Modeling and Optimizing CPS Software Testing based on Petri Nets

Liqiong Chen a,b, Guisheng Fan c,* Huiqun Yu c

aDepartment of Computer Science and Information Engineering, Shanghai Institute of Technology, Shanghai 200235, China
bShanghai Key Laboratory of Computer Software Evaluating and Testing, Shanghai 201112, China
cDepartment of Computer Science and Engineering East China University of Science and Technology, Shanghai 200237, China

Abstract

Software testing is an important means to ensure the quality of software. However, there is a lack of effective modeling and optimization of CPS software testing. In this paper, Petri nets are used to model the underlying devices, components, connectors and test cases of CPS software. Aspect oriented programming extracts the crosscutting concerns of CPS software testing. The behaviors and their relationships are described based on AOP, and a weaving mechanism is used to dynamically integrate these models into the test model of CPS. The correctness of the model is analyzed by using the operation semantics and related theories of Petri nets. Based on the state space of constructed model, a strategy is proposed to dynamically select the test suite. The experiment results show that this method can effectively describe the CPS software testing process, which can improve the quality of software testing.

Keywords: Cyber Physical Systems; petri nets; testing; aspect oriented programming; verification

(Submitted on September 29, 2017; Revised on November 12, 2017; Accepted on November 23, 2017)

© 2017 Totem Publisher, Inc. All rights reserved.

1. Introduction

Cyber-Physical Systems (CPS) involves communication, computation and actuating through heterogeneous and widely distributed physical devices and computational components [15]. The economic and societal potential of CPS is believed to be tremendously greater than what has been achieved by existing systems in terms of autonomy, flexibility, and versatility [5]. In CPS, it is usually required to test all the scenes for completely covering various possible execution paths, thus ensuring that the system can meet the actual requirements. Therefore, the research of CPS software testing method is of great significance to ensure the normal operation of the system [4].

Existing modeling and optimization techniques lack the support to describe the important features of CPS, thus they cannot get the test cases to satisfy the actual requirements. However, a CPS includes hardware, software, and a physical environment, which makes the problem of software testing more challenging than conventional client-server systems.

(1) CPS run in an open environment. The state of physical devices is not completely predictable, and the operating environment is also not completely under control. Therefore, it must have a modeling language that can be used to accurately describe the testing specification of CPS.

(2) A unifying method for modeling CPS is desired. The problem lies in the absence of any comprehensive tool to do so. What is missing is a semantically common method of relating Cyber, network, and physical actions. Therefore, how to select and design the appropriate model to describe the various elements of CPS software testing process is an important problem.

(3) The scale and complexity of CPS. After constructing the models of various elements of CPS software testing, it is necessary to study how to effectively optimize the test cases and fully cover various possible executions, thus reducing the test overhead and improve the test efficiency.
(4) CPS has a complex structure, such as a discrete and continuous component. Traditional methods cannot effectively analyze the related properties. It is necessary to propose a method to verify the related properties of CPS.

Aspect-oriented programming (AOP) [10] can be used to abstract the various elements of business logic, thus reducing the coupling between the various elements and improving the reusability of the program, while improving the efficiency of development. The significance of aspect oriented design (AOD) at more abstract level has come to front recently [3], which can be done within different modeling notations, at different levels of abstraction, and at different moments during the software development process. However, the requirements of aspect-oriented analysis often appear ambiguous and vague. Therefore, formal methods can be used to analyze AOP, thus increasing the semantic constraints. Petri net is a mathematically based technique for modeling and verifying software artifacts [8].

In this paper, Petri nets based approach to modeling and optimizing CPS software testing is proposed. Petri nets are used to model the underlying devices, components, connectors and test cases of CPS software. The test case selection strategy for CPS is proposed and aspect specification provides means to observe behaviors of basic aspect schema, such as test cases, testing process. The weaving mechanism dynamically integrates these schemas into a test model of CPS. The operational semantics and related theories of Petri nets help prove the effectiveness of the proposed method at design time, and dynamically analyze the CPS software testing process. The experimental results demonstrate that the proposed method can not only contribute to the improvement of testing quality, but also to the current research trend.

The remainder of this paper is organized as follows: Section 2 is the related works. Section 3 presents the system framework, and the syntax and semantics of model is also given. In Section 4, the CPS software testing process is modeled. Section 5 presents the analysis techniques. In Section 6, we explain the effectiveness of our methods by several experiments. Section 7 is conclusion.

2. RELATED WORKS

As for an emerging research area, many researchers have carried out the related works for CPS. Though many researchers have realized the importance of studying CPS, most of the studies are focused on the design architecture of CPS [13]. It is less involved in the design and analysis process, and there is no clear consensus on the features of CPS.

With the application of CPS software, CPS testing technology has become a hot research field. [11] proposes a prototype of a model-based CPS testing tool, which includes three phases, that is, test case generation, test case execution, and consistency analysis. Based on the concept of hybrid consistency in Abbas and Fainekos, a CPS consistency testing algorithm is proposed in [12]. The authors in [2] analyze the most commonly used modeling methods for CPS and outline the CPS consistency testing techniques. [11] reduces the overall testing time of highly configurable CPS by using a combination of use case iterative allocation and feedback.

Formal methods provide the support for analyzing and validating CPS software. Some researchers have applied formal methods to CPS, such as Petri nets, process algebra, etc. For example, [7] establishes an attack model of CPS attack model by using Petri nets, and gives a hierarchical modeling method. [17] explains the analysis and verification issues in the CPS information security model, and Petri nets are used to model CPS information flow security attributes and their composition. The author in [16] constructs a CPS security model by using time automata, which includes components and their relationships, attackers and recovery mechanisms. Some scholars try to use formalized models to solve CPS software testing problems. [6] proposes a formal model of Net Condition/Event Systems (NCES), and a test framework for CPS software is given. [14] uses Colored Petri Nets to model the behavior of CPS and its relationships, thus supporting automatic generation of test cases based on coverage criteria.

3. System Framework

3.1. Framework and Requirement

In order to effectively solve the problems in selecting the test cases for CPS software, we propose a framework for modeling and optimizing CPS software testing process, which is shown in Figure 1. This framework introduces the aspect-oriented concepts into the CPS software testing process. The function of CPS software is abstracted as the core concerns, and the testing process and related operations are extracted as the crosscutting concerns. Meanwhile, we provide a formal language to describe the behavior of CPS software and then analyze and verify the properties with the related technologies of Petri nets. The specific process includes three components.
(1) Requirement analysis: we analyze the characteristics of CPS software testing. The aspect-oriented method is adopted to separate the crosscutting concerns and core concerns of CPS software testing. Among them, the crosscutting mainly focuses on the related processes and operations of CPS software testing, while the core concerns are the underlying devices, function modules and their relations, etc.

(2) Modeling CPS software testing process. Based on the high Petri nets, we construct a semantic description model and semantic annotation method that are suitable for characterizing CPS software testing. Petri nets are used as the modeling tools to characterize the relationships between basic elements. Then, we analyze the internal principle of crosscutting concerns, the relationship between models. By adopting the well-defined weaving process and related weaving rules, the test cases and testing process are abstracted as crosscutting concerns and woven into the core model of CPS software, thus forming the test model of CPS software.

(3) Analysis and verification: Petri nets are used to analyze the correctness and testability of CPS software testing, and the evaluation function of test cases is given based on the state space of constructed model. According to the dynamic execution results of test cases, a test case selection strategy is proposed.

Figure 1. System framework

CPS mainly consists of three components, that is, computing, physical and hybrid subsystem. Computing subsystem is mainly used to support the real-time monitoring and data processing. Physical subsystem is the dynamic physical processes underlying the hybrid system, and hybrid subsystem is the bridge between the computing and physical subsystem. Each subsystem contains the corresponding components (computational units), connectors and their properties. Among them, the components can realize the certain functions of the system, such as physical detection, resource conversion, data computation. The connectors are mainly used for realizing the communication requirements between components, such as data transmission, sharing, conversion. The component and connector are the basic elements of a hybrid subsystem.

**Definition 1:** CPS software testing requirements is a six- tuple \( \Xi = \{C, SE, TC, RL, TW, Bp\} \):

1. \( C, SE, T C \) represent the limited set of element, available device and test case.
2. \( RL: C \times C \rightarrow [>, +, ||] \) is a relation function between elements. \( >, + \) and \( || \) represent the sequence, choice and parallel relation.
3. \( TW: C \rightarrow SE^* \) is the function of available device of the element, and \( TW (C_i) = SE = \{SE_{i,1}, SE_{i,2}, \ldots, SE_{i,m}\} \) represents the set of available device of element \( C_i \), \( j \) represents the \( j \)th available device of element \( C_i \).
4. \( Bp: SE \rightarrow TC \times (0,1) \) describes the test suite that can be executed by the available devices and their success probability. \( Bp(SE_i) = (BS(SE_i), sd_i) \).
BT (tc) is the element that can be tested by test cases. This paper assumes that: Each device stores the historical testing data, that is, the previous test can find the wrong test cases of the device. Once the error information of device is detected by the test case, the system immediately removes the error so that the device can be correctly executed. Let \( \forall SE_{ij} \in C, \exists tc_{ij} \in TC \), which makes \( SE_{ij} \in BT (tc_{ij}) \), that is, all the elements can be tested.

3.2. Syntax and semantics of model

Because the function of CPS are composed of many independent elements, the execution process of CPS software is the core concern according to the idea of AOP. While the crosscutting concern refers to the non-functional requirements of the CPS, it mainly refers to the attributes of elements and some control strategies. The focus of this paper is the crosscutting concerns. Accordingly, the model can be divided into core model and test model. The core model is used to describe the execution process of CPS. The weaving of execution process of core model and crosscutting model will form a test model of CPS software based on a certain weaving rules.

Definition 2: A 4-tuple \( \Omega = (\Sigma; \Gamma, TI, TA) \) is called a core net (CN), where:

(1) \( \Sigma \) describes the basic structure of CPS, \( N = (P, T, F) \) is a Petri net, \( P, T, F \) represent the place, transaction and arc. \( IO \subseteq P \) is a special type of place, which is the interface of \( \Omega \) and is represented by a dotted circle. \( D \) is a non-empty individual set of \( \Sigma \), \( f_p, f_s \) are the given formula set of \( D \) (Boolean expressions) and the set of symbols, the individual set determines the types, operations and functions of the annotation in \( \Sigma \). \( A \subseteq T \rightarrow f_s \). For \( t \in T \), the free variable in \( A(t) \) must be in the directed arc with one end of the arc is \( t \). \( A : T \rightarrow f_s \). If \( (p,t) \in F \) or \( (t,p) \in F \), then \( A(t,p) \) or \( A(p,t) \) is the symbol set, the default value is empty. \( \lambda : T \rightarrow N^* \times (0,1) \) is \( \forall t \in T, \lambda(t) = (a_i, \eta_i) \), \( a_i \) describes the priority of transition, the default value is 0. The smaller the value of \( a_i \), the greater the priority of transition. \( \eta_i \) is the firing probability of transition \( t_i \). \( M_0 : P \rightarrow f_s \times R^* \) is the initial marking of \( \Sigma \).

(2) \( \Gamma = \{ I[j] \in N^* \} \) is a finite set of pages, each page is a CN model. \( TI \subseteq T \) is a set of alternative nodes. \( TA \) is a page allocation function.

Individual set \( D \) is mainly used to describe the resources, such as resource and data packet. The task and component can control the operation of CPS by using the message. The data packet of message in CPS is abstracted as an individual \( e \). If there are no special instructions, the individuality in the model is \( e \). \( \forall x \in (P \cup T) \setminus \{ x \} \), we denote the pre-set of \( x \) as \( \{ y | y \in (P \cap T) \} \), the post-set of \( x \) as \( \{ x | x \in (P \cap T) \} \). In order to distinguish the transitions and places in each model, \( N_{ia}x \) is the element \( x \) in model \( N_i \).

The distribution of individuals in each place at time \( t \) is called the marking of \( CN \), denoted by \( M \). \( \forall p \in P, M(p) = (d_i, q_i), (d_i, q_i), \ldots, (d_i, q_i) \) is the sorted individual in place \( p \) under marking \( M \). \( \{ q_i, q_i, \ldots, q_i \} \) is the corresponding number. The firing of transition \( t \) is effective under marking \( M \) if and only if transition \( t \) is enable and it doesn’t have the transition whose priority is greater than \( t \). Let \( M \) be a marking of \( \Omega \). The model will reach a new state \( M' \) by \( M'[H(M)] \). \( M' \) is called the reachable marking. All the possibly reachable states of \( M \) are denoted by \( R(M) \) and \( M \in R(M) \). \( Pro(M) \) is the reachability probability of \( M \), which is equal to the multiply of probability of all transition from \( M_0 \) to \( M \).

4. Modeling CPS software testing process

In this section, we will use Petri nets to model the basic component of CPS and crosscutting concerns, then dynamically weave the concerns into the core model by using weaving rules, thus forming the test model of CPS.

1. The core concerns of CPS software

In this section, \( CN \) is used to model the core concerns of CPS software, and the basic model is constructed by using the hierarchy of \( CN \) model and the interface. The model of component \( CN_i \) is shown in Figure 2. The specific execution flow is as follows: if the component \( C_i \) gets the input parameter \( (M(p_{in}) \neq \emptyset) \), it will fire \( t_i \) to select one available device \( SE_{ij} \) to realize its function. If the device is in the running position \( (M(p_{out}) = d^{p_{out}}_{ij}) \), the transition is fired to make the component run successfully and be in \( p_{out}^0 \).

![Figure 2. CN model of component](image-url)
We will construct the core model of CPS software based on the requirements. The core model of requirement \( \Xi \) is shown in Figure 3, the specific steps are as follows: (1) According to the construction method of component \( CN \), the model of all components in CPS software testing process are constructed. (2) Places and transitions are introduced to describe the beginning and termination operation and position of the whole system. The data transmission process between components is modeled according to the method proposed in [9]. The initial marking \( M_0(p) = \phi \), and the priority of all transitions is set to 3.

(2) The testing concern of available device

The test model of the available device \( SE_{i,j} \) is shown in Figure 4. Places \( p_w, p^O, p_{wC}, p_f, p_{ftc} \) represent the waiting for testing, the correct output interface, the successful test case, the failed position and the failed use case. If \( d^t \in M(p_{in}) \) contains an individual, it fires the transition \( t_{in} \) to do the testing process. Transitions \( t_e, t_f \) represent the successful deployment and failure deployment of testing process, the corresponding firing probability is equal to the probability of successful and failure deployment. After the test, the device will be re-placed into the place \( p_w \) for the next round of test. All transitions have a priority of 4.

(3) Testing concerns of component

The test concerns of component are shown in Figure 5 (a) - (c). Figure 5 (a) is used to describe the process that the component tests each available device: If it gets the input parameters \( (M(p_{in}^t)) \), and the test case is effective for the component \( C_i \), then fire the transition \( t_{in} \) to initialize each device to be tested and puts the test case back into the place \( p_{ct} \) (The
subsequent components can use the test case), while the component is in the testing position \( p_{we} \). If the device \( SE_{ij} \) successfully deploys the test case, then invokes the corresponding operation \( t_{uw} \) to output the individual \( d^c_{ij} \) to the result buffer \( p_{we} \). Similarly, we can reselect test case to do the testing process if the testing of component is failed (Figure 5 (b)), and the correct deployed devices got by the testing may be provided to the component for selecting (Figure 5 (c)). If all available devices have been tested, the test results are summarized \( (t_c) \) and the correct device is stored in the place \( p_r \) for the component.

Testing concern is modeled in Figure 6 (a) - (b). The function of transition \( t_{tc} \) in Figure 6 (a) is used to select a test case (which is called the selected test case) from place \( (p_{new}) \) for this test according to a certain strategy and store the selected test suite in the place \( p_{tu} \). For the failed component testing, we also modeled it (Figure 6 (b)). When the testing of component \( C_i \) fails (all available devices of component \( C_i \) are not deployed correctly under the selected test suite, or if component \( C_i \) does not belong to the effective set of the selected test case), the transition \( t_{tc} \) is fired to cause the system to be in the position to reselect the test case \( (p_{tc}) \), then fires the transition \( t_{tc} \) to reselect the test case.

![Figure 6. Modeling testing concerns](image)

The weaving rules of crosscutting concerns are as follows: We weave the corresponding concerns into the core model \( \Omega \) and add the related subnet models, the priorities of the components and testing concerns are set to be 0 and 3 respectively. The test model of the component can be constructed according to the weaving rules of the concerns. The specific model is shown in Figure 7.

![Figure 7. Modelling crosscutting concerns](image)

5. Model analysis

5.1. Model and analysis

The CN model of component \( C_i \) is denoted by \( CN_i \), the set of transition and place in model \( CN_i \) are marked as \( T_i \) and \( P_i \). \( M \) is the marking of the system, then:

1. The function \( T\ M \ (M,\ CN_i) \) is the value of individual of all the transitions under marking \( M \) in the test model, which is called the mapping of \( M \) on the component \( C_i \). (2) \( \delta^c \in \delta(M, M') \). The firing sequence \( \delta^c_{c-s} = \{t|t \in \delta^c \cap T_i \} \) is called the projection sequence on component \( C_i \). Let \( M \) be a marking of the model, and if \( M(p) = \varepsilon \), then \( M \) is called a normal termination marking of the model, that is, it has realized the required function. The set \( M^\delta (\Omega) \) is the set of the normal termination marking of model \( \Omega \). We can set \( M^\delta = \max\{\text{pro}(M) \} , \ M \in M^\delta (\Omega) \).

Theorem 1: Let \( \Omega \), be the test model of CPS, \( \forall \mathcal{M} \in R(M) \), \( TM(M, C_i) \) is the projection of the marking \( M \) on the component \( C_i \), then \( \forall \mathcal{M} \in R(\mathcal{M}) \), \( \forall \delta^c \in \delta(M, M') \) has \( TM(M, C_i)(\delta^c_{c-s} = TM(M', C_i)) \).

Proof: Because \( TM(M, C_i) \) is the projection of marking \( M \) on the component \( C_i \). According to the definition of function \( TM \), \( \forall P_i \in P_i \), it has \( M(p) = TM(M, C_i) (p) \). \( TM(M, C_i) \) is a marking of component \( C_i \), we can randomly select a greatest firing set \( H_i(M) \) under the marking \( M \).
Because $C_i$ is a component of CPS, according to modeling process of component $C_i$, we can draw that: $\forall t \in T_i$, $(t \in O_i \land p(t)) \land (p_i(t)) \Rightarrow \emptyset$. Therefore, the firing of transition $t_i$ is only related to the change of all individuals in places $p_i$, that is, $\exists M(T_i, C_i)$, there is a greatest firing set $H_2(TM(M, C_i))$, which makes $H_2(TM(M, C_i)) = H_1(M) \land T_i$. Let $TM(M, C_i) = H_2(TM(M, C_i)) \Rightarrow \delta_i \in (M, H_1(D), H_2(M)) \Rightarrow M_2$.

Because $H_2(TM(M, C_i)) = H_1(M) \land T_i$, therefore, $M = TM(M_2, C_i)$, we can get $\forall M' \in R(M) \land \delta_i \in (M, M')$, there is $TM(M, C_i)$ \ $\delta_i \Rightarrow TM(M', C_i)$.

In summary, $\forall M \in R(M_0)$, $TM(M, C_i)$ is the projection of marking $M$ on the component $C_i$, then $\forall M' \in R(M)$, $\exists \delta_i \in (M, M')$, it has $TM(M, C_i)$ \ $\delta_i \Rightarrow TM(M', C_i)$.

Theorem 1 explains that the constructed model can decompose CN model of CPS to each element based on the actual requirements, thus reducing the analysis complexity. Therefore, the use of CN to model and analyze complex CPS software testing is feasible.

**Theorem 2:** Let $\Omega_i$ be the test model of CPS software, $\forall M \in R(M_0)$, $\delta_i$ be a firing sequence from $M_0$ to $M$, $\forall C_i \in C$:

1. If $M(CN_i = p_{in_i}) = d_{i \leftarrow j}$, then $SE_i \in \delta$.
2. If $t_{c_b \leftarrow f} \in FT(M)$, and $\exists d \in M(p_{in_i})$, which makes $C_i \in BT(t_{c_b})$, then $\exists M' \in R(M)$, $CN_i \in \delta_i \in FT(M')$.

**Proof:**

(1) Let $\{M_1, M_2, \ldots, M_k\}$ be the set of marking from the initial marking $M_0$ to $M$.

Because $CN_i = CN_i \in \delta$.

Because $CN_i = CN_i(d_{i \leftarrow j})$, and because $M(CN_i = p_{in_i})$ \ $d_{i \leftarrow j}$, therefore, $CN_i \in \delta$.

Because $CN_i = CN_i(d_{i \leftarrow j})$, we can get $\exists \delta_i \in (M, M_2, \ldots, M_3)$, which makes $d_{i \leftarrow j} \in M(p_{in_i})$.

Because $CN_i = CN_i(d_{i \leftarrow j})$, we can get $CN_i \in \delta$.

Because $CN_i = CN_i(d_{i \leftarrow j})$, we can get $CN_i \in \delta$.

In summary, if $M(CN_i = p_{in_i}) = d_{i \leftarrow j}$, then $SE_i \in \delta$.

(2) Because $t_{c_b \leftarrow f} \in FT(M)$, we can get $M(p_{in_i})$ \ $d_{i \leftarrow j}$, and $CN_i \in \delta_i \in FT(M)$. Therefore, $\exists M \in R(M)$, $M(p_{in_i})$ \ $d_{i \leftarrow j}$.

Because $\exists d \in M(p_{in_i})$, which makes $C_i \in BT(t_{c_b})$, we can get $\exists M_t \in FT(M)$.

Because $\exists d \in M(p_{in_i})$, there is $\exists M_t \in R(M)$, $M_t(p_{in_i})$ \ $d_{i \leftarrow j}$.

Because $CN_i = CN_i(p_{in_i})$, $CN_i \in \delta_i \in FT(M(t_{c_b}))$.

Therefore, $\exists M \in R(M_0)$, $CN_i \in \delta_i \in FT(M')$.

5.2 Selection strategy of test case

In this section, we will propose a selection strategy of test case based on the test model of CPS. Then the selection strategy is viewed as a crosscutting concern and waved into the test model of CPS software, and gives the analysis tools and enforcement algorithm. For complex systems, the larger of the test suite, the greater the management overhead between test cases. Therefore, it is necessary to minimize the size of the selected test cases while ensuring the effectiveness of test cases.

**Definition 3:** Let $VW(M, t_{c_b}) = \{SE_i | t_{c_b} = BS(SE_i) \land M(SE_i, p_{in_i})\}$ be the effective set of test case $t_{c_b}$ under the marking $M$, $val(M, t_{c_b}) = |VW(M, t_{c_b}) \land SE_i| / |SE_i|$ is called the effectiveness of test case $t_{c_b}$ on component $C_i$ under the marking $M$.

The effectiveness of test case $t_{c_b}$ on component $C_i$ is the percentage of the number of devices that the errors can be discovered divided by the total number of available devices of component $C_i$ when test case $t_{c_b}$ is used to test the component $C_i$ under the marking $M$.

$Mval(M, t_{c_b}) = \sum_{C_i \in \delta_i \in FT(T_i)} val(M, C_i, t_{c_b})$ is the highest effectiveness of component $C_i$ based on the set of test case $t_{c_b}$.

**Definition 4:** According to the relationship between total effectiveness and test case, the effectiveness-oriented test case selection strategy is: Initialization the set of selected test case $CTC = \emptyset$, and the set of component to be tested $DC = C$, the set of device to be tested $DSE = SE$:

(1) Computing the optional test case for components: $\forall C_i \in DC$, $OC(C_i) = \{t_{c_b} | C_i \in BT(t_{c_b}) \land t_{c_b} \in Top_n(C_i, TC_i)\}$. The function $Top_n(C_i, TC_i)$ is the first $n$ test cases of component whose probability of successful deployment is high.
(2) If \( DC=\emptyset \), then output \( CTC \), otherwise, it will select the current testing component \( dc \): \( DC \) is ordered in accordance with the value of \( |OC(dc)| \) from the small to large, and the value is assigned to the \( DC \). Taking the first component \( dc \) as the current component, \( dc = dc1 \), then do Step (3).

(3) Evaluating the optional test case for current component \( dc \): \( \forall tc \in OC(dc) \), then compute the effective device of \( tc_j \) under \( DC \). \( EBT(tc_j, dc, DSE) = \{SE_i \mid C_{dc} \wedge BT(tc_j) \wedge SE_i \in SE \wedge tc_j \in BS(SE_i, dc) \} \cap DSE \), then do Step (4).

(4) Selecting the test case for \( dc \): Taking the test case \( tc \) with the highest total effectiveness in \( OC(dc) \) as the test case of \( dc \) and add it to the set of selected test case: \( CTC = CTC \cup tc_j \), \( DSE = DSE \cup EBT(tc_j, dc, DSE) \). If \( SE(dc) \cap DSE=\emptyset \), then \( DC = DC - dc \) and do Step (2), otherwise do Step (4).

The purpose of effectiveness-oriented test case selection is to improve the probability of successful deployment of the set of test case and reduce the overhead of switching between test cases, while ensuring that all devices in the system have at least one test case. The weaving rules of effectiveness-oriented test case selection strategy are as follows: the priority of transition introduced by the concern of selection strategy is set to 2. Under the marking \( M \), the selected test case of component \( C_i \) is \( tc_i \) and the enforcement condition of transition \( \tau_{ci} \) is as follows: \( po(tc_i) \geq po(tc) \), \( d_i \in M(p_{max}) \). The specific enforcement algorithm is shown in Algorithm 1.

**Algorithm 1. Enforcement of algorithm 1**

**Input:** \( CTC = \emptyset \), \( DC = C \), \( DSE = SE \)

**Output:** \( CTC \)

1. foreach(\( C_i \) in \( DC \), \( OC(C_i) \)) Taking the first \( n \) successful deployment of test cases
2. if \( DC \) not null then\( \{sort(\{DC\mid OC\})\} \); // The components to be tested are sorted
3. \( dc = top(\{DC\}) \)
4. foreach(\( tc_i \) in \( OC(dc) \)) Computing \( EBT(tc_i, dc, DSE) \)
5. \( tc = max(M|\{\mu(tc_i, M, \tau_{ci})\}) \), \( \tau_{ci} \) \( OC(dc) \)
6. \( CTC = CTC \cup tc_j \), \( DSE = DSE \cup EBT(tc_i, dc, DSE) \)
7. if \( SE(dc) \cap DSE=\emptyset \) then \( DC = DC - dc \), goto Step 2;
8. else go to Step 5;
9. else output \( CTC \)
10. end

<table>
<thead>
<tr>
<th>( SE )</th>
<th>( BS )</th>
<th>( SE )</th>
<th>( BS )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( SE_{11} )</td>
<td>{(4.95%), (5.87%), (7.90%)}</td>
<td>( SE_{11} )</td>
<td>{(1.83%), (3.80%), (6.97%)}</td>
</tr>
<tr>
<td>( SE_{12} )</td>
<td>{(1.98%), (5.93%)}</td>
<td>( SE_{12} )</td>
<td>{(1.81%), (2.84%), (3.82%)}</td>
</tr>
<tr>
<td>( SE_{13} )</td>
<td>{(1.93%), (4.82%), (5.89%), (6.90%), (7.80%)}</td>
<td>( SE_{13} )</td>
<td>{(6.88%)}</td>
</tr>
<tr>
<td>( SE_{14} )</td>
<td>{(7.96%), (6.88%)}</td>
<td>( SE_{14} )</td>
<td>{(1.82%)}</td>
</tr>
<tr>
<td>( SE_{15} )</td>
<td>{(1.89%), (5.94%)}</td>
<td>( SE_{15} )</td>
<td>{(3.89%), (2.80%), (6.84%)}</td>
</tr>
<tr>
<td>( SE_{16} )</td>
<td>{(1.84%), (3.85%), (5.96%)}</td>
<td>( SE_{16} )</td>
<td>{(3.88%), (6.85%), (12.87%)}</td>
</tr>
<tr>
<td>( SE_{17} )</td>
<td>{(1.89%), (3.91%)}</td>
<td>( SE_{17} )</td>
<td>{(2.89%), (9.91%)}</td>
</tr>
<tr>
<td>( SE_{18} )</td>
<td>{(1.91%), (3.90%), (5.87%)}</td>
<td>( SE_{18} )</td>
<td>{(2.90%), (7.97%)}</td>
</tr>
<tr>
<td>( SE_{19} )</td>
<td>{(3.89%)}</td>
<td>( SE_{19} )</td>
<td>{(2.83%)}</td>
</tr>
<tr>
<td>( SE_{20} )</td>
<td>{(2.93%), (3.90%), (4.97%)}</td>
<td>( SE_{20} )</td>
<td>{(2.93%), (10.83%)}</td>
</tr>
<tr>
<td>( SE_{21} )</td>
<td>{(2.93%), (4.82%), (5.89%), (6.92%)}</td>
<td>( SE_{21} )</td>
<td>{(2.86%)}</td>
</tr>
<tr>
<td>( SE_{22} )</td>
<td>{(3.81%), (6.95%), (9.84%)}</td>
<td>( SE_{22} )</td>
<td>{(2.98%)}</td>
</tr>
<tr>
<td>( SE_{23} )</td>
<td>{(3.89%), (5.94%), (9.97%)}</td>
<td>( SE_{23} )</td>
<td>{(1.84%), (6.93%)}</td>
</tr>
<tr>
<td>( SE_{24} )</td>
<td>{(2.89%), (4.90%), (6.85%)}</td>
<td>( SE_{24} )</td>
<td>{(1.85%), (3.86%), (5.89%), (8.96%)}</td>
</tr>
<tr>
<td>( SE_{25} )</td>
<td>{(2.82%), (5.87%), (6.93%)}</td>
<td>( SE_{25} )</td>
<td>{(1.83%), (3.92%), (2.84%), (5.80%)}</td>
</tr>
<tr>
<td>( SE_{26} )</td>
<td>{(3.89%), (9.74%)}</td>
<td>( SE_{26} )</td>
<td>{(1.89%), (3.85%), (5.89%), (6.87%)}</td>
</tr>
<tr>
<td>( SE_{27} )</td>
<td>{(1.84%), (2.82%), (5.89%), (7.92%)}</td>
<td>( SE_{27} )</td>
<td>{(6.79%), (7.87%)}</td>
</tr>
<tr>
<td>( SE_{28} )</td>
<td>{(1.89%), (2.90%), (7.85%)}</td>
<td>( SE_{28} )</td>
<td>{(4.93%), (6.92%), (8.89%), (8.91%)}</td>
</tr>
<tr>
<td>( SE_{29} )</td>
<td>{(2.99%), (6.86%)}</td>
<td>( SE_{29} )</td>
<td>{(4.82%), (7.81%), (10.91%)}</td>
</tr>
<tr>
<td>( SE_{30} )</td>
<td>{(2.86%), (5.79%), (7.99%)}</td>
<td>( SE_{30} )</td>
<td>{(4.89%), (6.94%)}</td>
</tr>
<tr>
<td>( SE_{31} )</td>
<td>{(2.86%), (10.90%)}</td>
<td>( SE_{31} )</td>
<td>{(7.79%), (8.80%), (10.87%)}</td>
</tr>
</tbody>
</table>

### 6. Experiments

In this paper, we use a Material Balancing System as an example to illustrate the modeling and analysis process. Material Balancing System contains the following components: component \( C_j \) is used to perform the oil distillation and produce kerosene. Component \( C_j \) performs the gas oil industrial cracking, while component \( C_j \) is used to perform residue coking.
industry. Component $C_t$ tracks the storage of individual tank. The balance component $C_b$ accurately tracks the crude oil, the system will carry out material balance according to the actual requirements. Production control component ($C_0$) includes production cost computation ($C_7$) and allocation of production ($C_8$). The execution process can be expressed as $C_1 > C_2 > (C_3 || (C_5 + C_6)) > C_7 > C_8$. The specific available equipment and test cases are shown in Table 1. The material balance system is provided with 10 test cases.

According to Table 2, we can draw that the action set of each test case is $\{C_1, C_2, C_3, C_4, C_5\}, \{C_2, C_4, C_5, C_6, C_7\}, \{C_2, C_5, C_6, C_8\}, \{C_1, C_4, C_5, C_8\}, \{C_1, C_5, C_8\}, \{C_2, C_5, C_6, C_8\}, \{C_1, C_4, C_5, C_8\}, \{C_1, C_4, C_5, C_8\}, \{C_1, C_4, C_5, C_8\}, \{C_1, C_4, C_5, C_8\}$. Based on the requirements of Material Balancing System, the three test concerns are woven into the core model of Material Balancing System according to the construction steps of test model of CPS software. The test model of Material Balancing System is shown in Figure 8.

![Figure 8. Modelling Material Balancing System](image_url)

Based on the execution semantics of the test model, the corresponding properties can be analyzed, which includes the correctness of the model, the correctness of the component execution and testing process, etc. Based on the state space of $\Omega$, we can get the system is executable, that is, there is a sequence of devices which can realize the corresponding function. Mapping into model $\Omega$, there is a state $MeR(M_0)$, and $M(p_0) = 1$. According to the effectiveness-oriented selection strategy (we can take $n = 2$), the selected test case is $\{tc_1, tc_2, tc_3, tc_6, tc_7\}$.

In order to effectively evaluate the effectiveness of the method, this paper designs several experiments and evaluates the effectiveness of the method, thus illustrate the practicality of the method. First, 1000 devices (800 sensors, 200 network devices) are generated as physical resources of the system. The material balance in the simulation includes 48 tasks, 16 sensors and 6 communication processes. And the testing management platform is used to generate 260 test cases. Each device and test case contain the basic information, such as the test requirements, the mapping relationship and the probability of successful deployment ($sd_{ij}$ of successful deployment is greater than 95%).

The purpose of Experiment 1 is to demonstrate the effectiveness of using the core model and the test model. The specific experimental steps are as follows:

1. Taking 30 test cases, the 400 device resources are divided into the available devices of five material balance, and the components in each material balance contain 10, 20, 30, 40 and 50 available devices. Let each material balance has 10 test cases in the initial state, then increase the test cases (The increment is 10). For each material balance, we can do Step (2) - (3).

2. According to the construction steps of the core model, we can construct the core model of material balance and compute the number of reachable states of the corresponding core model.
(3) Construct each crosscutting concern model and integrate it into the core model to form a test model of material balance and compute the number of reachable states of the test model.

The results of experiment 1 (2) show that the state space of core model does not change with the increase of devices under the condition of the same set of test cases. For example, the reachable states of core models of 5 material balance in the experiment are 35. The reason is that core model is mainly used to describe the execution process of material balance, and the size of the test cases and available device. Therefore, the state space of core model is related with the set of test case and the set of available devices. The experimental results in Experiment 1 (3) are shown in Figure 9, which shows that: (1) The state space of test model will non-strictly decrease with the increase of available devices. We can get that the probability of failed component reduces with the abundance of available device, that is, the reselection operation of test case is reduced, so the state space of the model is not strictly decreasing; (2) For the same device, the state of the model does not decrease with the increase of test cases.

![Figure 9. The results of Experiment 1](image)

The purpose of Experiment 2 is to verify the effectiveness of the effectiveness-oriented selection strategy. The specific experimental steps are as follows:

1. Taking 800 devices as the resources, and these devices are also divided into 10 groups. Components in each group of material balance have 1-20 available devices, and the total number of each group is 80.
2. Taking 100 test cases as the set of test case, and each material balance are tested by using these test cases. The test case of each available devices is 1-10.
3. The effectiveness oriented selection strategy is used to construct the selected test case $PST$ for each material balance. The set of test case $SST$ is selected at random, and the set of test case $CST$ is selected based on the probability of successful deployment (Each component selects the test case with the highest success probability). The selection of test cases based on their effectiveness (Each component selects the test case with the maximum effectiveness) $AST$, then compute the size of each selected test cases.
4. Using the above method and further testing and running the material balance, then compute the total probability of successful deployment from component $C_3$ under the above four test cases.

The results of Experiment 2 are shown in Figure 10. We can draw: (1) From the size of test case (Figure 10 (a)), the result of $CST$ is the worst while $AST$ is the best. The result of our proposed method is between them. The main reason is that $CST$ constructs the test suite based on the probability of successful deployment, and the same test case does not necessarily
have the best probability on the different devices. Therefore, the constructed test suite by using the CST is the largest. AST will firstly select the test cases with the best function, so the test suite is the smallest. Although the selected test cases of our proposed method is larger than AST, it is generally smaller than SST and CST. In some cases, for example, the PST in G1 is larger than that in SST. The main reason is that the probability of successful deployment of test cases with relatively better function is low. (2) In the total probability of successful deployment of C3 (Figure 10 (b)), we can get that the value of CST is the highest while that of AST is relatively lower. The overall result of PST is slightly lower than CST. But it is higher than SST and AST in 10 groups.

Based on the above experimental results, the method proposed in this paper reduces the number of test cases while keeping the probability of successful deployment higher, which reduces the analysis complexity.

7. Conclusions

CPS bridges the gap between cyber components, typically written in software, and the physical world. In this paper, we propose a test model of CPS and test case selection strategy based on the faced problems in CPS software. The example and simulation results show that the method can achieve the following results: (1) We research the test model of CPS software by analyzing the related requirements, which is used to characterize the basic elements of CPS, which includes the device, components, connectors, test cases and their relations. The related techniques of Petri net theory is used to prove that the test model can correctly characterize the execution process of CPS software testing process and dynamically reflects the test case selecting process based on the state space of the model. (3) A test case selection strategy is proposed based on the probability of successful deployment of test cases. This strategy improves the total probability of successful deployment of the selected test case and reduces the number of selected test cases, thus reducing the switching number of test case. (4) It has the advantages of using AOP ideas; the state space of corn model and test model will grow with increasing scale of system. While we only use the corn model in analysing the functional properties of CPS, the test model is used to analyse the CPS software testing process.

Acknowledgements

This work is partially supported by the NSF of China under grants No. 61772200, No. 61702334 and No. 61472139, Shanghai Pujiang Talent Program under grants No. 17PJ1401900. Shanghai Municipal Natural Science Foundation under Grants No. 17ZR1406900 and 17ZR1429700. Educational Research Fund of ECUST under Grant No. ZH1726108. The Collaborative Innovation Foundation of Shanghai Institute of Technology under Grants No. XTCX2016-20.

References


**Biography**

**Liqiong Chen** received his B.S. degree from Anhui University of Technology in 2004, and Ph.D. degree from East China University of Science and Technology (ECUST) in 2009. She is presently an associate professor of the Department of Computer Science and Information Engineering Shanghai Institute of Technology. Her research interests include formal methods for complex software systems.

**Guisheng Fan** received his B.S. degree from Anhui University of Technology in 2003, M.S. degree from East China University of Science and Technology (ECUST) in 2006, and Ph.D. degree from East China University of Science and Technology in 2009, all in computer science. He is presently a research assistant of the Department of Computer Science and Engineering, East China University of Science and Technology. His research interests include formal methods for complex software systems, service oriented computing, and techniques for analysis of software architecture.

**Huiqun Yu** received his B.S. degree from Nanjing University in 1989, M.S. degree from East China University of Science and Technology (ECUST) in 1992, and Ph.D. degree from Shanghai Jiaotong University in 1995, all in computer science. He is currently a Professor of computer science with the Department of Computer Science and Engineering at ECUST. From 2001 to 2004, he was a Visiting Researcher in the School of Computer Science at Florida International University. His research interests include software engineering, high confidence computing systems, cloud computing and formal methods. He is a member of the ACM, a senior member of the IEEE, and a senior member of the China Computer Federation.