Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Architecture-level particular risk modeling and analysis for a cyber-physical system with AADL*

Ming-rui XIAO[‡1], Yun-wei DONG[‡1], Qian-wen GOU[1], Feng XUE[2], Yong-hua CHEN[2]

[1]*School of Computer Science and Engineering, Northwestern Polytechnical University, Xi'an 710072, China*

[2]*Nari Group Corporation/State Grid Electric Power Research Institute, Nanjing 210000, China*

E-mail: xiaomingrui@mail.nwpu.edu.cn; yunweidong@nwpu.edu.cn; gqwen@mail.nwpu.edu.cn; xue-feng@sgepri.sgcc.com.cn; chenyonghua@sgepri.sgcc.com.cn

Received Aug. 25, 2020; Revision accepted Oct. 6, 2020; Crosschecked Oct. 15, 2020

Abstract: Cyber-physical systems (CPSs) are becoming increasingly important in safety-critical systems. Particular risk analysis (PRA) is an essential step in the safety assessment process to guarantee the quality of a system in the early phase of system development. Human factors like the physical environment are the most important part of particular risk assessment. Therefore, it is necessary to analyze the safety of the system considering human factor and physical factor. In this paper, we propose a new particular risk model (PRM) to improve the modeling ability of the Architecture Analysis and Design Language (AADL). An architecture-based PRA method is presented to support safety assessment for the AADL model of a cyber-physical system. To simulate the PRM with the proposed PRA method, model transformation from PRM to a deterministic and stochastic Petri net model is implemented. Finally, a case study on the power grid system of CPS is modeled and analyzed using the proposed method.

Key words: Human-cyber-physical system (HCPS); Particular risk analysis; Architecture Analysis and Design Language (AADL); Deterministic and stochastic Petri net (DSPN); Particular risk model

https://doi.org/10.1631/FITEE.2000428 **CLC number:** TP311

1 Introduction

The cyber-physical system is a safety-critical embedded system composed of human interaction, physical process, and computing process, which are integrated deeply with each other. It is also called the human-cyber-physical system (HCPS) (Ji et al., 2019). The computing process is constrained by human interaction and the physical process through perception, and the results of the computing process will affect human interaction and the physical process. In 2000, debris from an aircraft tire burst penetrated the fuel tank and ignited the engine, causing the hydraulic system to fail and eventually a crash (Bi, 2017). In 2009, at the Itaipu Hydropower Station in Brazil, because of heavy rainfall, the insulation of a transmission line was reduced and there were many short-circuit faults. At the same time, the safety and stability control system (SSC) performed incorrect actions, causing the system to oscillate and eventually collapse. Human factors and the physical environment play important roles in the safety assessment of HCPS. The safety assessment process begins in the early phase of system development, and runs through all the system development phases. The safety assessment process includes requirement generation and verification, both supporting system development. The process provides a methodology of evaluating system functions to determine whether the associated hazards have been

‡ Corresponding authors

* Project supported by the State Power Grid Company Science and Technical Plan Project, China (No. 5100-201940008A-0-0-00)

ORCID: Ming-rui XIAO, https://orcid.org/0000-0002-1926-9590; Yun-wei DONG, https://orcid.org/0000-0001-9882-9121

properly addressed. The American Society of Automotive Engineers (SAE) has released a standard, Aerospace Recommended Practice ARP4761 (Society of Automotive Engineers, 1996), for the aircraft safety assessment guidelines, and this standard includes particular risk analysis (PRA). According to the ARP4761 standard, the safety assessment process is divided into four phases: functional hazard assessment (FHA), preliminary system safety analysis (PSSA), system safety analysis (SSA), and common cause analysis (CCA). In the CCA phase, there are three main analysis methods: zonal safety analysis (ZSA), common mode analysis (CMA), and PRA. The first three phases are used for evaluating safety of the software function of the system inside, and in the last phase, PRA is carried out on the architecture. PRA is an analytical method that evaluates particular risks outside the system. The purpose is to analyze the impact of external events that may cause serious or catastrophic damage to the system. The PRA method is also applicable to other cyber-physical systems.

There are three shortcomings of the traditional PRA method. First, particular risks in the standard take into account only the natural environment outside the system and the hardware installed inside the system. The traditional PRA method does not consider human factors, and there is no detailed description or classification for complex risks. It leads to a lack of a significant basis for the particular risk assessment of HCPS. Second, there is no complete process to provide specific analysis guidance in the traditional PRA method. This deficiency makes it difficult to clearly explain how particular risks affect the HCPS. Third, the qualitative analysis method is used to analyze system safety in the traditional PRA method, such as logical reasoning and experience summarization. However, the qualitative analysis process lacks accurate data to provide support, and it is easily misled by man's will or experience. Therefore, the traditional PRA method needs improvement.

Human activity has become one of the most important factors affecting system safety. According to statistics from relevant agencies, about 70%–90% of all the human-machine system failures in the world today are directly or indirectly caused by human factors (Luo, 2017). Furthermore, the bad actuation of external physical to system operation may cause some system anomalies. Several system accidents have been caused by terrible environmental factors in recent years. According to statistics, from 2011 to 2013, accidents caused by lightning strikes, typhoons, and icing accounted for 67.15% of all the faults in the State Power Grid Company (Zou, 2015). If these problems are analyzed and found in the late stage of system development, it will cost a lot of time and money to rework. Therefore, these particular risks need to be considered in the early design stage of the system. Architecture Analysis and Design Language (AADL) (Society of Automotive Engineers, 2017) is an excellent modeling language that supports non-functional attribute analysis during the early development phase with error model annex (Society of Automotive Engineers, 2013). However, AADL has weaknesses in the PRA evaluation of HCPS. The AADL core language standard supports modeling of the software components, execution platform components, and composite components, and error model annex is used in the safety analysis of the information system. However, it cannot support modeling of human factor and physical factor. Therefore, it is necessary to propose a particular risk model (PRM) to support our PRA method.

Given the problems existing in the traditional PRA method and AADL language, a PRA method based on AADL PRM is proposed. In our method, the semantics of the abstract component is extended to support particular risk component modeling. AADL provides an annex extension mechanism to support researchers in system modeling, so we use this extension method to design a human component and a physical component. According to the software requirement and human-machine operation specifications, we need to establish the architecture model, human component model, and physical component model. Through these three models, the particular risk information is described in detail. This effectively overcomes the shortcomings of description ability and classification in the traditional PRA method. Then we need to establish an error model based on the system's safety requirements. Error propagation in the error model is the link between PRM and the architecture model, so that these models can be integrated to construct the PRA model. Our methodology performs impact analysis to determine the particular risks, failure states, operational requirements, and the propagation path of risk. According to the state transitions between components, the specific process of how the particular risks affect the system can be worked out. To further assess the impact of the particular risks, the occurrence probability is indispensable. The deterministic and stochastic Petri net (DSPN) (Marsan and Chiola, 1987) model is suitable for describing and analyzing the dynamic behavior of complex systems. Therefore, the PRA model is transformed to the DSPN model for the purpose of adopting the simulation tool TimeNet (Zimmermann, 2017) to calculate the probability of system failure. The quantitative analysis method weakens the bias caused by man's will, and overcomes the disadvantages of the traditional PRA method which does not have sufficient data to support it.

Contributions of this paper are summarized mainly as follows: (1) We extend an AADL subclause language as an AADL-PRA annex model with a human component and a physical component, and integrate the proposed model with an architecture model and an error model into a PRA model. (2) A new PRA analysis method is proposed based on the PRA model to obtain a PRA analysis table.

2 Related work

The AADL with EMV2 can support system fault modeling and safety analysis in the early phase of system development. Dong et al. (2011) and Delange and Feiler (2014) built an AADL reliability model based on an architecture model and an error model, transformed the reliability model to a generalized stochastic Petri net (GSPN) model, and calculated the system reliability. However, the AADL core language cannot provide sufficient modeling of HCPS, and lacks the capability to model human factors and physical information. It can analyze only from the software aspect without considering whether the external physical risk will affect the information system.

Edward's SHEL model can be used as a conceptual model of human factors in safe work (Wang et al., 2017). The model name is composed of the first letters of software, hardware, environment, and liveware. It shows the main factors and interactions to form the interface with human. This model is used to analyze human factors and sources of human error. James Reason of the University of Manchester put forward the famous Reason model in his psychology book *Human Error* (Reason, 1990), which revolutionized the aviation and other industries' views on safety. Reason believed that accidents occur where there is a problem with the interaction between system elements during production. These failures destroy the integrity of the system and make the system vulnerable to unsafe manipulation factors, leading to catastrophic consequences. However, they have not put forward a complete human factor analysis model to standardize human error modeling. Therefore, it is necessary to extend the PRM at the architecture level to support the analysis in the early stage of system development.

Banerjee et al. (2012) proposed a framework called "body area network and device analysis and design" (BAND-AiDe) that enables the modeling of the cyber-physical system (CPS) using an extended AADL. This framework supports the dynamic behavior modeling of physical processes (i.e., differential equations) by developing a new annex called BAN-CPS. Nonetheless, the capacity of the BAN-CPS annex is limited to supporting environmental factors and physical entities with physical behavioral characteristics, such as physiological signals, blood pressure, and skin temperature. In addition, their method does not support the modeling of human subjective behavior, or consider the impact of physical anomalies on the computing control system.

Kim et al. (2006) used Korea-HRA and human error assessment and reduction technique (HEART) human factor reliability analysis methods in the analysis of accidents such as the collision of passenger trains and trucks, and compared the advantages and disadvantages of each method. However, these methods do not model the process of human factors in detail. In addition, there is no further consideration of the impact of human behavior on system safety.

Wei et al. (2019) proposed model-based safety analysis for grid cyber-physical systems (GCPSs) using probabilistic model-checking of stochastic multiplayer games (SMGs), and built a model for GCPS including the environment. It uses SMGs to model the system and environment as two players that compete for a particular goal. However, as it cannot describe human activity or physical behavior, it is not sufficient to analyze the detailed particular risk behavior. Our previous work (Wei et al., 2014) has created hazard model annex (HMA) to compensate for EMV2. The safety model can be constructed by annotating the architecture model with the error model and hazard model. Factors such as spark and flame are simply considered as possible error events in the system. However, CPS is closely related to the external environment. With the change of the external physical environment or different inputs by a human, the system will perform different tasks. Therefore, based on past work, we expand a more detailed PRM and describe how particular risks affect the safety of the cyber system in CPS.

3 Particular risk model

To make AADL support modeling of HCPS, we first define formal semantics for a PRM. Particular risks refer to the events related to human errors and mutation of physical factors, which may put the system into an unsafe state and then lead to system failure. Therefore, the formal semantics of the human component model and physical component model is separately introduced in detail. Then, the syntax rules of the PRM annex are defined based on the AADL.

The PRM annex provides the human component model and physical component model to describe human operation and the physical process, respectively. These are integrated with the computing system, and each component is specifically modeled via a particular risk (PR) annex library and PR annex subclause.

3.1 Formalization of the particular risk model

3.1.1 Human component model

The human component model describes actors who trigger and participate in the operation of the HCPS, and a series of actions taken by actors to complete specific tasks. As shown in Fig. 1, the human and the computing system can interact through sensors and actuators. The computing system can perceive a specific human operation through sensors, and the human can receive the information conveyed by the computing system through actuators. The human component model is defined as a tuple $\mathrm{HM} = (A, \mathrm{IM}, \mathrm{IP}, \mathrm{OI}, \mathrm{RP}, \mathrm{II}, \mathrm{ISQ})$.

1. A is a set of all actors, A={a_1, a_2, $\cdots$, a_n}. The element a_i ($i = 1, 2, \cdots, n$) is a single actor that interacts directly with the system during system operation. Each a_i plays a role in completing the corresponding action during the operation of HCPS. The actors with different roles have different access permissions. Therefore, the role type needs to be defined for each actor. Here is an example:

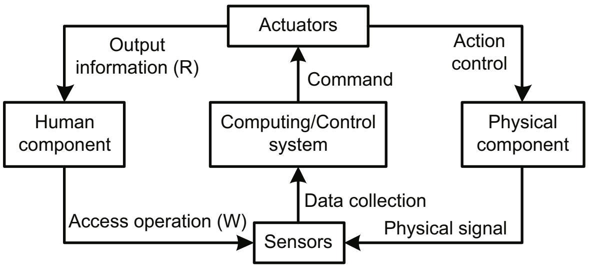


Fig. 1 Framework of the human-cyber-physical system

```
Actor:
  a1: captain–actor is given the role of captain
  a2: navigator–actor is given the role of navigator
```

2. IM is a set of all interaction modes that may exist in the execution of a task, IM={im_1, im_2, $\cdots$, im_n}. The element im_i is defined as a specific access operation where the actor interacts with the system, such as inputting instructions and reading parameters. Each im_i is defined to be executed by only one actor. There are multiple interaction methods between the actor and the system, but all interaction methods can be abstracted into two types of access operations: read operation (R) and write operation (W). The operation R indicates that the actor obtains the required information from the system, and the operation W indicates that the actor enters relevant information during system operation. To describe the interaction information in detail, the property set needs to be defined for each interaction mode im_i. Here is an example:

```
interactiveMode:
  im1: OperationSpecification=>[
       TYPE: W;
       operationName: load shedding;
       NHEP: 5.0E-6–nominal human error probability
      EPC: {2.2, 1.4, 1}–error producing condition
      APOA: {0.5, 0.3, 0.16}–actual impact value of EPC
  ];
```

3. IP is a set of all interaction interfaces, IP={ip_1, ip_2, $\cdots$, ip_n}. The element ip_i is a human-machine interaction interface between the human and the information system, such as monitor and console. ip_i can refer to the port which provides

component features in the AADL, ip_i∈Ports. Ports can be divided into in ports and out ports according to the transmission direction, and can also be divided into data ports, event ports, and event data ports according to whether there is a receiving queue. Similarly, the property set ipp_i is defined for each interactive interface ip_i.

4. OI is a set of all relations between IM and IP, OI=R_{r}∪R_{w}, where R_{r}=IM×IP is a set of relations between the read operation and the interaction interface and R_{w}=IM×IP is a set of relations between the write operation and the interaction interface. OI={oi_1, oi_2, ⋯, oi_n}, where oi_i indicates that the interaction mode om_i is executed through the interaction interface ip_i. Here is an example:

OperationInterface:
oi_1: (im_1, ip_1)–im_1 is executed through ip_1

5. RP is a set of all role permissions existing in the human component model, RP={rp_1, rp_2, ⋯, rp_n}. Each element rp_i needs to define the operation interface that the role type can execute, rp_i={oi_1, oi_2, ⋯, oi_m}, where oi_1∈OI. rp_i indicates that the role with permission rp_i can execute operations such as oi_1, oi_2, ⋯, oi_m. During system operation, actors with different role types are assigned different permissions.

6. II is a set of all relations between interaction subject IS and OI that represents the interaction intent of actors. IS={is_1, is_2, ⋯, is_n}. Each interaction subject is_i is represented as a set of actors that access the operation interface oi_i. According to the definition of role permission, the actor who can execute the operation interface oi_i is defined as the interaction subject is_i. is_i={a_1, a_2, ⋯, a_m} indicates that the operation interface can be executed by a_1, a_2, ⋯, a_m. II = IS×OI. The interaction intention ii_i indicates that the operation interface oi_i can be executed by is_i. We define a logical operator ∗ which indicates that the left and right sides of the operator are optional return values; that is, $\mathrm{ii}_1 = (\{a_1, a_2, a_3\}, \mathrm{oi}_1) = (a_1, \mathrm{oi}_1)*(a_2, \mathrm{oi}_1)*(a_3, \mathrm{oi}_1)$.

7. ISQ is a set of all interaction sequences existing in the human component model, ISQ={isq_1, isq_2, ⋯, isq_n}. Each interaction sequence isq_i is defined as a chronological sequence, such as isq_i={ii_1^1, ii_4^2, ii_2^3, ii_4^4, ii_3^5}, where ii_i^j∈II (i refers to the serial number of ii and j refers to the clock variable of ii). The interaction sequence is used to describe the sequence of multiple interaction intent executions during system operation. The interaction subject may execute the same operation interface multiple times to complete different interaction intents. The sequential relation of the interaction intent is executed sequentially under the constraint of the clock variable. Therefore, the clock variable needs to be defined to distinguish different interaction intents. Each interaction intent consumes a unit of time, such as (ii_1^1, ii_1^2). When the clock is 1, the interaction intent ii_1 is executed, and when the clock is 2, the interaction intent ii_1 is executed again. The order between the interaction intents also satisfies the partial ordering relation, but it does not meet the total ordering relation.

3.1.2 Physical component model

CPS is a system in which the computing process and the physical system cooperate deeply. The behavior of a physical system is a continuous process changing with time. As shown in Fig. 1, the external physical changes are monitored and sensed by sensors, and the detected signals are sent to the computing system. The external physical behavior continuously changes with time. When the external physical parameters change to a certain value, they will affect the normal operation of the sensor and further cause the system to change its operating state. At the same time, the system can further affect the physical system by controlling the actuator components.

Because the operation state of HCPS is closely related to the external natural environment and physical machinery, the structure and behavior of the external operation environment of the system can seriously threaten the safety of system operation. To dynamically and accurately describe the hybrid characteristics of the physical component model, the physical component model is defined as a tuple PM=(BS, BS_0, PV, CV, CB, CD, T).

1. BS is a finite set of discrete behavioral states in the physical component model, $\mathrm{BS} = \{\mathrm{bs}_1, \mathrm{bs}_2, \cdots, \mathrm{bs}_n\}$. Each bs_i represents the behavioral state that the physical component may be in.

2. BS_0⊆BS is the initial state of the physical component model, $\mathrm{BS}_0 = \{\mathrm{bs}_1, \mathrm{bs}_2, \cdots, \mathrm{bs}_m\}$. bs_i $(i = 1, 2, \cdots, m)$ indicates that the system is running normally.

3. PV is a set of all physical variables, $\mathrm{PV} = \{\mathrm{pv}_1, \mathrm{pv}_2, \cdots, \mathrm{pv}_n\}$. Each pv_i represents a defined

physical variable. The physical behavior modeling considers mainly the change of characteristic parameters of physical behavior. Therefore, accurate mathematical variables need to be used to characterize the physical behavior. The variables can be discrete variables defined on the set of integers, or continuous variables defined on the set of real numbers. The physical variables are defined as

$$< \mathrm{pv}_i >:< \mathrm{UnitType} >=< \mathrm{InitialValue} >.$$

4. CV is a set of all global clock variables in the physical component model, $\mathrm{CV} = \{\mathrm{cv}_1, \mathrm{cv}_2, \cdots, \mathrm{cv}_n\}$, and the number of elements in the set is limited. The physical behavior and clock constraints affect the physical components.

5. CB is a set of all continuous behaviors in behavior state bs_i, and describes the change of external physical variables when the physical component is in the state bs_i. The physical variables will change as the clock variables change. The change of the variables in the state can be expressed by a function. For example, the temperature in the external environment follows a normal distribution, $\mathrm{temp}{\sim}N(\mu, \sigma^2)$.

6. CD is a set of all trigger conditions, $\mathrm{CD} = \{\mathrm{cd}|\mathrm{cd} = (\mathrm{tb}, \mathrm{cs}, \mathrm{cr})\}$, and describes the dynamic change process of behavior states.

(1) tb is a set of all triggering behaviors, $\mathrm{tb} = \{\mathrm{tb}_1, \mathrm{tb}_2, \cdots, \mathrm{tb}_n\}$. tb_i describes the constraint condition of the physical variables. When the constraint conditions are not satisfied, the behavior state of the physical component may change. Here is an example:

$$< \mathrm{tb}_i >:< \mathrm{ConstraintModule} >.$$

The constraint module is used to encapsulate reusable constraint expressions. It is an equation or inequality that indicates the triggering condition of the transition.

(2) cs is a set of all clock constraints, $\mathrm{cs} = \{\mathrm{cs}_1, \mathrm{cs}_2, \cdots, \mathrm{cs}_n\}$. We define the corresponding clock constraint according to the clock variable $\mathrm{cs}_i{\in}\mathrm{CV}$. When tb_i is received and triggered and the clock constraint expression cs_i is true at this time, state $\mathrm{bs}_i{\in}\mathrm{BS}$ can change. Otherwise, it cannot change. Here is an example:

$$< \mathrm{cs}_i >:< \mathrm{ClockConstraintsModule} >.$$

The clock constraint module is used to encapsulate reusable clock constraint expressions. It is an equation or inequality that restricts the state transition.

(3) cr is a set of all clock resets, $\mathrm{cr} = \{\mathrm{cr}_1, \mathrm{cr}_2, \cdots, \mathrm{cr}_n\}$. When trigger event tb_i and time constraint cs_i are fulfilled at the same time, some global clock variables cv_i need to be reset.

Therefore, trigger conditions cd_i are defined as

$$< \mathrm{cd}_i >:< \mathrm{tb}_i > [< \mathrm{cs}_i >]/[< \mathrm{cr}_i >].$$

7. T is a set of all state transitions included in the physical component model, $T = \{t_1, t_2, \cdots, t_n\}$. t_i is defined as follows:

$$< t_i >:< \mathrm{bs}_i > \text{-}[< \mathrm{cd}_i >]\text{-}> < \mathrm{bs}_j >,$$

where $\mathrm{bs}_i, \mathrm{bs}_j{\in}\mathrm{BS}$ and $\mathrm{cd}_i{\in}\mathrm{CD}$.

3.2 Syntax

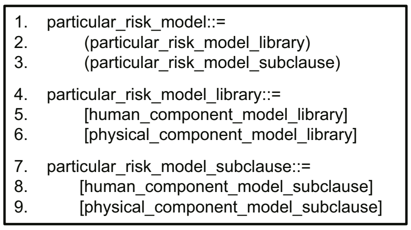
AADL provides an annex extension mechanism to support researchers in architectural modeling and non-functional attribute analysis of the system.

Based on the AADL core language standard and the AADL PRM semantics discussed in Section 3.1, we design the AADL PRM annex (PRMA) to support the modeling of particular risk information, and describe the PRMA syntax with the extended Backus-Naur form (EBNF). The PRMA includes the PRM library (PRMLib) and the PRM subclause (PRMSub), as shown in Fig. 2. The PRMLib defines some reusable declarations, including the human component model library (HMLib) and the physical component model library (PMLib). These reusable declarations can be reused in any of the PRMSub in the same package to form a human component model subclause (HMSub) and a physical component model subclause (PMSub).

3.2.1 Syntax of the human component model

1. Human component model library

HMLib contains a set of reusable declarations, such as role types and actors, which can be referenced



```
particular_risk_model::=
        (particular_risk_model_library)
        (particular_risk_model_subclause)

particular_risk_model_library::=
        [human_component_model_library]
        [physical_component_model_library]

particular_risk_model_subclause::=
        [human_component_model_subclause]
        [physical_component_model_subclause]
```

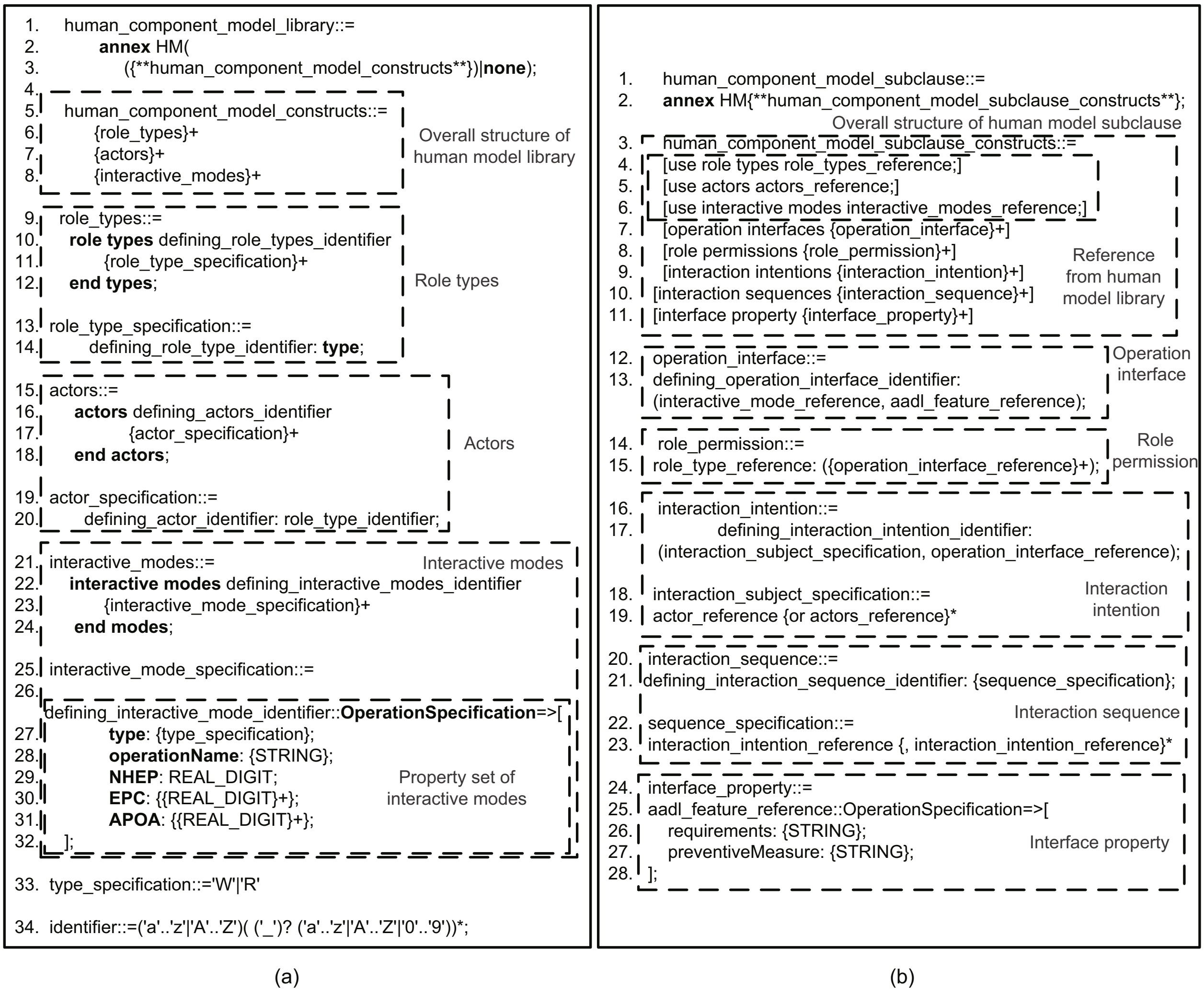
Fig. 2 Particular risk model annex

in HMSub. The grammar rules for the syntax of HMLib are shown in Fig. 3a.

HMLib primarily defines the role types, actors, and interactive modes. The role types and actors are defined to indicate to which type the actor belongs. Each actor plays a corresponding role in the operation of the system. The interactive mode is defined to indicate a specific access operation where the actor interacts with the system. By summarizing all access operations in HCPSs, they can be abstracted into two types, read operation and write operation, which are used to describe the access operations needed to model the interactive mode of the system. Finally, we define a property set for each interaction mode including the operation type, operation name, nominal human error probability (NHEP), error producing conditions (EPCs), and assessed proportion of affect (APOA). NHEP defines the probability that operators make mistakes when performing certain tasks, EPC defines the factors that may cause operators to make errors in certain situations, and APOA refers to specific scores made by domain experts based on the degree of influence of EPC-induced errors.

2. Human component model subclause

HMSub is defined in the architecture model according to the requirements. The grammar rules of HMSub are shown in Fig. 3b. In a human component model, if the actor performs some operation modes on the interaction interfaces, the component can be annotated with HMSub. The actor type and interaction mode defined in HMLib need to be referenced. All operations allowed by the interface are


Overall structure of human model library
Role types
Actors
Interactive modes
Property set of interactive modes
Overall structure of human model subclause
Reference from human model library
Operation interface
Role permission
Interaction intention
Interaction sequence
Interface property
(a)
(b)

```
human_component_model_library::=
      annex HM(
         ({**human_component_model_constructs**})|none);

human_component_model_constructs::=
      {role_types}+
      {actors}+
      {interactive_modes}+

role_types::=
   role types defining_role_types_identifier
      {role_type_specification}+
   end types;

role_type_specification::=
      defining_role_type_identifier: type;

actors::=
    actors defining_actors_identifier
          {actor_specification}+
    end actors;

actor_specification::=
     defining_actor_identifier: role_type_identifier;

interactive_modes::=
   interactive modes defining_interactive_modes_identifier
      {interactive_mode_specification}+
    end modes;

interactive_mode_specification::=

defining_interactive_mode_identifier::OperationSpecification=>[
        type: {type_specification};
        operationName: {STRING};
        NHEP: REAL_DIGIT;
        EPC: {{REAL_DIGIT}+};
        APOA: {{REAL_DIGIT}+};
];

type_specification::='W'|'R'

identifier::=('a'..'z'|'A'..'Z')( ('_')? ('a'..'z'|'A'..'Z'|'0'..'9'))*;
```

```
human_component_model_subclause::=
annex HM{**human_component_model_subclause_constructs**};

human_component_model_subclause_constructs::=
[use role types role_types_reference;]
[use actors actors_reference;]
[use interactive modes interactive_modes_reference;]
[operation interfaces {operation_interface}+]
[role permissions {role_permission}+]
[interaction intentions {interaction_intention}+]
[interaction sequences {interaction_sequence}+]
[interface property {interface_property}+]

operation_interface::=
defining_operation_interface_identifier:
(interactive_mode_reference, aadl_feature_reference);

role_permission::=
role_type_reference: ({operation_interface_reference}+);

interaction_intention::=
        defining_interaction_intention_identifier:
 (interaction_subject_specification, operation_interface_reference);

interaction_subject_specification::=
actor_reference {or actors_reference}*

interaction_sequence::=
defining_interaction_sequence_identifier: {sequence_specification};

sequence_specification::=
interaction_intention_reference {, interaction_intention_reference}*

interface_property::=
aadl_feature_reference::OperationSpecification=>[
   requirements: {STRING};
   preventiveMeasure: {STRING};
];
```

Fig. 3 Basic syntax of the human component model: (a) human component model library; (b) human component model subclause

defined in the operation interface section. The operational interfaces that each role type can perform are defined in the role permission section. The interaction intent during system operation is defined in the interaction intent section, including the interaction subject and operation interface. Since multiple interaction intents will follow the execution order, the order of interaction intents needs to be defined in the interaction sequence.

3.2.2 Syntax of the physical component model

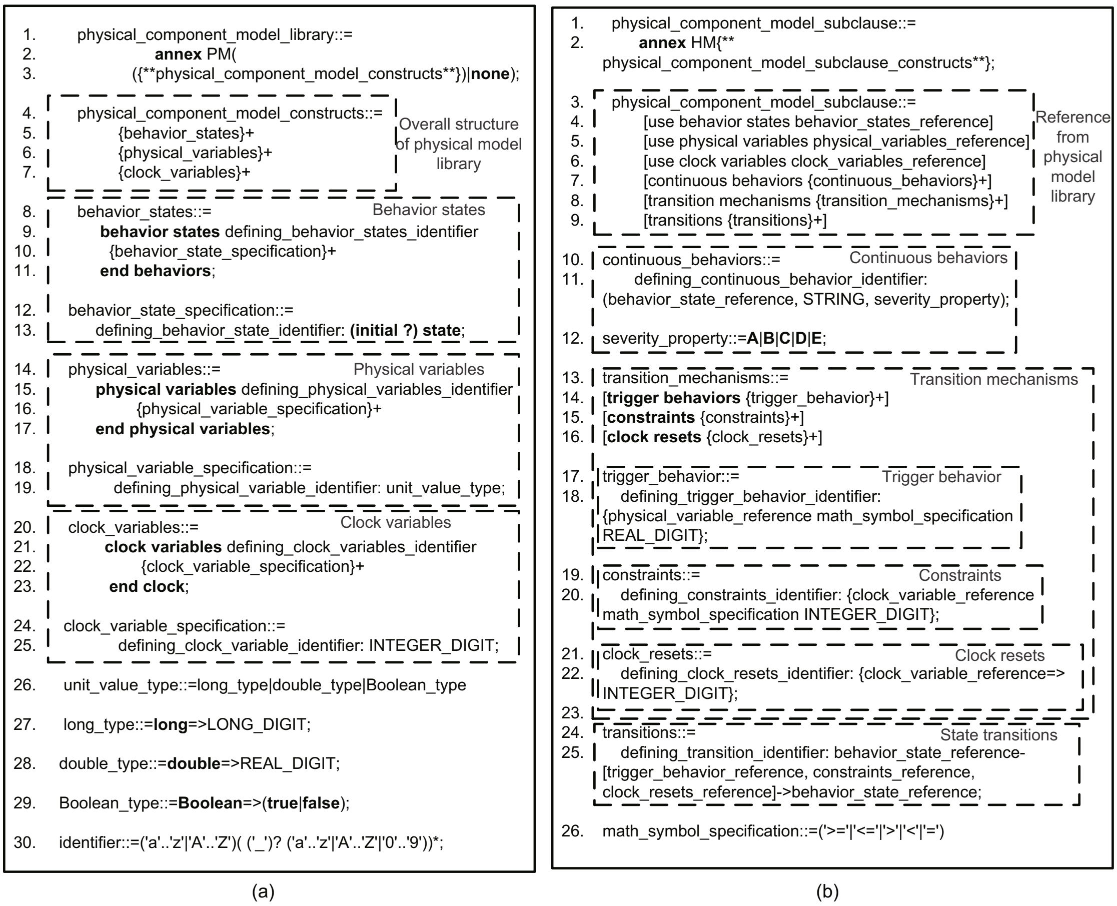
1. Physical component model library

PMLib contains a set of reusable declarations, such as role types and actors, which can be referenced in PMSub. The grammar rules for the syntax of PMLib are shown in Fig. 4a.

PMLib primarily defines the behavior states, physical variables, and clock variables. The behavior state is defined to indicate a stable and continuous state in which the physical component model may be. The physical and clock variables are defined to indicate the changing physical and clock parameters in the physical component model, respectively.

2. Physical component model subclause

PMSub is defined in the architecture model according to the requirements, and the grammar rules of PMSub are shown in Fig. 4b. In a physical component, if the defined physical parameters change with the clock, the component can be annotated with PMSub. The behavior state and physical and clock variables defined in PMLib need to be referenced. The change function of all physical variables is defined in the continuous behavior section. The transition constraints of the behavior state are defined in the transition mechanism section, which includes mainly triggering behavior, clock constraint, and clock reset. Transitions between behavior states are defined in the transition section to reflect the discrete characteristics of physical components.



```
physical_component_model_library::=
            annex PM(
          ({**physical_component_model_constructs**})|none);

physical_component_model_constructs::=
      {behavior_states}+
      {physical_variables}+
      {clock_variables}+

behavior_states::=
   behavior states defining_behavior_states_identifier
     {behavior_state_specification}+
   end behaviors;

behavior_state_specification::=
    defining_behavior_state_identifier: (initial ?) state;

physical_variables::=
    physical variables defining_physical_variables_identifier
          {physical_variable_specification}+
    end physical variables;

physical_variable_specification::=
       defining_physical_variable_identifier: unit_value_type;

clock_variables::=
      clock variables defining_clock_variables_identifier
           {clock_variable_specification}+
      end clock;

clock_variable_specification::=
       defining_clock_variable_identifier: INTEGER_DIGIT;

unit_value_type::=long_type|double_type|Boolean_type

long_type::=long=>LONG_DIGIT;

double_type::=double=>REAL_DIGIT;

Boolean_type::=Boolean=>(true|false);

identifier::=('a'..'z'|'A'..'Z')( ('_')? ('a'..'z'|'A'..'Z'|'0'..'9'))*;
```

(a)

```
physical_component_model_subclause::=
       annex HM{**
physical_component_model_subclause_constructs**};

physical_component_model_subclause::=
      [use behavior states behavior_states_reference]
      [use physical variables physical_variables_reference]
      [use clock variables clock_variables_reference]
      [continuous behaviors {continuous_behaviors}+]
      [transition mechanisms {transition_mechanisms}+]
      [transitions {transitions}+]

continuous_behaviors::=
      defining_continuous_behavior_identifier:
(behavior_state_reference, STRING, severity_property);

severity_property::=A|B|C|D|E;

transition_mechanisms::=
[trigger behaviors {trigger_behavior}+]
[constraints {constraints}+]
[clock resets {clock_resets}+]

trigger_behavior::=
   defining_trigger_behavior_identifier:
{physical_variable_reference math_symbol_specification
REAL_DIGIT};

constraints::=
   defining_constraints_identifier: {clock_variable_reference
math_symbol_specification INTEGER_DIGIT};

clock_resets::=
   defining_clock_resets_identifier: {clock_variable_reference=>
INTEGER_DIGIT};

transitions::=
   defining_transition_identifier: behavior_state_reference-
[trigger_behavior_reference, constraints_reference,
clock_resets_reference]->behavior_state_reference;

math_symbol_specification::=('>='|'<='|'>'|'<'|'=')
```

(b)

Fig. 4 Basic syntax of the physical component model: (a) physical component model library; (b) physical component model subclause

4 AADL-based particular risk analysis method

In this section, the PRA method based on AADL is proposed to assess the impact of particular risks on the HCPS. Fig. 5 describes how PRA will be conducted with an explanation of major steps.

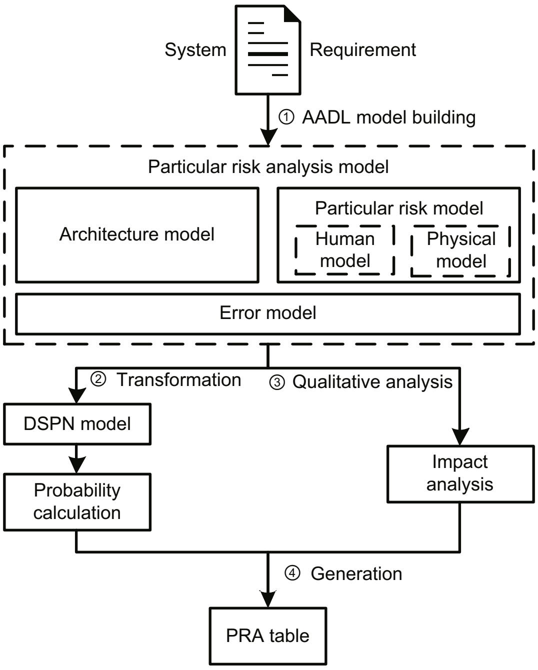


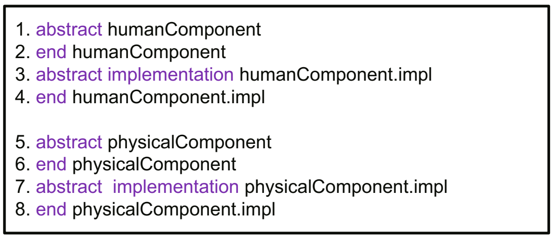
Fig. 5 Framework of the particular risk analysis method

4.1 Building a particular risk model

4.1.1 Architecture model

The architecture model is the basis of architecture-level PR analysis. In terms of a software requirement specification, the architecture model of the HCPS is designed, including software components, execution platform components, composite components, and their interactions using the AADL language.

AADL provides an annex extension method to support researchers in system modeling. The annex subclause in AADL needs to be defined in the component. However, the AADL core language does not support the extension of new components. Therefore, the definition of the abstract component is proposed in AADL. The abstract component is a general category used to describe the component types and implementations. Through this component, one can define a base view of the conceptual component in the system architecture and refine some general descriptions. To support AADL to define particular risk, we define the abstract components as two types, human component and physical component, as shown in Fig. 6.



```
abstract humanComponent
end humanComponent
abstract implementation humanComponent.impl
end humanComponent.impl

abstract physicalComponent
end physicalComponent
abstract  implementation physicalComponent.impl
end physicalComponent.impl
```

Fig. 6 Extension of abstract component definition

4.1.2 Human component model

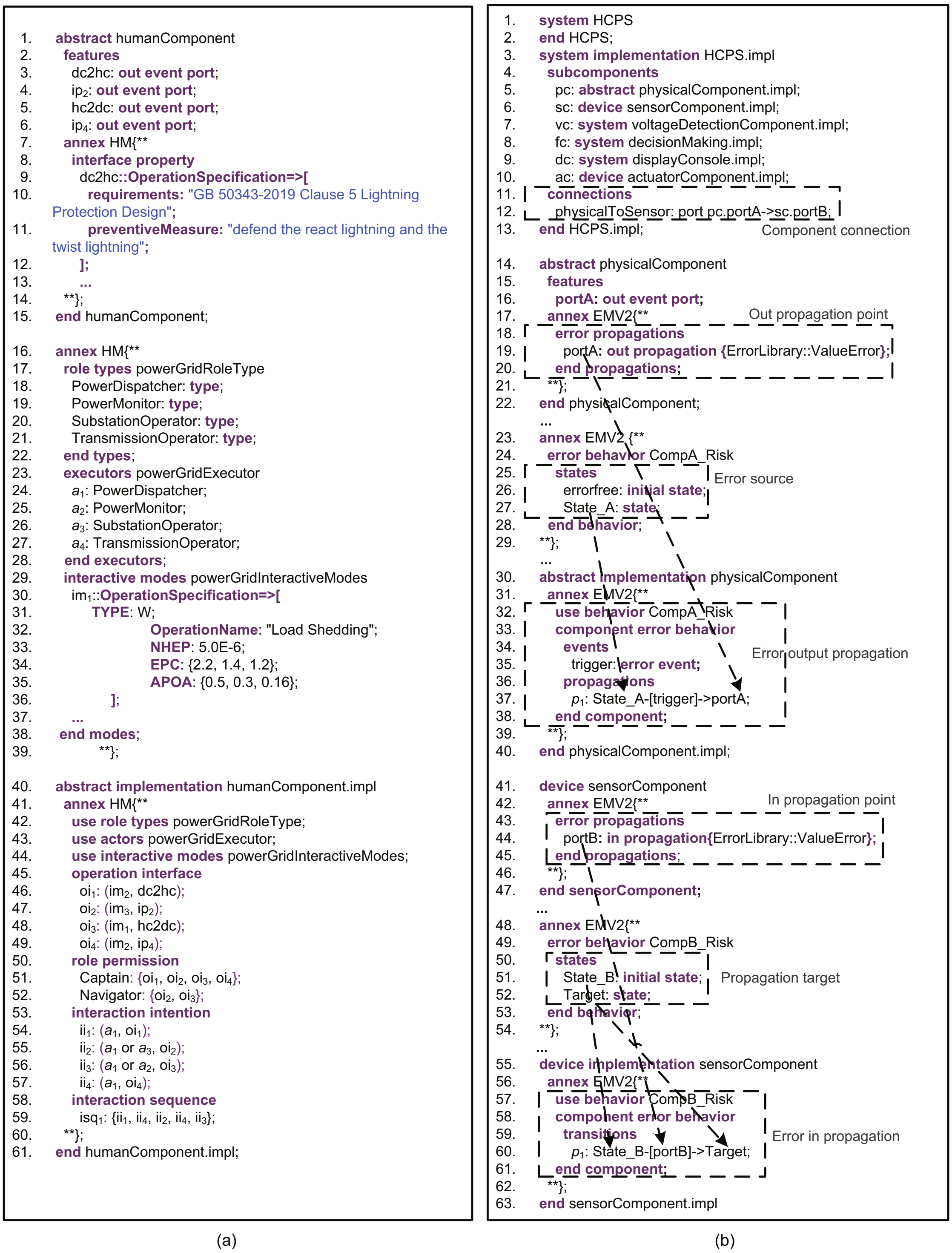
Human plays an important role in the operation of the HCPS. Human error is the main cause of large human-machine system accidents. According to human-machine operation specifications, engineers can build a human component model including actors, interaction modes, operation interfaces, role permissions, interaction intent, and interaction sequences using the human component model annex extended in this study. The human component model subclause template is shown in Fig. 7a.

In the human component model template, first, the role types, actors, and interactive modes need to be referenced from HMLib. Then, all operations for the interactive interface are defined in the operation interfaces to describe which operation interfaces exist in the system. The access permission given to each role is defined in the role permissions. After that, we use the defined actor set and operation interface set to describe the interaction intent. Finally, the execution order of the interaction intent is defined in the interaction sequence.

4.1.3 Physical component model

In the definition of particular risks, some physical phenomena such as gusts can cause system errors. Therefore, these risks need to be modeled for safety assessment in the early stage of system design. We define the physical component model subclause template as shown in Fig. 8.

Similarly, behavior states, physical variables, and clock variables need to be referenced from



```
abstract humanComponent
  features
    dc2hc: out event port;
    ip2: out event port;
    hc2dc: out event port;
    ip4: out event port;
  annex HM{**
    interface property
      dc2hc::OperationSpecification=>[
        requirements: "GB 50343-2019 Clause 5 Lightning Protection Design";
        preventiveMeasure: "defend the react lightning and the twist lightning";
      ];
      ...
  **};
end humanComponent;

annex HM{**
  role types powerGridRoleType
    PowerDispatcher: type;
    PowerMonitor: type;
    SubstationOperator: type;
    TransmissionOperator: type;
  end types;
  executors powerGridExecutor
    a1: PowerDispatcher;
    a2: PowerMonitor;
    a3: SubstationOperator;
    a4: TransmissionOperator;
  end executors;
  interactive modes powerGridInteractiveModes
    im1::OperationSpecification=>[
        TYPE: W;
                OperationName: "Load Shedding";
                NHEP: 5.0E-6;
                EPC: {2.2, 1.4, 1.2};
                APOA: {0.5, 0.3, 0.16};
          ];
    ...
 end modes;
        **};

abstract implementation humanComponent.impl
  annex HM{**
    use role types powerGridRoleType;
    use actors powerGridExecutor;
    use interactive modes powerGridInteractiveModes;
    operation interface
      oi1: (im2, dc2hc);
      oi2: (im3, ip2);
      oi3: (im1, hc2dc);
      oi4: (im2, ip4);
    role permission
      Captain: {oi1, oi2, oi3, oi4};
      Navigator: {oi2, oi3};
    interaction intention
      ii1: (a1, oi1);
      ii2: (a1 or a3, oi2);
      ii3: (a1 or a2, oi3);
      ii4: (a1, oi4);
    interaction sequence
      isq1: {ii1, ii4, ii2, ii4, ii3};
  **};
end humanComponent.impl;
```

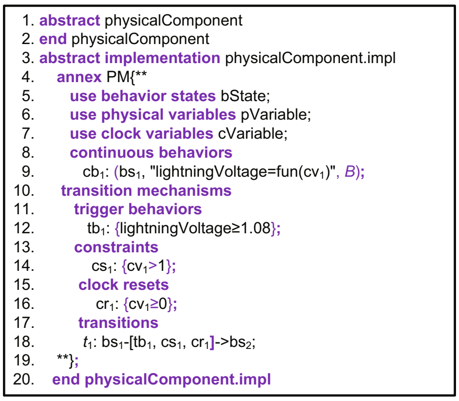
(a)

```
system HCPS
end HCPS;
system implementation HCPS.impl
  subcomponents
    pc: abstract physicalComponent.impl;
    sc: device sensorComponent.impl;
    vc: system voltageDetectionComponent.impl;
    fc: system decisionMaking.impl;
    dc: system displayConsole.impl;
    ac: device actuatorComponent.impl;
  connections
    physicalToSensor: port pc.portA->sc.portB;
end HCPS.impl;

abstract physicalComponent
  features
    portA: out event port;
  annex EMV2{**
    error propagations
      portA: out propagation {ErrorLibrary::ValueError};
    end propagations;
  **};
end physicalComponent;
...
annex EMV2 {**
  error behavior CompA_Risk
    states
      errorfree: initial state;
      State_A: state;
  end behavior;
**};
...
abstract implementation physicalComponent
  annex EMV2{**
    use behavior CompA_Risk
    component error behavior
      events
        trigger: error event;
      propagations
        p1: State_A-[trigger]->portA;
    end component;
  **};
end physicalComponent.impl;

device sensorComponent
  annex EMV2{**
    error propagations
      portB: in propagation{ErrorLibrary::ValueError};
    end propagations;
  **};
end sensorComponent;
...
annex EMV2{**
  error behavior CompB_Risk
    states
      State_B: initial state;
      Target: state;
  end behavior;
**};
...
device implementation sensorComponent
  annex EMV2{**
    use behavior CompB_Risk
    component error behavior
      transitions
        p1: State_B-[portB]->Target;
    end component;
  **};
end sensorComponent.impl
```

(b)

Fig. 7 An example of the PRA model: (a) human component model subclause template; (b) error propagation mechanism

PMLib. Then the continuous behaviors are defined inside each behavior state. After that, the transition mechanism between states will be defined as a triple, including trigger behaviors, behavior constraints, and clock resets. We use these behavioral states and transition mechanisms to form the transition module in the physical component model.



```
abstract physicalComponent
end physicalComponent
abstract implementation physicalComponent.impl
   annex PM{**
      use behavior states bState;
      use physical variables pVariable;
      use clock variables cVariable;
      continuous behaviors
         cb1: (bs1, "lightningVoltage=fun(cv1)", B);
    transition mechanisms
       trigger behaviors
          tb1: {lightningVoltage≥1.08};
       constraints
          cs1: {cv1>1};
        clock resets
           cr1: {cv1≥0};
        transitions
         t1: bs1-[tb1, cs1, cr1]->bs2;
   **};
  end physicalComponent.impl
```

Fig. 8 Physical component model subclause template

4.1.4 Constructing the PRA model

The architectural model is used mainly to describe the information system in the HCPS. The two proposed models are used mainly to describe the physical processes involved in the operation of the HCPS system. To assemble our extended model and the architectural model into the PRA model, the error model will be used as a connection point. EMV2 is used to design the error state, error event, transition, and error propagation to describe the occurrence of the fault, fault behavior, error propagation, and failure behavior, respectively.

1. Building error model between components. The safety requirement specification contains the requirements for system safety that must be satisfied by the HCPS. From this document, architects need to determine the possible risk event (RE), error event (EE), repair event (R), error states, and other information in the system. RE, EE, and R can be used as transition conditions between states. RE refers to the event that may occur in the human component model and physical component model, while EE refers to the event that may occur in the computing system. R has different definitions for different components. In the computing system, it usually refers to event (R_c) that immediately returns to a normal state after a transient fault occurs. In the human component model, it usually refers to the repair event (R_h) of repair personnel after the system fails. In the physical model, it usually refers to the autonomous recovery event (R_p) of the natural environment. At the same time, it is necessary to determine the possible transition according to the relationship between the error state and the error event. An error behavior transition specifies the transition from a source state to a target state if a transition condition is satisfied. To consider the impact of PRM on system safety, the occurrence probability of error events will be determined by the physical process defined in the PRM. In the human component model, the error producing conditions (EPCs), NHEP, and other information are defined as the benchmarks to determine the probability of error events. The severity level defined in the physical component model is used as a reference for the probability of error events. At the same time, it is necessary to consider the interaction intention and interaction sequence in the human factor model when establishing an error model. Interaction intent and interaction sequence describe mainly the interaction behavior of the executor, the interaction intention describes the possible operations of the interaction subject (actor), and the interaction sequence refers to the sequence of multiple interaction intentions. Some particular risk events that may occur in these interactions will cause the system to fail (such as a certain action performed by an actor without operation authority, or an incorrect operation sequence of an interactive intention). Therefore, we need to analyze unreasonable interaction intentions and interaction sequences, and model them as error events and error states in the error model.

2. Building error propagation between components. After the error model is defined within each component, these independent error models need to be connected. As shown in Fig. 7b, first, the physical connection needs to be established through the port defined in the architecture model, and then the port is referred to as the error propagation point in the error model. When the error state inside the component is triggered by an error event, it will propagate outward from the defined error propagation point, and some errors will also propagate from the error propagation point to the component.

3. Instantiating the system. The model built above is called a declarative model. To be able to analyze the model, the declarative model needs to be instantiated. The system instance model represents a runtime physical system generated by a declarative model. This instantiation process will analyze the dependency relationship in the declarative model, and assemble the independent and discrete components in the illustrative model to form the final systematic model. It can clearly indicate the various runtime conditions needed to make up

the system elements.

Finally, these models can be integrated into a PRA model by annotating the architectural model, human component model, and physical component model with the error model.

4.2 Impact analysis

The purpose of PRA analysis is to analyze the impact of external events that may cause serious or catastrophic damage to the system. Therefore, what particular risks occur and how they affect the HCPS should be identified. To answer these questions, based on the complete PRA analysis model, our methodology performs qualitative analysis to determine the particular risks, failure states, operational requirement, and the propagation path of risks.

We can extract the particular risks, failure states, and operational requirement from the complete PRA model. In addition, the risk propagation path can be determined by the error propagation mechanism defined in EMV2. For example, as shown in Fig. 7b, when State_A and trigger condition in the source component are satisfied, ValueError will be propagated out through portA. Then, if the destination component is in State_B, this transition will occur or the target will propagate the error out. Otherwise, the ValueError that has been propagated out will disappear.

4.3 Transforming the PRA model to the DSPN model

Traditional PRA analysis is usually a qualitative analysis method, so we add the quantitative analysis process to the traditional method. However, it is difficult to directly perform quantitative assessment for the AADL model. To solve this problem, the PRA model is converted into the DSPN model based on our previous work (Wei et al., 2014, 2018).

The model conversion process is mainly to convert the error model established in the system into a DSPN model. The functions of the extended human factor model and the physical model are similar to that of the AADL architecture. They are used only to guide the establishment of a complete error model. In addition, in the conversion process, for the definition of the probability value of the error event, the parameters defined in the human factor model and the physical model will be considered. Therefore, for the model transformation mainly the error model of the system is considered. The human component model and physical component model are used only to guide the establishment of the error model considering particular risk factors. The transformation process is mainly to extract the operating states, events (risk events, error events, and repair events), state transition, and error inPropagation/outPropagation existing in the PRA model, and then to perform model conversion according to the mapping rules defined in Table 1. The error state is mapped to place in DSPN, that is, "error state→place." The normal operation or initial state of the component is mapped to place with token, that is, "errorfree state→place with token." According to the probability distribution types, events can be divided into three types: position, fixed, and latency. These three types are mapped to exponential transition, immediate transition, and deterministic transition, respectively, that is, "events (position)→exponential transition," "events (fixed)→immediate transition," and "events (latency)→deterministic transition."

In addition, the error outPropagation and error inPropagation are mapped to place and immediate transition, respectively. The error outPropagation is the propagation point of the component that propagates errors outwards. When a certain event or error inPropagation is satisfied, the current state of the component will propagate the error from this

Table 1 Mapping rules between PRA and DSPN models

PRA model	DSPN model
Event (poisson)	Exponential transition ▯
Event (fixed)/Error inPropagation	Immediate transition ▮
Event (latency)	Deterministic transition ■
Error state/Error outPropagatio	Place ○
Errorfree state/Initial state	Place with token ⊙
State transition	Source place→transition ⊙→▮→○ transition→ destination place

propagation point; thus, error outPropagation is also mapped to place here. The error inPropagation is the propagation point of the component that propagates inward errors. When the error is passed in from error inPropagation, it may cause the component to transfer from the normal operating state to the error state. Therefore, the error inPropagation is mapped to the immediate transition in DSPN. The transition between states in the PRA model describes that when an event occurs, the state of the component will transfer from the source state to the destination state. Therefore, it will be mapped into two parts in the DSPN model. As shown in Table 1, a connection arc is established between the source place and the transition, and the other connection arc is established between the transition and the destination place.

4.4 Quantitative analysis

After converting the PRA model to a DSPN model, it is necessary to evaluate the impact of particular risk events based on the DSPN model. The transformation process also needs to consider the impact of the human physical component and physical component on system safety. Therefore, it is necessary to calculate the probability of risk events in the human physical component and physical component based on the information in the component.

HEART (Gertman and Blackman, 1994; Kirwan et al., 1997) considers the influence of human-machine interaction, task type, and environment on the human, so it can identify the task type and EPC and evaluate the probability of risk events. Therefore, this method is applied to human component model analysis including three steps: (1) task classification. The HEART method classifies the tasks into nine generic task types and gives the human error probability (HEP) for each task type. However, the general method is not applicable to some specific fields. In this case, domain experts need to divide the task types of the actors according to the actual situations, and provide basic data of NHEP for these task types. (2) Determining error producing conditions. Based on task classification, EPCs are used to adjust the human error probability in the HEART method, where the EPCs include operational experience, time pressure, and external conditions. Each EPC is assigned a weight value w_{EPC_i} to reflect the maximal impact of each EPC. (3) Probability calculation method. For a specific task, the impact degree of each EPC is different. Therefore, the HEART method determines the actual impact value of each EPC by means of expert scoring, records it as APOA_i, and uses Eq. (1) to correct the weight $w^{'}_{\mathrm{EPC}_i}$. Finally, HEP is obtained according to Eq. (2).

$$w^{'}_{\mathrm{EPC}_i} = (w_{\mathrm{EPC}_i} - 1) \cdot \mathrm{APOA}_i + 1, \tag{1}$$

$$\mathrm{HEP} = \mathrm{NHEP} \cdot \prod_{i=1}^{n} w^{'}_{\mathrm{EPC}_i}. \tag{2}$$

There are multiple behavior states in the physical component model, each of which has a continuous behavior changing with time. At the same time, each behavior state will trigger some error events with different probabilities, and these error events further cause the state to change. Therefore, when modeling the error model of the physical component, we need to consider the behavior severity level of the physical component model, as shown in Fig. 8, and use this as a basis to define the probability of the error event.

The tool TimeNet can analyze the structure and accessibility of an SPN model, which is suitable for the verification and performance analysis of the SPN model. TimeNet 4.0 provides the transient analysis and stability analysis of the SPN model. We use TimeNet mainly to calculate the failure probability of the system with a PRA model. The obtained probability is used as a quantitative evaluation benchmark for PRA analysis.

4.5 Presentation of PRA assessment

The final PRA table is generated based on the qualitative and quantitative analysis, as shown in Table 2. The PRA table is composed of eight parts of assessment information, i.e., name of component (Comp) where the particular risk event occurs, particular risk (PR) event, failure state (FS) of component where the particular risk event occurs, name of component (AC) affected by the particular risk event, operating requirement (OR) when particular risk event occurs, preventive measure (PM) for particular risk events, failure probability (Pro) of the affected component, and acceptance (isAccept) of the analysis results. Comp, PR, FS, OR, and PM can be extracted from the PRA model. AC is obtained by analyzing the connection between components and the error propagation direction. Pro

Table 2 PRA analysis table

Comp	PR	FS	AC	OR
physicalcomp	lightningStrike	groundFault	voltage_sensor	GB 50343-2019 Clause 5 lightning protection design
humancomp	errorInteraction	errorOperation	dc_write	State professional standard power dispatcher

Comp	PM	Pro	isAccept
physicalcomp	Defending the react lightning and twist lightning	1.11×10^{-4}	Unacceptable
humancomp	Technical training and theoretical examination	7.47×10^{-5}	Unacceptable

Comp: component; PR: particular risk; FS: failure state; AC: affected component; OR: operational requirement; PM: preventive measure; Pro: probability

is obtained by simulating and calculating the converted DSPN model. isAccept refers to whether the conclusion of the analysis is accepted according to the corresponding standards. If not, the architects need to modify the system model.

5 Case study

The PRM method and model transformation rules proposed in this study have been implemented as eclipse plug-ins. In Section 5.1, we will describe the tool implementation structure diagram. In Sections 5.2–5.4, we will select a power grid system to illustrate the feasibility of the proposed method. The safety and stability control (SSC) system in the power grid system is usually a three-level control structure composed of master station, sub-station, and executive station systems. The SSC ensures the safe and stable operation of the power system and effectively avoids the occurrence of power outage through a complete stability control strategy. However, hidden faults in the SSC, such as their own defects or human errors, affect the normal operation of the power system. Therefore, we refine the SSC architecture and take the executive station system as the specific analysis object. We use the proposed method to model and analyze the system, and obtain the final analysis results.

5.1 PRA prototype tool architecture

The implementation of a particular risk modeling and analysis tool with AADL is divided into three layers, as shown in Fig. 9. The bottom layer is the infrastructure of the tool. This is developed in the eclipse-integrated development environment. The middle layer focuses on PRA modeling, which is an AADL core language extension to provide particular risk modeling, text parsing, and system instantiation. The upper layer is the particular risk analysis function of the AADL model. With the AADL instance model as input, it can analyze particular risk events in the system, calculate the probability of error events caused by the human factor model according to the HEART method, and convert the AADL instantiation model into a DSPN model. To calculate the failure probability of subcomponents in the AADL model, the PRA tool is integrated with an existing calculation tool TimeNet, which can be used in failure probability calculation. Therefore, the final AADL model PRA prototype tool can complete functions such as particular risk modeling, model transformation, failure probability calculation, and PRA evaluation.

5.2 Building a particular risk analysis model

5.2.1 Architecture model of SSC

The architecture of SSC involves mainly the processor unit, memory unit, bus, voltage sensor, voltage variation detection (VVD_comp), display and console (DC_comp), decision-making control (DMC_comp), and actuator components (Fig. 10). VVD_comp, DC_comp, and DMC_comp are bounded together with the processor unit and memory unit. The processor, memory, sensors, actuators, and devices all communicate through the bus. The voltage sensor collects voltage signals from the power grid system and physical environment. VVD_comp detects whether there is an abnormal voltage, DMC_comp processes the data and issues command, DC_comp displays the information and processes the console information, and the actuator component executes commands such as cutting the machine for the power grid system.

The human component model describes how the power dispatcher reads the information from DC_comp and issues the control command to DC_comp. The behavior of generating error control instructions is defined as errorInteraction. According to the HEP of generic task types defined in the HEART method, the NHEP of errorInteraction is set to 5.0×10^{-6}, and three kinds of EPC are defined in this case, which are poor feedback, insufficient experience, and emotional stress. The weights of these three types of error-inducing conditions are set to 2.2, 1.4, and 1.2, and values of APOA of the three types of error-inducing conditions are set to 0.50, 0.30, and 0.16.

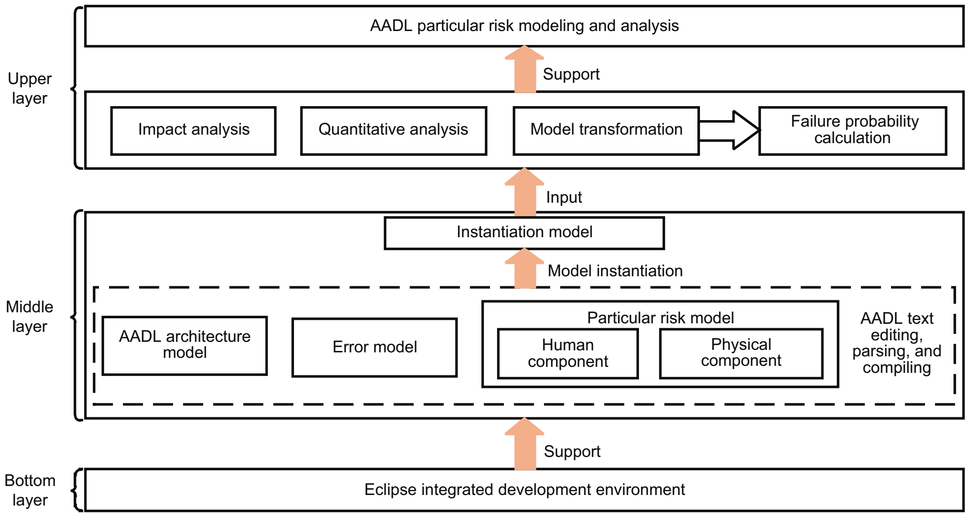


Fig. 9 Architecture of the PRA prototype tool

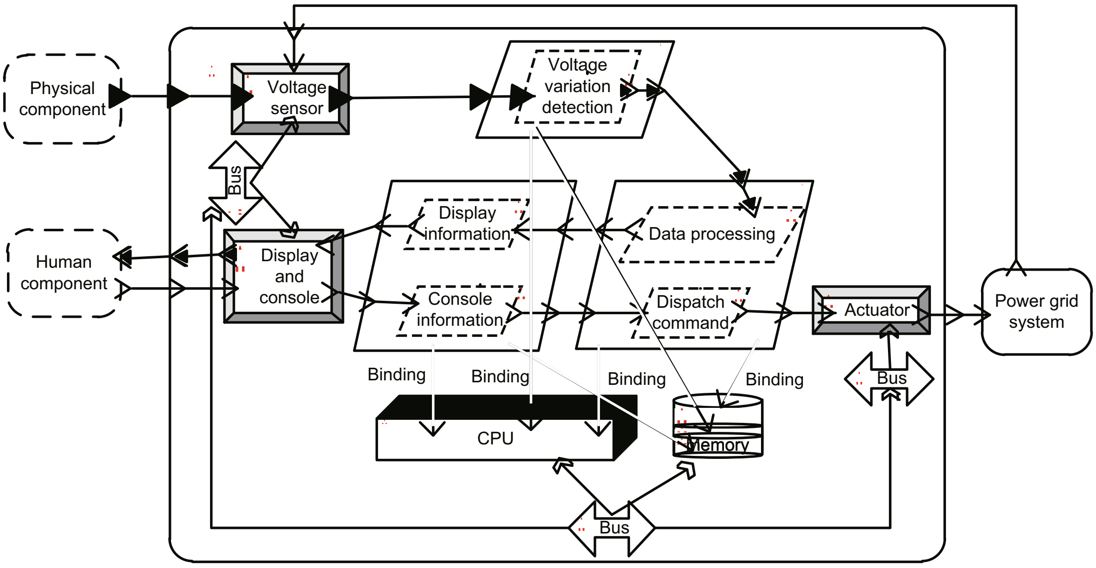


Fig. 10 Architecture model of SSC

Solid triangles on the boundary of the component: in/out data ports; hollow arrows on the boundary of the component: in/out event ports

In this case, lightningStrike is a common external physical excitation. The voltage of the lightning is considered as an important variable in the physical component. The voltage value can be set as a function of time, i.e., lightningVoltage = fun(cv_1), where cv_1 is the clock variable in the physical environment. At the same time, in the whole HCPS system, there is a boundary constraint on the voltage value of lightning, i.e., lightningVoltage$\geq 1.0 \times 10^8$. In addition, there is a constraint on clock variables, i.e., $\mathrm{cv}_1 > 1$. When these two constraints are satisfied, the state of the physical model will change (i.e., bs_1-[$\mathrm{tb}_1, \mathrm{cs}_1, \mathrm{cr}_1$]->$\mathrm{bs}_2$), and the clock may be reset (i.e., $\mathrm{cv}_1 \geq 0$).

5.2.2 Error model

Each component has more than one error state to represent its running condition. State O represents the non-fault state of the system and is the initial state. As shown in Fig. 11, a risk event such as lightningStrike may occur in the physical component, which will affect the state of the physical environment. The state of the environment component may change to the abnormal state F (single-phase-to-ground-fault). The voltage sensor component may transfer from the initial state to state F when receiving external stimulation. After receiving the abnormal information, VVD_comp calculates and judges which faults have occurred, and sends the fault information to DMC_comp. DMC_comp processes the fault information, sends it to DC_comp, and displays it to the human component, as shown in the green path in Fig. 11.

Then, the power dispatcher in the human component model reads the information from the display, and the dispatcher may transfer from the normal state O to the abnormal state F1 (nervous state), and may make a wrong judgment and issue a wrong control command to DMC_comp through DC_comp. Finally, the instruction controls the actuator component to make the wrong generator trip, as shown in the blue path in Fig. 11.

The components in SSC communicate with other components through ports. Through connections between the components, the error state

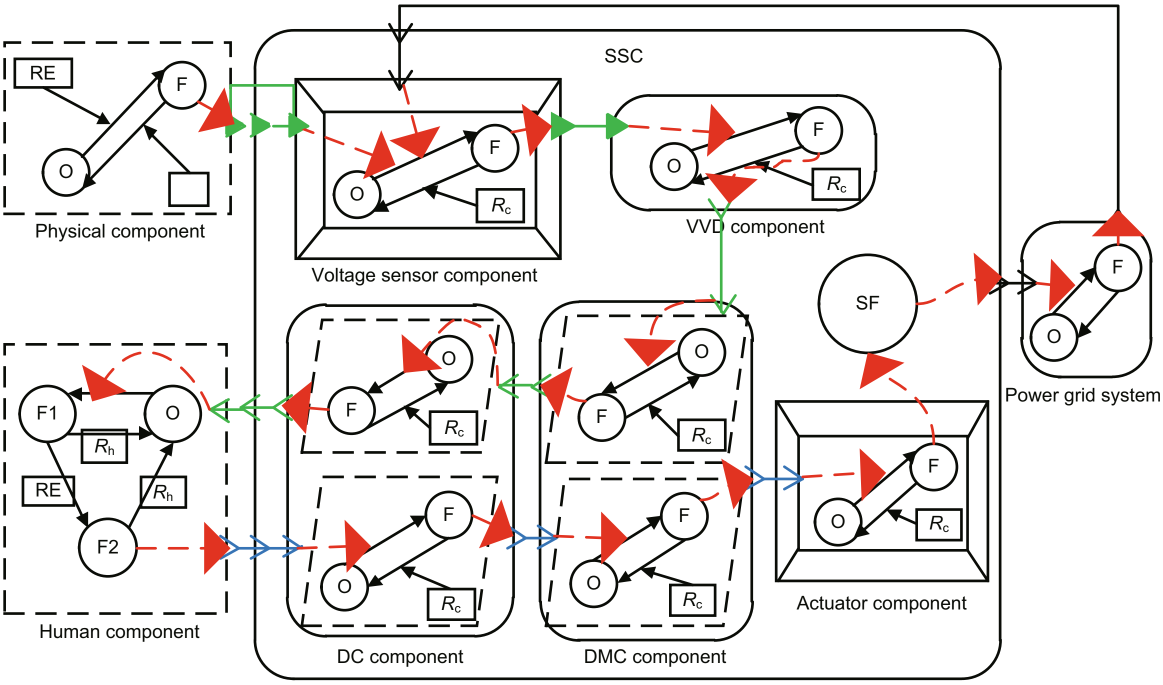


Fig. 11 PRA model of SSC

DC: display and console; VVD: voltage variation detection; DMC: decision-making control; RE: risk event; SF: system failure. R_p: repair event in the physical component; R_h: repair event in the human component; R_c: repair event in the computing component. Red dotted arrow: error inPropagation/outPropagation; solid triangle on the boundary of the component: in/out data port; hollow arrow on the boundary of the component: in/out event port. References to color refer to the online version of this figure machine of each component in SSC can be connected. The physical component affects the operation of the system through a voltage sensor, and human factor components can interact with the system through displays and console devices. Therefore, the SSC, human component, and physical component can be integrated into a complete PRA model.

5.3 Model transformation and quantitative calculation

Model conversion converts mainly the error model in the software system, converts the error state machine of each component into an element in the DSPN, and links the independent DSPN models according to the error propagation between the components. In addition, the error model in the human component model and physical component model need to be converted. It needs to consider the impact of human component model and physical component model on the safety of the system. For the human component model, according to Eqs. (1) and (2) defined in the HEART method, we use the parameters NHEP, EPC, and APOA that we defined in the human component model to calculate the probability of occurrence of risk events due to human factors. The calculation result of HEP is 9.246 72E-5. Then, the architecture engineer uses the severity level of continuous behavior defined in the physical component model as the basis for determining the probability of risk events.

According to the conversion rules between AADL and DSPN models, the established AADL PRM is converted into a DSPN model, as shown in Fig. 12. The error state machine of each component in AADL is transformed into an independent DSPN model, and then all DSPN models are linked according to the error propagation through the interaction interface between components. To obtain the probability of system failure caused by particular risk events, it is necessary to calculate the probability value that the number of tokens in place is 1. Therefore, according to the rules of TimeNet, the calculation expressions for each particular risk occurrence probability are expressed as $P\{\#\text{PlaceName}=1\}$. The parameter value needs to be set before simulation and calculation. The confidence level (the parameter represents the confidence level of the probability value) is set to 99%, the maximum relative error (the parameter represents the precision required for the simulation) is set to 5%, and the permitted difference for probability measures (the parameter can specify a small allowable difference, which can improve the estimation accuracy, but at the same time increase the simulation running time) is set to 5%.

5.4 Experimental results

From the PRA analysis method, the generated PRA table is shown in Table 2. Comp, PR, FS, OR, and PM can be extracted from the PRA model. For example, in the first analysis result, the event lightningStrike is the particular risk event having occurred in the physical component. It changes the physical component from the state errorfree to state groundFault. By analyzing the error propagation path between components, the affected component is the voltage sensor component. The operational requirement for the risk is “GB 50343-2019 Clause 5 Lightning Protection Design” (China Institute of Building Standard Design & Research, 2012), the preventive measure is “defending the react lightning and twist lightning,” the failure probability of the voltage sensor is 1.11E-4, the failure probability of the system affected by lightning strike is unacceptable, and the system design needs to be changed.

We also compare the failure probability of system components with and without the human component model and physical component model. Table 3 shows that when the human component model and physical component model are considered, the failure probability of the system components increases significantly. This is because these two

Table 3 Comparative analysis

Component	Failure state	Failure probability	
		With PRM	Without PRM
physicalcomp	groundFault	5.26×10^{-5}	
vol_sensor	errorSignal	1.11×10^{-4}	5.59×10^{-6}
vvdcomp	errorCmd	1.93×10^{-5}	9.70×10^{-5}
dmc_collect	errorInfo	6.11×10^{-5}	3.13×10^{-6}
dc_read	infoAbnormal	5.81×10^{-5}	2.87×10^{-6}
humancomp	nervous	5.26×10^{-5}	
	errorOperation	6.60×10^{-8}	
dc_write	writeAbnormal	7.47×10^{-5}	3.05×10^{-6}
actuator	errorAction	6.68×10^{-5}	3.30×10^{-6}
hcps	systemFailure	6.68×10^{-5}	3.30×10^{-6}

With PRM: with human component model and physical component model; Without PRM: without human component model and physical component model

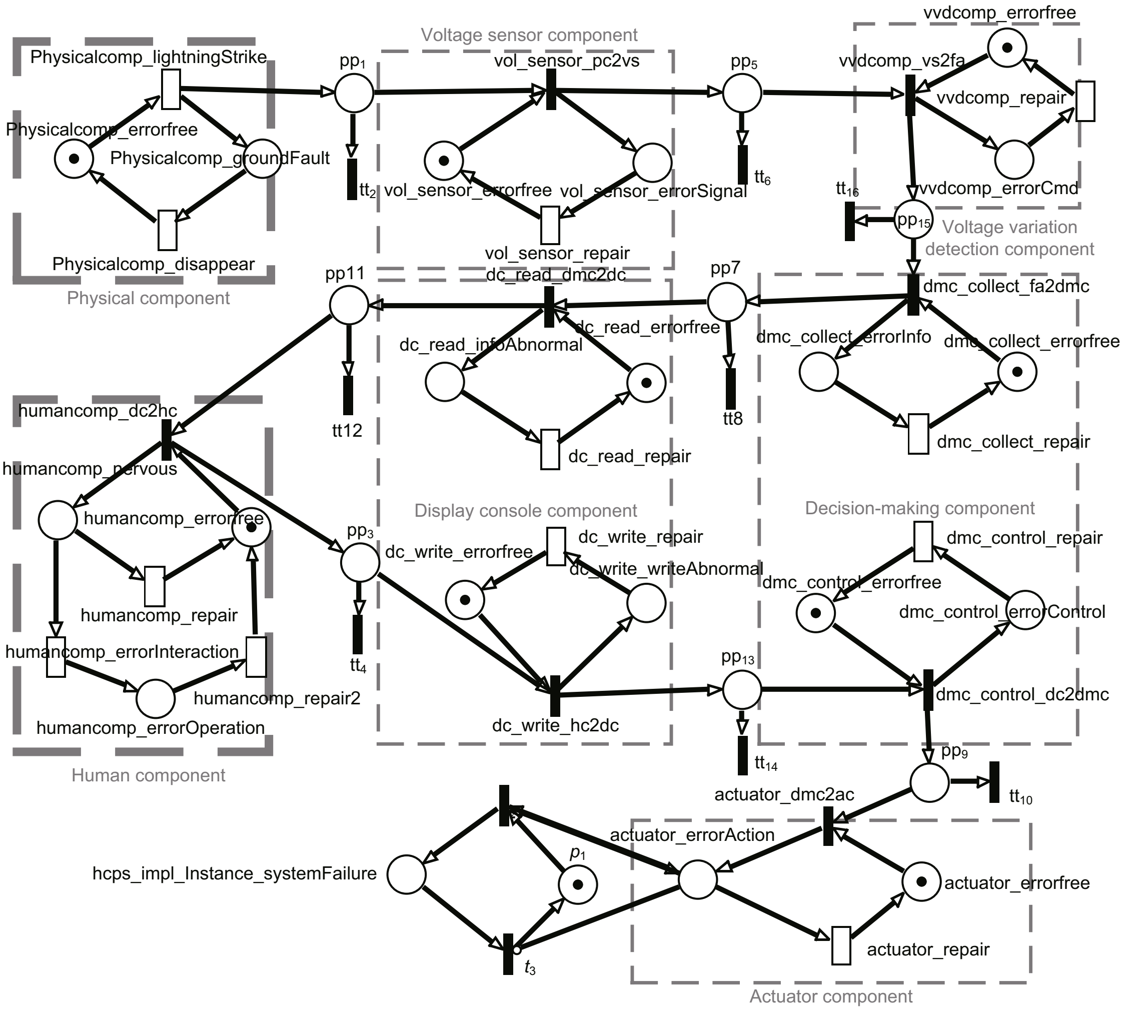


Fig. 12 DSPN model of the PRA model

models introduce risk events which increase the probability of system components changing from normal to failure. Table 3 also reflects the necessity of considering PRM when evaluating the safety of the system.

In summary, by modeling the particular risk event of the system in detail, it makes up for the deficiency in traditional PRA. Through quantitative assessment, it provides data support for PRA, which is more objective and persuasive than traditional PRA.

6 Conclusions and future work

In this study, we proposed a particular risk model (PRM) to improve the modeling capability of the Architecture Analysis and Design Language (AADL). The PRM makes up for the weaknesses of AADL in describing particular risks, taking into account human factors and physical events during the operation of the system. The PRM also enables detailed characterization of actor activities, modeling physical phenomena, and behaviors outside the system. In addition, an architecture-level particular risk analysis (PRA) methodology was proposed based on the PRM constructed by the architecture, human component, physical component, and error models. First, our methodology extracted risk-related information and performed impact analysis to determine the error propagation. Then, to guarantee that the probability calculated based on the DSPN model is entirely consistent with that based on the PRA model, we converted the PRA model into a DSPN model. Next, we used the tool TimeNet to calculate the failure probability of each component. This reduced the bias caused by man's will in the qualitative analysis method. Finally, by modeling and analyzing a specific case, the method gave specific analysis results and an accurate data basis. These results showed that it is necessary to carry out PRA in the early design phase of system development.

In the future, our work will focus on in-depth study of safety analysis methods for human-cyber-physical systems, and accumulate and draw on relevant data to establish a more reasonable PRM. In addition, we will conduct PRA for the current mainstream distributed software systems to verify the effectiveness of our proposed method.

Contributors

Ming-rui XIAO and Yun-wei DONG designed the research. Qian-wen GOU, Feng XUE, and Yong-hua CHEN processed the data. Ming-rui XIAO drafted the manuscript. Yun-wei DONG helped organize the manuscript. Ming-rui XIAO and Yun-wei DONG revised and finalized the paper.

Compliance with ethics guidelines

Ming-rui XIAO, Yun-wei DONG, Qian-wen GOU, Feng XUE, and Yong-hua CHEN declare that they have no conflict of interest.

References

Banerjee A, Kandula S, Mukherjee T, et al., 2012. BAND-AiDe: a tool for cyber-physical oriented analysis and design of body area networks and devices. *ACM Trans Embed Comput Syst*, 11(S2):49-77. https://doi.org/10.1145/2331147.2331159

Bi SY, 2017. Research on Tire Burst Safety Analysis Technology of Transport Category Aircraft. MS Thesis, Nanjing University of Aeronautics and Astronautics, Nanjing, China (in Chinese).

China Institute of Building Standard Design & Research, 2012. Technical Code for Protection of Building Electronic Information System Against Lightning. GB 50343-2012. National Standards of People's Republic of China (in Chinese).

Delange J, Feiler P, 2014. Architecture fault modeling with the AADL error-model annex. Proc 40th EUROMICRO Conf on Software Engineering and Advanced Applications, p.361-368. https://doi.org/10.1109/SEAA.2014.20

Dong YW, Wang GR, Zhang F, et al., 2011. Reliability analysis and assessment tool for AADL model. *J Softw*, 22(6):1252-1266 (in Chinese). https://doi.org/10.3724/SP.J.1001.2011.04014

Gertman DI, Blackman HS, 1994. Human Reliability and Safety Analysis Data Handbook. Wiley-Interscience, New York, USA.

Ji Z, Zhou YH, Wang BC, et al., 2019. Human-cyber-physical systems (HCPSs) in the context of new-generation intelligent manufacturing. *Engineering*, 5(4):624-636. https://doi.org/10.1016/j.eng.2019.07.015

Kim J, Jung W, Jang SC, et al., 2006. A case study for the selection of a railway human reliability analysis method. Proc Int Railway Safety Conf, p.22-27 (in Korean).

Kirwan B, Kennedy R, Taylor-Adams S, et al., 1997. The validation of three human reliability quantification techniques—THERP, HEART and JHEDI: part II—results of validation exercise. *Appl Ergon*, 28(1):17-25. https://doi.org/10.1016/S0003-6870(96)00045-2

Luo XL, 2017. Human Factors in Flight (3rd Ed.). Southwest Jiaotong University Press, Chengdu, China (in Chinese).

Marsan MA, Chiola G, 1987. On Petri nets with deterministic and exponentially distributed firing times. In: Rozenberg G (Ed.), Advances in Petri Nets 1987. Springer-Verlag Berlin Heidelberg, p.132-145. https://doi.org/10.1007/3-540-18086-9_23

Reason J, 1990. Human Error. Cambridge University Press, New York, USA. https://doi.org/10.1017/CBO9781139062367

Society of Automotive Engineers, 1996. Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment, ARP4761. National Standards of the United States of America.

Society of Automotive Engineers, 2013. Architecture Analysis and Design Language (AADL) Annex Volume 3: Annex E: Error Model Annex.

Society of Automotive Engineers, 2017. Architecture Analysis & Design Language (AADL) AS5506C.

Wang Q, Li X, Li S, et al., 2017. Risks and risk control of wind power enterprises. 13th Int Conf on Natural Computation, Fuzzy Systems and Knowledge Discovery, p.3070-3075. https://doi.org/10.1109/FSKD.2017.8393275

Wei XM, Dong YW, Yang MM, et al., 2014. Hazard analysis for AADL model. Proc IEEE 20th Int Conf on Embedded and Real-Time Computing Systems and Applications, p.1-10.

Wei XM, Dong YW, Li XL, et al., 2018. Architecture-level hazard analysis using AADL. *J Syst Softw*, 137:580-604. https://doi.org/10.1016/j.jss.2017.06.018

Wei XM, Dong YW, Sun PP, et al., 2019. Safety analysis of AADL models for grid cyber-physical systems via model checking of stochastic games. *Electronics*, 8(2):212. https://doi.org/10.3390/electronics8020212

Zimmermann A, 2017. Modelling and performance evaluation with TimeNet 4.4. In: Bertrand N, Bortolussi L (Eds.), Quantitative Evaluation of Systems. 14th Int Conf on Quantitative Evaluation of Systems, p.1-4.

Zou Y, 2015. Research on Fault Probability Model of Overhead Power Transmission Line Based on Environmental Factors. MS Thesis, Huazhong University of Science and Technology, Wuhan, China (in Chinese).