rescheduled in the task, then it should start a process of initiation. The cost of this process is pre-defined in the device manual. $E_{path} = \omega_i * l_i$, in which ω_i is the cost of unit reschedule path, it's related to the interlink implementation. $E_{execution} = \mu * t$, μ is the execution cost of the device in unit time, and t is the execution time defined by the task.

The time delay is calculated as Formula (3), in which $t_{response}$ is the time from the task sending out the reschedule requests to the task receiving the response from the device. t_{delay} is the waiting time of the device to participate in the task. For idle device, the waiting time is zero. But if there is no idle device in the system, we need to consider those devices already occupied in other task.

The service utility is the performance one device can provide in the task execution that can be calculated using Formula (4). We define it as related to the energy consumption of the device.

To measure a device, the B, E, T is the lower the better and the service utility Q is the bigger the better. So we need a tradeoff between the multi-objectives.

We define the path complexity B, energy cost E, time delay T and the service utility Q as the four targets in PSO as f_1, f_2 , f_3 , f_4 . In PSO [18], the search space iteration formula are:

$$V_{id} = W * V_{id} + C_1 * \xi * (P_{gd} - X_{id}) + C_2 * \eta * (P_{id} - X_{id}) \tag{5}$$

$$X_{id} = X_{id} + rV_{id} \tag{6}$$

In Formulas (5) and (6), V_{id} and X_{id} are the speed and position of particle i. P_{id} and P_{gd} are best position of particle i as in an individual vision (p_best) and best position of all particles as in a global vision (g_best). W is the inertia weight, which make the particle capable to explore the whole and new search space. C_1 and C_2 are acceleration constants. ξ and η is a random value in $[0,1]$. r is a constrain factor usually set to 1.

In the iteration of the PSO algorithm, the flying of particle is determined by both the individual local best solution and the global best solution. We need two evaluation mechanisms to achieve the optimism of multiple targets. The fitness evaluation function of local best solution is presented in Formula (7). n is the number of targets which should be 4 in our case. $\alpha_i \in [0,1]$ is random generated weight factor of each target and $\sum_{i=1}^{n} \alpha_i = 1$.

$$fit(s) = \sum_{i=1}^{n} \alpha_i * f_i \tag{7}$$

There may exist tradeoffs between the four targets. So the traditional multiple objectives weighting methods is not applicable. We propose the Pareto optimization method. In order to compare the pros and cons of the solution, we use the dominated concept [19]. For a minimization problem of n objectives, if solution x_1 dominates x_2 then all targets value of x_1 should not be bigger than x_2 and at least one objective value of x_1 should be smaller than x_2. So we have Formula (8), (9):

$$\forall i \in (1,2,...,n), f_i(x_1) \leq f_i(x_2) \tag{8}$$

$$\exists i \in (1,2,...,n), f_i(x_1) < f_i(x_2) \tag{9}$$

All mutual non-dominated solutions will form the set of Pareto best solutions. Here we propose the PSO based Pareto optimization algorithm as in Algorithm 1.

The Pareto optimization algorithm may output a set of best solutions. So we randomly chose one as the alternative device for the faulty one in the task. The complexity of PSO is lower than genetic algorithm and Ant Colony Optimization [20]. Its convergence speed is quick so it can find the best solution in a relative short time. And Pareto can solve the multiple objectives tradeoff problem. The efficiency of the two methods combined is applicable for our light-weighted reschedule process.

Algorithm 1 PSO-Pareto optimization solution finding algorithm

Require: *population size*, *max_generation*, W, C_1, C_2, α_i, number of target n, and other needed parameters
Ensure: The best solution set
  for each *particle* **do**
    initialize *particle*;
  end for
  for *Pareto Set* **do**
    initialize *Pareto Set*;
    for $i = (0,...,n)$ **do**
      calculate f_i;
    end for
    Pareto Set == non-dominated (f_i);
    Set *generation* = 1;
    while (*generation* < *max_generation*) **do**
      generation = *generation* + 1;
      calculate *next generation particles*;
      evaluate *new particles*;
      update *p_best*;
      update *g_best*;
      update *Pareto Set*;
    end while
  end for
  return *Pareto Set*

5 Model Analysis

Temporal formulas are built from elementary formulas (predicates and transitions) using logical connectives $\neg$, $\wedge$, and derived logical connectives $\vee$, $\rightarrow$, $\leftrightarrow$, universal quantifier $\forall$ and derived existential quantifier $\exists$, and temporal operators always $\Box$, sometimes $\Diamond$, next time X and until U.

The semantics of temporal logic is defined on behaviors (infinite sequences of states). The behaviors are obtained from the execution sequences of Petri nets where the last marking of a finite execution sequence is repeated infinitely many times at the end of the execution sequence [17]. Models are the description of the system. So the correct execution of business process is the first priority we need to concern. Here we analyze and demonstrate that our adaptive model with aspect-oriented method integrated can correctly reflect the executive process of the system.

Define the composed model as CM_i, the device model of device De_iM as De_i.

Theorem 1. Ψ denotes the adaptive CPS model. $R(\Psi)$ is the set of reachable state, $\forall S \in R(\Psi)$, $\forall De_f, De_g \in De$, $\forall tk_i \in Tk(De_f)$ and $\forall tk_j \in Tk(De_g)$:
(1) If $Re(tk_i, tk_j) = \succ$, then:
$\Box(Execution(S \wedge tk_i) \rightarrow \Diamond Execution(S' \wedge tk_j))$;
(2) If $Re(tk_i, tk_j) = ||$, then:
$\Box((S_0 \wedge (Execution(tk_i) \vee Execution(tk_j))) \rightarrow \Diamond(|M(De_f.p_e)| + |M(De_g.p_e)| \leq 1))$;
(3) If $Re(tk_i, tk_j) = +$, then:
$\Box((S_0 \wedge (Execution(tk_i) \vee Execution(tk_j))) \rightarrow \Diamond(|M(De_f.p_e)| = |M(De_g.p_e)| = 1 \vee |M(De_f.p_e)| = |M(De_g.p_e)| = 0))$.
Proof:
(1) As the two tasks execute in a sequence, if $f = g$, then tk_i, tk_j will be executed by the same device. From the model we present above, there exist the transition $De_f.t_{i,j}$ that will transfer the execution result back to the input of next task. So if the task tk_i is is enable to execute under the state S then there exist the state $S' \in R(S)$, $De_f.t_{i,j} \in ET(S)$. Then tk_j is enabled to execute. So proposition (1) proved.
(2) Assume the proposed proposition (2) is not established. That means: $\exists S \in R(S_0)$, that makes $|M(De_f.p_e)| + |M(De_g.p_e)| > 1$. In our device model there is a place p_e keeping the marks, which denote the termination of executions. So $|M(De_f.p_e)| \leq 1$ and $|M(De_g.p_e)| \leq 1$, and $|M(De_f.p_e)| = |M(De_g.p_e)|$. Because the executive order is parallel, so we know that under some state there is no possibility that two tasks execute at the same time. So the assumption is not established. Then the proposition (2) is true.
(3) This proposition can be proved the same way as (2). So the proposition (3) also is true. $\Box$

Theorem 1 proves our model can correctly describe the execution of business process of the system.

Theorem 2. Ψ denotes the adaptive CPS model. $\forall M \in R(M^\sigma)$, $\lambda(M^\sigma, M)$ is the triggering transition sequence from state M^σ to M. $M^t(\Psi)$ is the set of normal termination markings. $\forall M \in R(M_0)$, if $\exists CM_i \in CM$ makes $M(CM_i.p_s) \neq \emptyset$, then $\forall M' \in M^t(\Psi) \cup R(M)$ and $\forall \lambda^k \in \lambda(M, M')$, $CM_i.t_e \in \lambda^k$ is established.
Proof:
Because $M(CM_i.p_s) \neq \emptyset$ and ${}^\bullet CM_i.t_{in} \in CM_i.p_s$, so we have $CM_i.t_{in} \in VT(M)$. Because $CM_i.p_s^\bullet = CM_i.t_{in}$, so we know that $CM_i.t_{in} \in H(M)$. For any set like H_1, $M[H_1 > M_1$, when the model is initiated and start to execute tasks. According to the executive processes the model will be executing until all the tasks are finished. Then $CM_i.t_e$ is enabled. Because $M' \in M^t(\Psi)$ so $VT(M') = \emptyset$. That means $\forall \lambda^k \in \lambda(M, M')$, $CM_i.t_e \in \lambda^k$ is established. $\Box$

Theorem 2 proves that the model can sense the dynamic changes of the runtime system. And through the setting of triggering condition for the transitions, the system can be executed dynamically.

Theorem 3. Let ψ be the adaptive CPS model. Con_{De} is the crosscutting concern about devices. Assume Cut_1 and Cut_2 are two pointcuts. $\forall Cut_1, Cut_2 \in Con_{De}$, for $\forall De_i$, weave the aspect into the device model at two pointcuts sequential order get $De_i^{1,2}$ and $De_i^{2,1}$. If $M_1(De_i.p_{in}) = \varepsilon$, then $R(M_1, De_i^{1,2}) = R(M_1, De_i^{2,1})$.
Proof:
Because the different weaving order won't change the structure of the composed model, so models $De_i^{1,2}$ and $De_i^{2,1}$ will have the same places, transitions, arcs and other properties. Because $M_1(De_i.p_{in}) = \varepsilon$, so we know

that $M_1(De_i^{1,2}.p_{in}) = \varepsilon$. Because ${}^{\bullet}De_i^{1,2}.t_{in} = De_i^1.p_{in}$ and $De_i^1.p_{in}^{\bullet} = De_i^{1,2}.t_{in}$, so we have $De_i^{1,2}.t_{in} \in H(M_1)$. This means under the marking M_1 , the biggest set of triggering transitions includes the transition $De_i^{1,2}.t_{in}$. For any H set like H_1 and $M_1[H_1 > M_2$, if the situation like the error deviation of collected data value and reference value of a sensor is bigger than the threshold, or the situation of device unavailable, the respective transition t_d will be triggered to output the fault. If the device is terminated successfully then the respective transition t_o will be triggered and output the result. □

Theorem 3 proves that the weaving operations will keep the model consistent and stable. All the pointcuts of crosscutting concerns are compatible so the operation of adaptive aspect weaving into the core concern model is feasible.

Theorem 4. Let ψ be the adaptive CPS model. M_0 is the initial marking of the model.
(1) $\forall M \in TS(\Psi)$ and $\forall \lambda^k \in \lambda(M_0, M)$, then $CM_i.t_{fa} \notin \lambda^k$;
(2) $\forall M \in R(M_0)$ and $\forall \lambda^k \in \lambda(M_0, M)$, if $\exists De_i.t_{fa} \in \lambda^k$ and $\exists M' \in TS(\Psi)$, then $M' \in R(M)$.

Proof:
Assume the proposition (1) is not established. Then $CM_i.t_{fa} \in \lambda^k$. For $M_0[S_0 > M_1[S_1...M$, there exist $M_i(CM_i.p_{fa}) \neq \emptyset$. So $M_i(CM_i.p_o) = \emptyset$. According to Theorem 2, we can have $\forall M' \in R(M_i)$, $M'(CM_i.p_o) = \emptyset$. Then we have $M'(p_e) = \emptyset$, which will lead to $\forall M' \in R(M_i)$, $M' \notin TS(\Psi)$. Because $M \in R(M_i)$, so $M \notin TS(\Psi)$ conflicts with the condition. So the assumption is not established. Then proposition (1) is proved. We also can prove the proposition (2) as the same way. □

Theorem 4 proves that if the model terminates normally at some position, the system can finish all the requirements by executing tasks. When there is fault happened to some device the system can keep executing until to the termination.

6 Case Study

6.1 The case study description

In this section we use a case study to demonstrate how our adaptation approach works. Assume a simple case includes three tasks. Task 1 is the smoke-detecting task that its workflow includes several location-fixed smoke sensors' cooperation. Task 2 is the temperature-sampling task, which involves several location-fixed temperature sensors coordinately working as a workflow too. All sensors keep sampling the data in the environment. The sampled data of temperature and smoke are both taken into consideration by an analysis module. If the analyzed data show that the environmental parameters are out of normal range, then a sprinkling operation is needed in order to improve the current environment situation. After the decision is made, the sprinkling Task 3 is invoked to execute. Sprinklers are movable in a certain range while the sensors in Task 1 and 2 are all fixed installed. The scheduling program will invoke the most appropriate sprinkler to perform the task. The model of the combined tasks is shown in Figure 6. Tasks consist of components by the basic relationships as sequence, parallel, loop, choice according to the business requirement. Components are the physical devices as the smoke sensors, the temperature sensors and the sparkling actuators and other devices. Nonempty place p_{ri} will lead to the process of the runtime inspection that may detect and export fault in the task procedure. And token in place p_{fa} will call for the adaptation procedures.

To demonstrate, we assume there is a fault happened to one temperature sensor in Task 2. And the fault is an abnormal execution of the temperature that causes the sensor cannot output the normal data. Then inspection module will detect the fault and the place p_{fa} will has a token. This model state will call for the adaptation procedure. The device and task adaptation model are separately weaved into the task model as shown in Figure 7. For the device adaptation, with the reference threshold value compared, the strategy is to adjust the core parameters of the temperature sensor using a control loop controlling method to bring it back to normal. The strategy of the task adaptation is to do a PSO-Pareto optimization for rescheduling the backup temperature sensors to replace of the faulty one. This reschedule computing process is represented as a transition in the task adaptation. The two adaptation strategies are performed separately and simultaneously. If the adjustment of the device finished earlier, then the temperature sensor will continuously providing service in the task after its recovery. And if the task adaptation finished earlier, that means before the current temperature sensor finishing adjustment and back to normal, the adaptation has already find the best alternative backup temperature sensor for the task. Then the chosen backup sensor will replace the faulty one. The backup temperature sensor will be weaved into the task model using an *around* advice approach at the input and output of the faulted sensor in Task 2 to replace the execution of the current temperature sensor. Due to space limitation, the graphic weaving demonstration of the faulted sensor replacing will not be included.

6.2 Case analysis

Here we discuss the correctness of the composition model after using weaving approach.

The weaving of two adaptation strategies for the faulty temperature sensor can be refined to a sequential order. As elaborated in Theorem 3, the different weaving order won't change the structure of the composed model. As the situation is the error deviation of the collected data value and reference data value of a sensor, the runtime inspection aspect will output the fault to p_{fa}. Theorem 3 proves that the weaving operations will keep the model consistent and stable, and the operation of adaptive

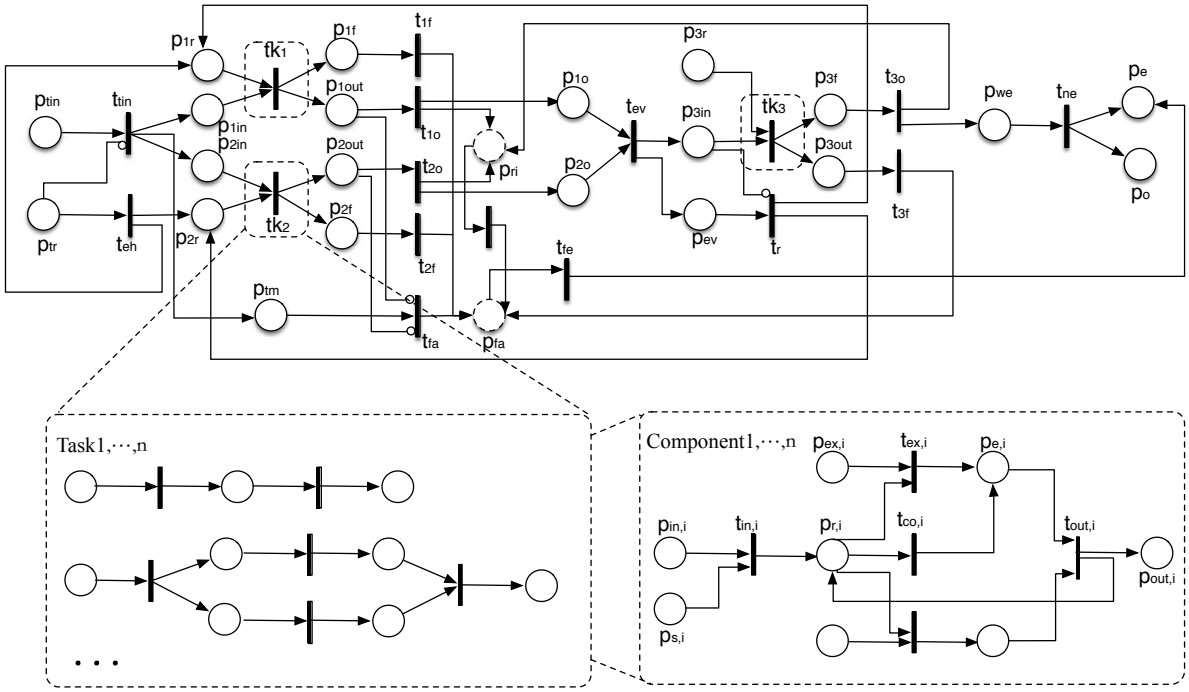

Figure 6. A three-task case model of CPS

aspects weaving into the core concern model is feasible. And the token in p_{fa} will invoke the adaptation procedure. Theorem 4 proves that when there is fault happened to some device the system can keep executing until to the termination. The adaptation will recover or replace the faulty device, and lead the execution back to the core model till the termination p_e.

From the hierarchical composition point of view, we use the SAM theory to prove the correctness of the composition. The system can be defined by a set of compositions here in the case referred as the tasks including the core tasks and the adaptation tasks. Each task or adaptation consists of a set of components and its relationships constraints. In the case these components are devices such as sensors and sparkling actuators. Each component can be described using a temporal logic formula and a Petri net. Then as in this case, the architecture property specifications include:

(1) All tasks will be eventually terminated.

$\Box(Task(i) \rightarrow \Diamond Terminate(Task(i)))$

(2) If Task 2 failed, then the adaptation procedure of sensor recovery and task recovery must be invoked and executed until Task 2 is normally terminated.

$\Box(Fault(Task(2)) \wedge Adaptation(k) \wedge (k = sensor \vee task) \rightarrow \Diamond(Resultout(z) \wedge Terminate(Task(2))))$

(3) If Task 2 failed and start the adaptation procedure, its sequential task as Task 3 won't be initiated until Task 2 is terminated.

$\Box(Fault(Task(2)) \wedge \exists Adaptation(Task(2)) \rightarrow \Diamond(\neg Initiate(Task(3))UTerminate(Task(2))))$

According to the net of runtime fault inspection shown in Figure 7, the corresponding property specification is:

$\Box(Invoke(Task(inspection)) \wedge DataIn(x) \wedge (x = abnormality \vee error) \wedge DataAnalysis(x) \rightarrow \Diamond Resultout(y))$

In the case situation when the sprinkling task is needed, for all the movable sprinklers in the sprinkling range, there should be at least one available for Task 3 to invoke. That means not all of them are currently occupied by other tasks. The corresponding property specification is:

$\Box(Invoke(Task(3)) \wedge \exists SprinklersInRange(i) \rightarrow \Diamond(\exists AvailableSprinklers(i)UInitiate(Task(3))))$

Similarly, more detailed analysis such as processes inside the Task 1 and 2 and adaptations can also be specified using the same approach. Then the conjunction of all constraints can be implied by the conjunction of all the property specifications. Then we can say the conjunction of all constraints holds in the integrated behavior model of composition. Furthermore, symbolic model checking tool can be used to verify these specifications.

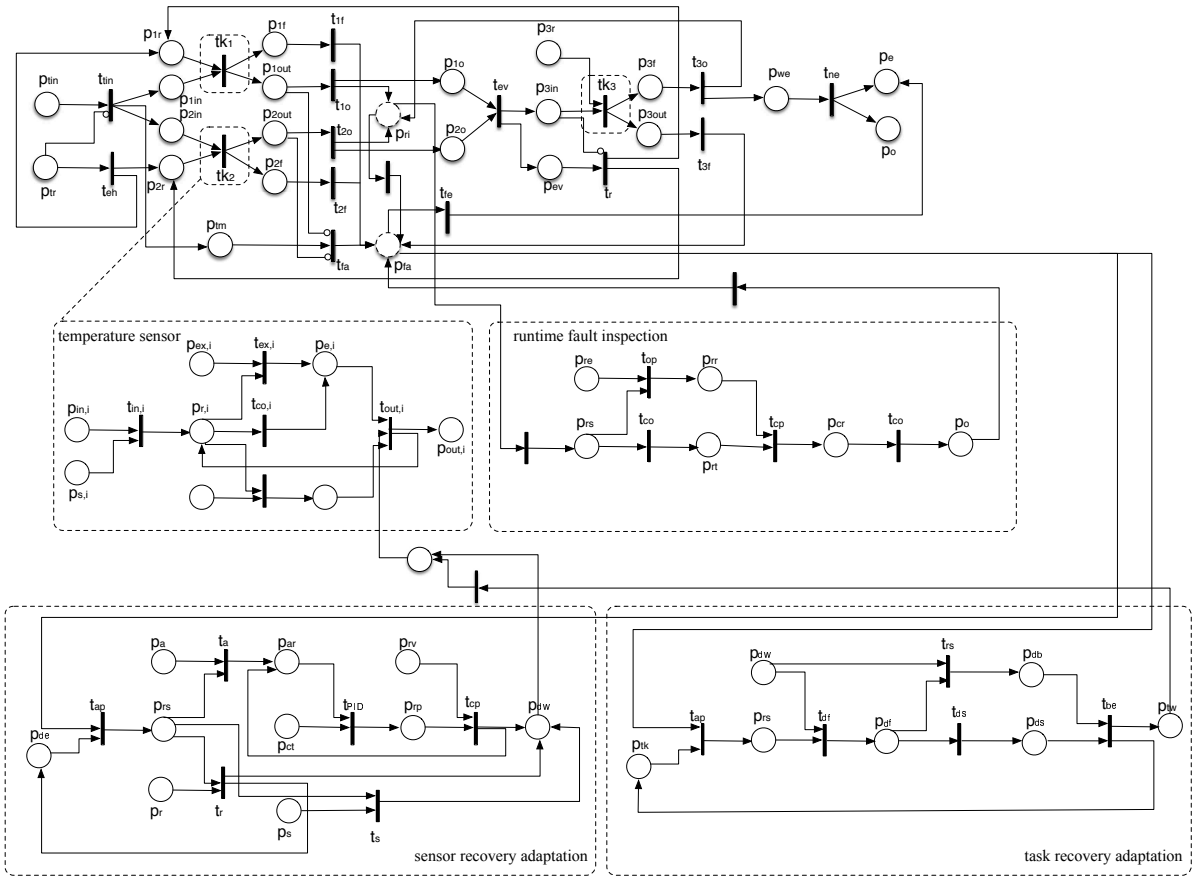

Figure 7. The composition model with fault adaptation aspect model weaved

7 Conclusion

Cyber physical systems are applied in critical areas such as emergency response, health care and Smart Grid etc. These systems desire the dependable and robust execution during the runtime. Applicable runtime adaptation design of the CPS is necessary and essential. From the prospective of modeling, the approach should be universal and simple. From the prospective of reengineering, the approach should applicable for the existing models and better not damage their structures. From the prospective of system design, the adaptation should be light-weighted and reusable and efficient.

We propose an aspect-oriented Petri net approach to modeling adaptive cyber physical systems. The CPS model is composed by core concern model and aspect model. The core concern model is described as device model and task model using Petri nets. The latter represented as the workflow of certain business processes execution with device model composed. The runtime system behavior changes and environment changes are monitored and fault information is outputted. Respective fault types are separated as crosscutting concerns. The models of fault inspection as well as device adaptation strategy and task adaptation strategy are designed as aspects using Petri nets. Then these aspect nets can be weaved into the core concern nets when the adaptation needed at the pointcuts using weaving rules.

For the device adaptation strategy, two fault types are analyzed and the control loop concept is integrated. For the task adaptation, we apply a rescheduling method with PSO-Pareto algorithm to find the best solution of the backup devices. This algorithm is capable for analysis the trade-offs between the proposed four criteria of devices and within short time and small cost, it can find out the most applicable backup device for the suspended task to continue.

Some characteristics of the model such as correctness are analyzed by validation and verification methods. As flexible and efficient and reusable, this approach can make model resiliently composable and expandable for ultra-large system.

8 Acknowledgement

This work was partially supported by the NSF of China under grants No. 61173048 and No. 61300041,

Specialized Research Fund for the Doctoral Program of Higher Education under grant No. 20130074110015, and the Fundamental Research Funds for the Central Universities under Grant No.WH1314038.

References

[1] E. Lee, Cyber Physical Systems: Design Challenges. Proceedings of 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing, pp. 363-369, 2008.
[2] C. Girault and R. Valk, Petri nets for System Engineering: A Guide to Modeling. Verification, and Applications. Berlin: Springer-Verlag, 2003..
[3] G. Kiczales, Aspect Oriented Programming. Proceedings of the European Conference on Object-Oriented Programming, Heidelberg: Springer-Verlag, 1997.
[4] R. Alur, T. A. Henzinger, P. Ho, Automatic Symbolic Verification of Embedded Systems. Proceedings of Real-Time Systems Symposium, pp.2-11, 1993.
[5] B. Cheng, R. de Lemos, H. Giese, P. Inverardi and J. Magee, Software Engineering for Self-Adaptive Systems: A Research Roadmap. Lecture Notes in Computer Science: Software Engineering for Self-Adaptive Systems, vol.5525, pp. 1-26, 2009.
[6] J. Kramer, J. Magee, A rigorous architectural approach to adaptive software engineering. Journal of Computer Science and Technology, vol.24, no.10, pp. 183-188, 2009.
[7] F. Dalpiaz, P. Giorgini and J. Mylopoulos, Software self-reconfiguration: a BDI-based approach. Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems, Vol. 2, pp. 1159-1160, 2009.
[8] M. Salifu, Y. Yu, B. Nuseibeh, Specifying monitoring and switching problems in context. Proceedings of the 15th IEEE International Requirements Engineering Conference, pp. 211-220, 2007.
[9] L. Chung, Dealing with security requirements during the development of information system. Proceedings of Advanced Information Systems Engineering, pp. 234-251, 1993.
[10] Y. Zhang, I. Yen, F. Bastani, A. Tai and S. Chau, Optimal Adaptive System Health Monitoring and Diagnosis for Resource Constrained Cyber-Physical Systems. Proceedings of the 20th International Symposium on Software Reliability Engineering, pp. 51-60, 2009.
[11] L. Phan, I. Lee, Towards a Compositional Multi-modal Framework for Adaptive Cyber-physical Systems. Proceedings of the IEEE 17th International Conference on Embedded and Real-Time Computing Systems and Applications, vol.2, pp. 67-73, 2011.
[12] K. Zhang, J. Li, A. Fortelle and X. Zhou, Agent Based Adaptive Cooperative Models and Mechanisms of Multiple Autonomous Cyber-Physical Systems. Proceedings of the 14th ACIS International conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, pp. 159-164, 2013.
[13] J. Andersson, R. de Lemos, S. Malek and D. Weyns, Modeling Dimensions of Self-Adaptive Software Systems. Lecture Notes in Computer Science: Software Engineering for Self-Adaptive Systems, vol.5525, pp. 27-47, 2009.
[14] D. Xu and K. Nygard, Threat-driven modeling and verification of secure software using aspect-oriented Petri nets. IEEE Transactions on Software Engineering, vol.32, no.4, pp. 265-278, 2006.
[15] D. Xu, X. D. He and Y. Deng, Compositional schedulability analysis of real-time systems using time Petri nets. IEEE Transactions on software engineering, vol.28, no.10, pp. 984-996, 2002.
[16] N. Yang, H. Yu, H. Sun and Z. Qian, Modeling Activity Diagrams with Extended Petri Nets. Intelligent Automation and Soft Computing, vol.17, no.5, pp. 559-569, 2011.
[17] X. He, H. Yu, T. Shi, J. Ding and Y. Deng, Formally analyzing software architectural specifications using SAM. The Journal of System and Software, vol.71, pp.11-29, 2004.
[18] J. Kennedy and R. Eberhart, Particle Swarm Optimization. Proceedings of IEEE International Conference of Neural Networks, vol.4, pp. 1942-1948, 1995.
[19] K. Deb, Evolutionary algorithms for multi criterion optimization in engineering design. Proceedings of Evolutionary Algorithm in Engineering and Computer Science, pp. 135-161, 1999.
[20] R. Eberhart and Y. Shi, Comparison between genetic algorithms and particle swarm optimization. Lecture Notes in Computer Science, vol. 1447, pp. 611-616, 1998.

Zhilin Qian is a Ph.D. candidate in Computer Application Technology at East China University of Science and Technology. Her research interests include software modeling, formal methods and Cyber physical systems.

Huiqun Yu is a professor in the Department of Computer Science and Engineering at East China University of Science and Technology. He is a senior member of the IEEE and CCF, and a member of the ACM. His research interests include software engineering, information security, formal methods, trustworthy computing and cloud computing.

Guisheng Fan is an associate professor in the Department of Computer Science and Engineering at East China University of Science and Technology. His research interests include software engineering, formal methods and cloud computing.