Reliability Modeling of Fault Propagation in Software-Hardware Coupled Intelligent Ship Systems Using Petri Nets and Embedded Dynamic Bayesian Networks

Xiaofang Luo*[1], Linghui Guo[1], Xiangdong Ma[2*], Xu Bai[3], Jingling Li[3]

1. School of Economics and Management, Jiangsu University of Science and Technology, Zhenjiang 212100, China

2. School of Engineering, Lancaster University, Lancaster, UK

3. School of Naval Architecture & Ocean Engineering, Jiangsu University of Science and Technology, Zhenjiang 212100, China

Abstract: Maritime autonomous surface ships are increasingly recognized as complex cyber-physical systems, where autonomous navigation depends on tightly coupled software and hardware components. Conventional reliability assessment approaches often assume independence between software and hardware, which limits their accuracy in capturing fault propagation mechanism. This paper proposes a novel reliability modeling framework that explicitly incorporates software-hardware coupling. First, Fault propagation paths are represented using Petri Nets (PN) to capture dynamic interactions among components. The PN structure is then transformed into an embedded Dynamic Bayesian Network (DBN), enabling dynamic probabilistic reasoning over fault dependencies. By integrating PNs with DBNs, a coupled reliability model is constructed for intelligent ship autonomous navigation systems. A case study on an autonomous navigation task demonstrates the effectiveness of the proposed method in quantifying system reliability, identifying critical software and hardware components, and highlighting fault propagation effects that are overlooked in a decoupled model. The results confirm that the proposed approach enhances both the accuracy and interpretability of reliability assessment for intelligent ship systems.

Keywords: Maritime autonomous surface ship; Fault propagation; Software-hardware dependency; Cyber-physical system; Petri Net; Dynamic Bayesian Network; Autonomous navigation; System reliability

*Corresponding author: luoxiaofang@just.edu.cn; xiandong.ma@lancaster.ac.uk.

1. Introduction

With the rapid development of artificial intelligence, cloud computing, and the Internet of Things, the maritime industry is undergoing accelerated digital transformation. As a representative of this transformation, Maritime Autonomous Surface Ships (MASSs) have emerged as a global research hotspot and a key focus in maritime innovation. Equipped with advanced automation, MASSs can support human operators or even achieve fully autonomous decision-making and control [1]. Their autonomous navigation capabilities promise significant benefits, including improved transportation efficiency, reduced energy consumption and emissions, and lower operational costs. However, the high degree of automation and technological integration also introduces new safety challenges. Intelligent ships rely on complex Cyber-Physical Systems(CPSs), where the malfunction of critical components may cause task-level or even system-wide failures [2]. Existing studies have shown that inadequate robustness of the perception system can lead to environmental sensing errors [3], complex task decision-making logic increases the risk of software failures [4], and hardware components unreliability may directly lead to navigation failures [5]. As a result, reliability analysis of autonomous navigation systems has become a central issue in ensuring the safety of intelligent ships, providing theoretical foundations for their design and operation.

Intelligent ship systems are composed of numerous hardware and software modules. Hardware includes sensors and actuators, while software encompasses algorithms for perception, decision-making, and control. During mission execution, these components exhibit strong coupling relationships through physical connections, software deployment, and information exchange. Although extensive research has been conducted on hardware reliability models [6][7]and software reliability models [8][9], most existing approaches assume independence between software and hardware [10]. This assumption overlooks the fact that failures frequently propagate across hardware and software boundaries. For example, hardware anomalies can trigger software

malfunctions, and conversely, abnormal software outputs may damage hardware components. A study conducted at Stanford University on the Multiple Virtual Storage/System Product (MVS/SP) operating system found that approximately 35% of all observed software failures were related to hardware issues [11]. Furthermore, software subsystems are often embedded within hardware platforms, and their outputs essentially consist of a series of alternating signals. Changes in these signals can result in voltage and current fluctuations in hardware components, eventually leading to the physical damage. Studies have shown that considering system fault dependencies yields higher reliability estimates than models that assume independence [12]. Traditional reliability assessment methods struggle to address the complex failure modes and mutual interactions arising from the deep coupling of hardware, software, data, and networks in intelligent ship systems. Therefore, developing a reliability evaluation approach at system level that accounts for software-hardware coupling is essential to improving the design, validation, and maintenance capabilities of intelligent ships.

Recognizing this challenge, researchers have proposed various approaches to model software–hardware interactions. Varuvel et al.[13] used continuous-time Markov chains to quantify the impact of hardware degradation on software execution. Andrade et al.[14] analyzed the influence of hardware-related events on real-time programs using temporal measurement and machine learning. Zhu et al.[15] classified failures into hardware, software, and interaction-induced failures, and established a unified system reliability model based on Markov chains. Zheng et al.[16] applied copula functions to characterize dependencies between software and hardware failures and established a corresponding system reliability model. However, this approach did not account for the impact of system architectural design and relied heavily on historical data for parameter calibration, which may introduce model bias in emerging systems or rare failure scenarios due to insufficient samples. Joha et al.[17] considered the interaction between system hardware and software, and proposed a reliability modeling and analysis approach for multi-hardware-software systems. Huang et al. [18] proposed

a software-hardware coupling Failure Modes and Effects Analysis (FMEA) method, in which a failure mode coupling matrix and a corresponding calculation model were established to evaluate the Risk Priority Number (RPN). Zeng et al.[19] introduced a path-integral-based analysis method that considers the interaction between software and hardware during system performance degradation and failure, enabling reliability assessment of non-repairable co-designed software-hardware systems. Seo et al.[20] proposed an importance-guided modeling approach to evaluate the reliability of safety-critical software in integrated software-hardware systems. However, this method did not consider the correlation between software and hardware, nor did it assess the reliability of those less critical components. Stewart et al. [21] extended Architecture Analysis and Design Language (AADL) to model component failure behaviors using formal methods for the safety assessment of complex software-hardware systems. Although these methods advanced the field, they often remain at the macro level, relying heavily on historical data, overlooking system architecture, or neglecting detailed modeling of software-hardware dependencies. Consequently, a comprehensive framework for system-level reliability modeling under software–hardware coupling is still lacking.

In the context of intelligent ships, this gap is particularly critical. Their autonomous navigation systems integrate perception, localization, prediction, decision-making, and control modules with diverse hardware platforms, leading to deep cyber-physical coupling. Traditional reliability techniques, such as Bayesian Network (BN) [22], Fault Tree Analysis (FTA)[23], and Markov Processes[24] are effective for hardware-centric systems but insufficient for intelligent ships. These models cannot adequately represent software faults caused by logical complexity, abnormal inputs, or environmental conditions, nor can they capture interaction chains such as sensor data affecting software decisions or software controlling hardware execution. As a result, fault propagation across software and hardware remains poorly understood, limiting the accuracy of system reliability evaluations.

Table 1 summarizes the literature on software–hardware interaction reliability. Although prior studies have considered coupling effects, most approaches remain at a macro level, classifying failures into software, hardware, or interaction-induced categories [15][17][19]. These methods lack system-level analysis of how architecture influences failure behavior and rarely model detailed dependencies between specific software and hardware components. Consequently, existing frameworks cannot adequately capture system-level failure mechanisms, and research on reliability design under software–hardware coupling remains limited. For highly integrated, mission-driven systems such as intelligent ships, current approaches show clear limitations in structural modeling, fault-to-task mapping, and interaction mechanism representation. This highlights the need for a system-architecture-based modeling and evaluation method that accounts for software–hardware coupling to support the safety design and operation of intelligent ships.

Table 1 Literature review on software-hardware coupling reliability

Literature		[13]	[15]	[16]	[17]	[19]	[20]	[22]	Our paper
Research view	Software-driven	√					√		
	Hardware-driven								
	Software-hardware coupling		√	√	√	√		√	√
Level	partial system structure	√							
	System		√	√	√	√		√	√
Coupling granularity	Abstract interaction		√	√	√	√	√		
	Specific logic	√						√	√
System structure									√

Petri Net (PN) and Dynamic Bayesian Network (DBN) models have been widely applied in reliability studies. However, when the research object is further extended to cyber–physical systems characterized by complex architectures and tightly coupled software–hardware interactions, their direct application still faces notable limitations. PN-based models primarily focus on event-driven state transitions and are capable of describing system operational logic and fault propagation paths. However, they lack

robust support for systematic probabilistic inference and uncertainty quantification. In contrast, DBNs emphasize the modeling of probabilistic dependencies, but their network structures typically rely on predefined assumptions and are often constructed based on component independence or simplified coupling relationships, making it difficult to capture the dynamic evolution mechanisms of software–hardware interactions in complex systems. Therefore, existing PN and DBN approaches alone are insufficient to support comprehensive reliability analysis of complex systems with tightly coupled software and hardware.

To address these issues, this paper proposes a software–hardware coupling reliability modeling method that integrates PN with DBN. Specifically, PN is employed to explicitly characterize the dynamic interaction behaviors and fault propagation paths between software and hardware in complex systems, and their structural information is further formally mapped into the temporal dependency structure of a DBN. On this basis, the DBN is used to perform probabilistic reasoning on the uncertainties in the system state evolution process and to evaluate system-level reliability. In this way, the proposed method achieves an organic integration of software-hardware coupling mechanism modeling and uncertainty-aware probabilistic inference, providing a new modeling and evaluation framework for reliability analysis of complex systems with tight software–hardware interactions. A comparison is conducted between the method presented in this paper and several commonly used reliability analysis methods, as summarized in Table 2. The proposed framework makes three key methodological contributions:

(1) Formalzation of software-hardware interaction mechanisms: It constructs a PN-based causal propagation structure that explicitly represents cross-layer fault dependencies.

(2) Dynamic reliability modeling: It embeds the PN-derived causal pathways into a DBN, thereby enabling the system to capture asynchronous temporal evolution and dynamic variations in reliability throughout mission execution.

(3) Unified probabilistic reasoning: It integrates the PN-DBN interaction model into a system-level Bayesian framework, forming a unified probabilistic reasoning structure capable of representing both component-level behaviors and mission-level system performance.

Table 2 Method comparison

Method	Modeling perspective	Hardware-software coupling	Dynamic failure propagation
Fault tree analysis	Static logical structure	Assumes independence	No temporal evolution
Markov chain	State-transition based	Limited (state explosion for complex coupling)	Yes
Reliability block diagram	Functional configuration	Independent blocks	Static
DBN	Time-sliced probability model	Dependency modeling possible	Temporal degradation
PN	Discrete-event mechanism modeling	Logical interaction description capability	Strong process propagation capability
Proposed PN-DBN	Mechanism-driven CPS dependency modeling	Structural and functional mapping	Toke-flow +logic-triggered probabilistic evolution

The remainder of this paper is organized as follows. Section 2 analyzes the software-hardware architecture of the intelligent ship's autonomous navigation system, the task failure logic,and the characteristics and impacts of coupled failure modes. Section 3 presents a system reliability modeling method for intelligent ships that considers software-hardware coupling. Section 4 evaluates task reliability, identifies and analyzes key software and hardware components, examines model uncertainty, and compares the results with those of a non-coupled model. It also discusses the applicability, limitations, and future research directions of the proposed approach. Section 5 concludes the study.

2. Analysis of system structure of intelligent ships and software-hardware coupling failures

2.1 Typical structure of intelligent ship systems

As system logic structures vary with different tasks, this study focuses on a single-task scenario, specifically analyzing the logical architecture of an autonomous navigation system, as shown in Fig. 1.

In the autonomous navigation system, the software components in the intelligent operation control system are primarily responsible for information processing and transmission during task execution. The perception software processes echo signals received from the perception devices to monitor status and extract target parameter information, which is then transmitted to the state prediction software. The positioning software detects the ship's own position, attitude, and velocity. Based on the target parameters from the perception software and the self-motion data from the positioning software, the state prediction software calculates target information, including the target's position, speed, and attitude. The target recognition software receives target information from the state prediction software, identifies the target type, and produces refined target data. The threat assessment software evaluates the threat level of the identified targets based on the information provided by the target recognition module. The decision-making software is responsible for mode switching, behavior decision-making, trajectory planning, and action planning. Once the threat information is received, the decision-making software determines the appropriate response mode according to the operational scenario. Finally, the motion control software generates position and attitude control commands based on the decisions and plans from the decision-making module, enabling precise control of the ship's movements.

In addition, feedback mechanisms are commonly observed in complex systems [25]. Throughout the autonomous navigation task, the system continuously receives environmental feedback through environmental perception devices, while the actuators influence the environment and thereby affect the perception inputs. These

environmental inputs are fed back into the system, enabling dynamic adjustments to target tracking, threat assessment, and motion planning strategies. In this manner, the autonomous navigation system operates in a closed-loop fashion, integrating internal decision logic with real-time environmental feedback to maintain safe and efficient navigation.

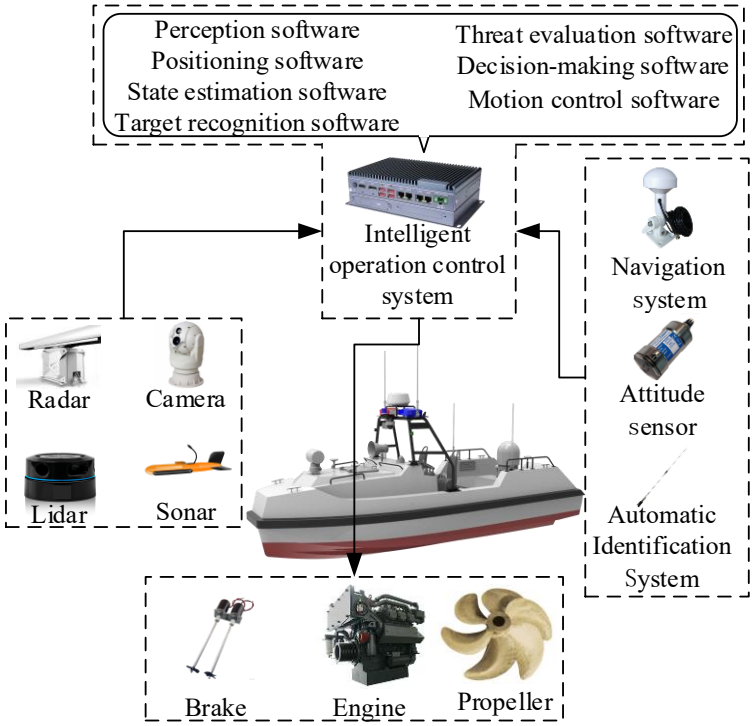


Fig.1 Intelligent ship structural logic

Compared with hardware devices, the failure mechanisms of software are fundamentally different. Software operates strictly according to its design, and software failures primarily result from design flaws and coding errors [26]. Software reliability has become a central concern in modern system design and development. However, in some cases, software that operates correctly can nevertheless contribute to accidents. This is not only due to issues inherent in the software but also due to the software-hardware interactions, especially when poor or incorrectly formatted input data are involved. For example, distorted input data may cause sensor voltage signals to be converted into incorrect digital values, leading to software parsing errors. Similarly, delayed sensor data updates can result in unstable control loops or logical deadlocks within the software [20]. In addition, the reliability of a software module also depends

on the health of the hardware platform on which it is deployed. Hardware degradation, such as processor overheating, memory faults, timing drift, or communication bus instability, may not directly cause software logical faults but can lead to execution failures, timing violations, or abnormal outputs. Therefore, in the intelligent ship system, software failure analysis should not only take into account the internal failure rate of the software itself and external data errors but also account for software failures induced by hardware platform malfunctions.

Likewise, hardware components such as actuators may fail due to incorrect commands. For instance, if the decision-making software misjudges the current heading and issues continuous rudder commands, the rudder actuator may overheat or even experience motor burnout as a result of prolonged high-load operation.

2.2 Software-hardware coupling characteristics

As an intelligent system integrating both software and hardware, the functional realization of an intelligent ship is based on a sequential information processing chain. Sensors first collect environmental and operational data, which is then analyzed, processed, and integrated by various software modules in the intelligent operation control system. The decision-making software evaluates the processed data to formulate navigation strategies and operational commands, which are subsequently transmitted to control software and actuators to drive the ship's actions. This constitutes a clear information transmission flow that aligns with the logical structure of intelligent ship operations.

The system also continuously interacts with the external environment. Actuators influence the ship's motion and behavior, which in turn alters environmental conditions such as relative wind, flow, and surrounding obstacle dynamics. These changes affect the measurements obtained by perception devices, forming an environmental feedback loop.

In intelligent ships, faults tend to propagate along this information flow and its associated feedback pathways. For example, a sensor malfunction, such as missing

values, incorrect readings, out-of-range data, redundant or frozen signals, noisy measurements or data type mismatches, can cause downstream software modules to process incorrect input. This can generate erroneous outputs and misjudgments that may result in abnormal ship behavior. Consequently, given the characteristics of fault propagation, reliability analysis of software-hardware coupling should be based on information transmission, environmental interactions, and control dependences. The information transmission structure of the intelligent ship system is illustrated in Fig.2.

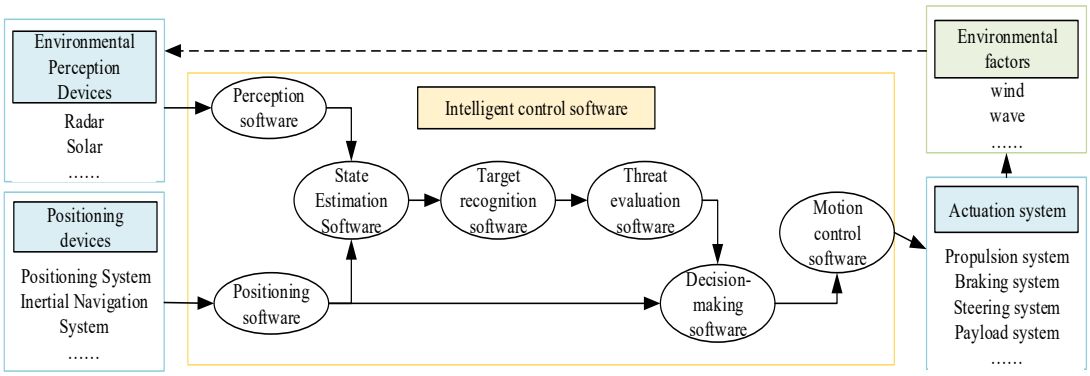


Fig. 2. Information flow in an intelligent ship system

2.3 Typical software-hardware coupling FMEA

FMEA is used to identify potential failure modes within a system and their causes, and to evaluate the effects of each failure mode on system performance. To improve the accuracy of reliability modeling for software-hardware coupled systems in intelligent ships, this study introduces Coupling Failure Modes and Effects Analysis (Coupling-FMEA) prior to the modeling phase. Since the focus of this work is on the reliability assessment of the system itself, the Coupling-FMEA targets key software modules and hardware components, systematically identifying potential coupling failure modes, failure mechanisms, and impact pathways during their interactions. It should be noted that, when identifying coupling-related failure effects, environmental feedback, such as wind speed, wind direction, current speed, and current direction, is also taken into consideration. These environmental factors can be influenced by actuator behavior and, in turn, affect the performance of hardware sensing devices, thereby participating in the propagation of coupled failures. To ensure clarity and maintain focus on the system reliability assessment, the Coupling-FMEA in this study is used solely for structural

failure identification, rather than for comprehensive risk prioritization.

Table 3. System coupling boundaries

Functional Module	Hardware Component	Software Module	Key Coupling Interfaces
Perception	Radar, cameras, sensors, etc.	Perception Processing Software	Sensor Data Input
Position	Satellite positioning, inertial navigation, etc.	Navigation and Positioning Software	Position Information Input
Decision-making	Onboard computing platform	State Estimation Software, Target Recognition Software, Threat Assessment Software, Decision-Making Software	Information Transmission
Actuation	Propulsion, braking, etc.	Control Software	Command Output

To conduct a software-hardware coupling failure analysis for the intelligent ship's autonomous navigation system, it is essential to clarify the system's functional boundaries and modular composition. The autonomous navigation task of intelligent ships relies on core functional modules, including perception, positioning, decision-making, and execution. The perception module primarily collects environmental data through radar, cameras, and various sensors. The positioning system integrates information from electronic nautical charts and navigation systems to achieve high-precision ship localization. The decision-making module leverages perception and positioning information to perform route planning and behavior judgment, relying on complex software logic and its shipboard computer platform to achieve autonomous control. The execution module physically responds to navigation commands by coordinating hardware components such as the propulsion, braking, and steering systems through the main control system. The system's functional modules and key interfaces are summarized in Table 3.

To perform Coupling-FMEA analysis, the coupling interfaces are first sorted based on the composition of system functional modules and their key interdependencies. All interaction interfaces and dependency paths among software-hardware, software-software, hardware-software components and environmental feedback are then

extracted. For each coupling relationship, potential failure modes are identified, and the failure propagation paths and impact scope within the system are determined, including their effects on task execution, functional modules, and ultimately the system-level reliability.

Table 4. Coupling-FMEA table

Coupled Components	Coupling Type	Failure Mode	Consequences
Perception Devices → Perception Software	Data Dependency （Hardware–Software）	Data loss or delay	Perception module failure, inaccurate scene reconstruction
Positioning Devices → Positioning Software	Data Dependency （Hardware–Software）	Signal offset	Large positioning error, deviation in the basis of path planning
Perception Software → State Estimation Software	Data Dependency （Software–Software）	Perception data error, loss, or delay	State estimation deviation, incorrect situational awareness
Positioning Software → State Estimation Software	Data Dependency （Software–Software）	Position error and signal loss	State prediction deviation, affecting target tracking and prediction
State Estimation Software→ Target Recognition Software	Data Dependency （Software–Software）	State estimation error and model lag	Reduced identification confidence, incorrect target classification
Target Recognition Software → Threat Assessment Software	Data Dependency （Software–Software）	Target type misclassification and feature extraction errors	Threat level misclassification, erroneous triggering or failure to trigger safety strategies
Threat Assessment Software → Decision-Making Software	Data Dependency （Software–Software）	Threat level anomaly or delayed safety strategy activation	Incorrect decision strategy or action selection error
Positioning	Data Dependency	Position	Path planning

Software → Decision-Making Software	（Software–Software）	misalignment or inconsistency	deviation and obstacle avoidance failure
Decision-Making Software → Control Software	Control Dependency （Software–Software）	Command conflicts or action mismatches	Control failure and action delay
Onboard computing platform → Software	Resource Dependency (hardware degradation)	Processor overheating, memory faults	Software timing violation, incorrect processing
Control Software → Actuator Devices	Control Dependency （Software–Hardware）	Abnormal parameters, protocol incompatibility, frequency instability	Execution failure, abnormal behavior, and system loss of control
Actuator Devices→ Environment→ Perception	Environment-mediated Dependency (feedback loop)	The actuator did not perform or made an incorrect action	Collected distorted / abnormal data

The typical coupling failure modes and impact analysis for intelligent ships are presented in Table 4, highlighting the key coupling relationships among major components, associated failure modes, and their consequences on system performance and mission execution.

3. Reliability modeling method considering software-hardware coupling

3.1 Proposed overall modeling framework

To achieve quantitative modeling of software-hardware coupled reliability, this paper proposes a dual-layer modeling framework that integrates propagation modeling and structural modeling. At the propagation level, a Coupling-FMEA is conducted to analyze failure modes and the impact of hardware-software interactions. Based on Coupling-FMEA, a PN is developed to model fault propagation mechanisms, and a graph-structure transformation method is applied to embed fault propagation relationships into DBN. At the structural level, a DBN is employed to represent the system logical structure and the state of the component based on system structure analyze. The overall process of reliability modeling for intelligent ship systems with software-hardware coupling is shown in Fig. 3.

Based on the intelligent ship's software-hardware system, the following assumptions are made in this study.

1) Each device is considered as an independent entity, and only devices directly related to autonomous navigation tasks are included.

2) Software modules are categorized by function, and their reliability status is affected only by upstream fault propagation and their own logic.

3) The state transitions of hardware components follow a first-order continuous-time Markov process, with exponentially distributed sojourn times and constant state transition rates.

4) Hardware maintenance and software updates are not considered during the operation of intelligent ships.

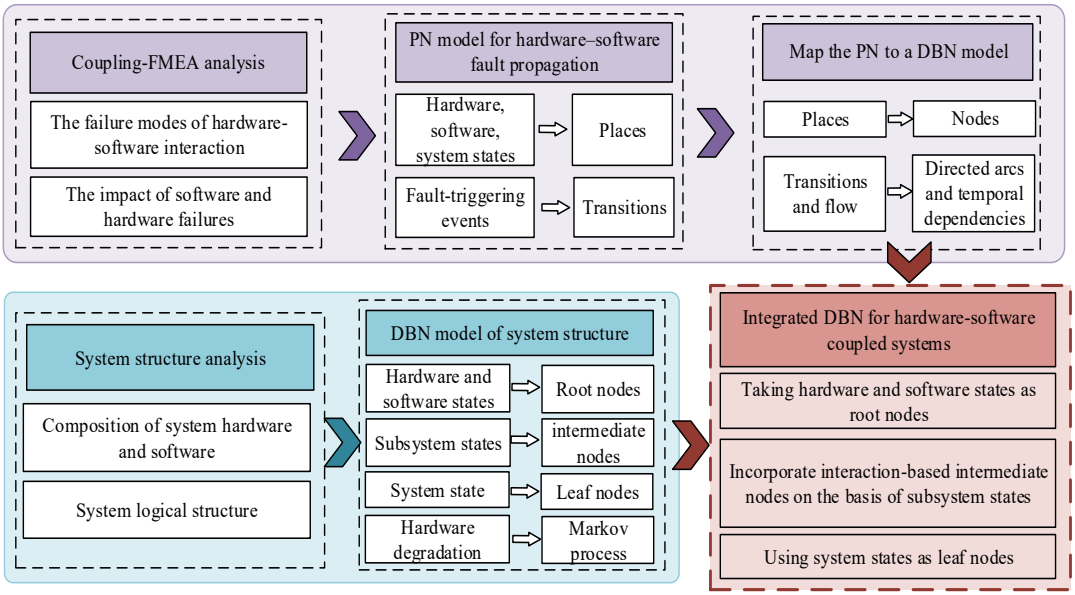


Fig. 3. The proposed reliability modeling flowchart

3.2 Key construction method

3.2.1 Construction of the software-hardware coupling model

PN, as a graphical mathematical modeling and analysis method, has been extended and applied in fault analysis and reliability fields. PN possesses the capability to model discrete event dynamic systems [29], and is well-suited to characterize the flow of information between components in intelligent ships, as well as the dynamic mechanism of fault propagation. With its intuitive graphical representation and rigorous

mathematical formalism, PN can clearly illustrate the entire process from the occurrence of software or hardware faults to the resulting operational anomalies in the vessel. The basic elements of a PN include a set of places, transitions, and flow relations, represented by "*P*", "*T*", and "*F*", respectively. The set of places denotes fault states, representing the current condition of software and hardware components. The set of transitions denotes fault events, representing events triggered by the current hardware or software states. Flow relations are represented by directed arcs, whose directions go from places to transitions or from transitions to places, as shown in Fig. 4.

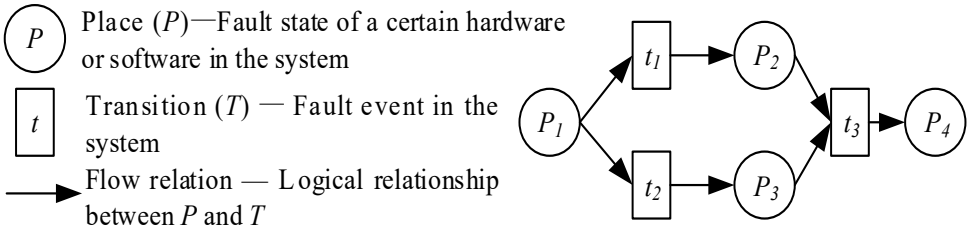


Fig. 4. Basic form of fault path representation using PN

To accurately represent the dependency and propagation of software and hardware failures within intelligent ship systems, this study employs a PN to model the fault states and their transmission relationships. PN enables a clear depiction of how software or hardware modules transition from one fault state to another through the activation of specific failure events, thus capturing the dynamic propagation behavior across system components. In this framework, software-internal failures caused by logic flaws or code defects are denoted as P_{SW}, while hardware-internal failures resulting from degradation or aging are denoted as P_{HW}. To reflect faults that emerge through interactions, the model defines P_{SF} and P_{HF}: P_{SF} represents software failures induced either by abnormal data received from upstream modules despite no intrinsic fault, or by its own internal P_{SW} failure; P_{HF} similarly represents hardware execution failures caused either by erroneous signals from upstream modules or by its own internal P_{HW} faults. P_{RE} denotes the relative environmental variations induced by the ship's own operation. Based on the coupling types identified in the Coupling-FMEA (Table 4), the propagation paths of faults across software and hardware are structured accordingly, as shown in Fig. 5.

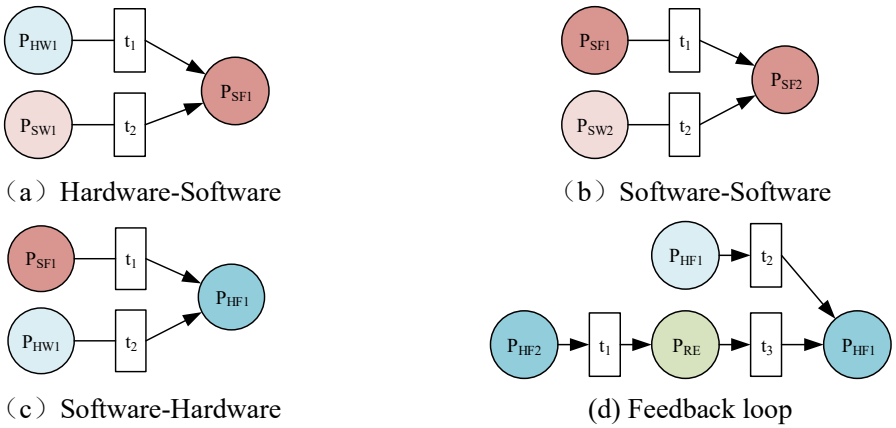


Fig. 5 Description of software-hardware failure propagation paths

Based on the software-hardware Coupling-FMEA analysis in Section 2.3, the system's various types of coupling in the system are categorized, and the interaction relationships and key dependency paths among functional modules are extracted. These dependencies form potential channels for failure propagation. For each pair of modules with a dependency relationship, the corresponding failure state nodes and failure event transitions are defined to construct a PN model that represents the dynamic evolution of system failures. The overall evolution of failures and the potential task-level consequences are represented within the PN, where P_{TF} denotes system failure or mission loss. As illustrated in Fig. 6, the PN diagram includes representative forms of software-hardware coupling, encompassing interactions such as hardware-to-software, software-to-software, software-to-hardware and feedback loop. In this model, P_{HW1} and P_{HW2} denote sensor failures, while P_{HW3} represents the shipboard computer failure. P_{SF1}, P_{SF2}, and P_{SF3} characterize software failures. Additionally, actuator P_{HF4} influences the relative environmental node P_{RE1}, which subsequently provides feedback to sensor P_{HF1}.

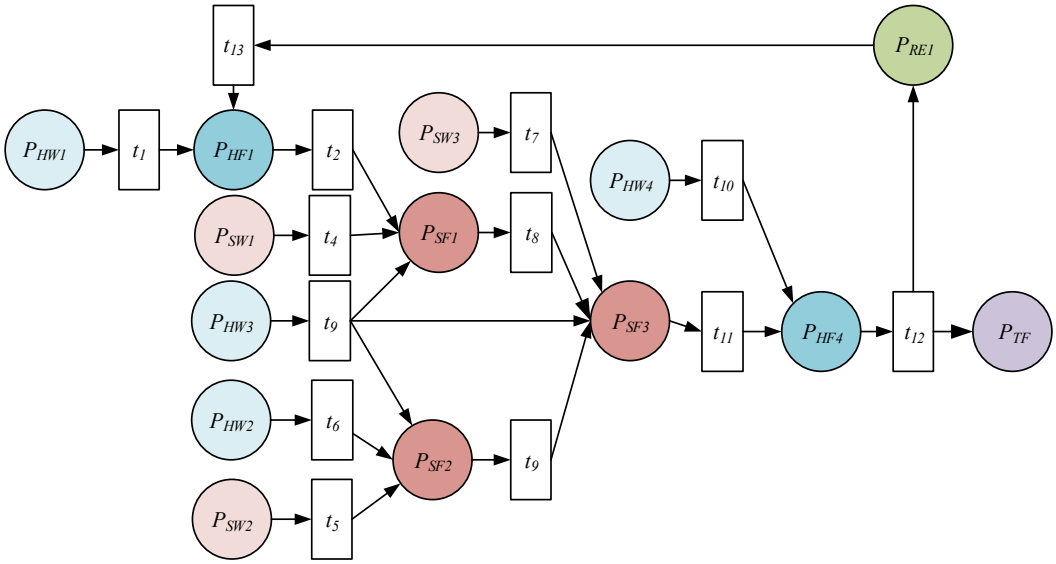


Fig. 6 An example of the PN model

3.2.2 DBN independent model construction

While PN is effective in describing the dynamic process of fault propagation, it has limitations in probabilistic reasoning and uncertainty quantification. In contrast, BN provides strong probabilistic analysis capabilities, allowing the quantitative representation of causal relationships and failure probabilities between nodes based on prior knowledge [28]. However, considering that BN is insufficient to capture the dynamic characteristics of software-hardware interactions. Therefore, the BN is further extended into a DBN by introducing time slices and temporal state-transition dependencies.

In the DBN model, each node represents a random variable. Root nodes represent the states of software or hardware components. Intermediate nodes represent subsystem states, while leaf nodes reflect the overall system state. Directed edges from parent to child nodes indicate causal relationships and conditional dependencies among random variables. To incorporate temporal dynamics, hardware components are further modeled with time-dependent state transitions. As shown in Fig. 7, the DBN model under independent software and hardware assumptions includes hardware and software nodes that correspond to the P_{SW} and P_{HW} states defined in the PN of Fig. 6.

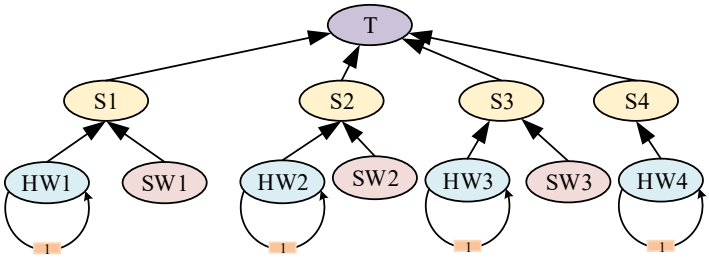


Fig. 7 DBN model under the Independence Assumption

3.2.3 Mapping method between PN and DBN

Since places and transitions in a PN can be mapped to state variables and causal events in a DBN, a clear structural correspondence exists between the two. This enables the transformation of the fault-propagation logic and state-evolution mechanisms described by the PN into the structural and parametric representation of a DBN, thereby achieving unified modeling. By combining the logical clarity of failure propagation of PN with the probabilistic computation strengths of DBN, this approach facilitates the quantification of software-hardware coupling failures and supports comprehensive reliability evaluation of intelligent ship systems.

To ensure consistency in this transformation, a PN-DBN Structural Mapping Matrix (SMM) is constructed to explicitly define the one-to-one correspondence between PN places/transitions and DBN nodes/edges. The SMM formalizes the mapping rules and ensures that the logical relationships represented in the PN are preserved in the DBN structure. Since the PN in this study focuses on the causal structure of fault propagation rather than quantitative firing rates, the SMM is defined as a binary topological mapping.

The SMM M_{p} captures the one-to-one correspondence between PN places and DBN state variables. The Eq. (1) ensures that each fault state modeled in the PN is represented as a discrete variable in the DBN.

$$M_p(i,j)=\begin{cases}1, \text{if PN place } p_i \text{ corresponds to DBN node } x_j,\\ 0, \text{otherwise.}\end{cases} \tag{1}$$

The structural SMM encodes the causal dependencies within a single time slice. Eq .(2) represents the topological structure of fault propagation in the PN. This matrix maps PN's intra-slice propagation relationships to DBN's intra-slice directed edges.

$$SMM(i,j)=\begin{cases}1, \text{if there exists a causal path } p_i \to t_k \to p_j \text{ in the PN,}\\ 0, \text{otherwise.}\end{cases} \tag{2}$$

The temporal mapping matrix defines cross-time-slice dependencies, representing how current states influence future states in the DBN. The Eq. (3) encodes the hardware/software degradation or evolution effects that are not captured by static PN semantics alone.

$$SMM_{DBN}(i,j)=\begin{cases}1, \text{ if PN contains feedback loop } p_i(t) \text{ to } p_j(t+1),\\ 0, \text{otherwise.}\end{cases} \tag{3}$$

As illustrated in Fig. 8, which shows the conversion of the PN model from Fig. 6 into a DBN model. The specific mapping steps are as follows:

1) Convert the places in the PN, which represent fault states, into discrete nodes in the DBN. Each node's state denotes the fault condition of a software or hardware component.

2) Convert the transitions and flow relations that represent state changes into directed edges within and across time slices in the DBN. These edges capture both causal relationship within a slice and temporal dependencies across slices.

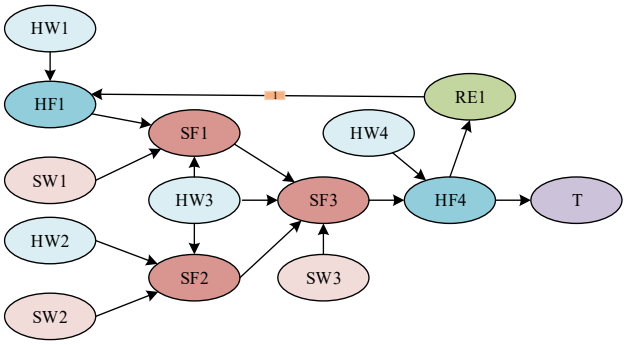


Fig. 8 Conversion from PN to DBN model

3.2.4 Uncertainty modeling and parameter construction

In complex systems, fault-tolerant control can detect, isolate, and compensate for faults; therefore, component failures do not necessarily lead to system failures. Consequently, Therefore, the Conditional Probability Table (CPT) within the DBN is utilized to represent the impacts of component failures in terms of probabilities. However, due to the scarcity of comprehensive statistical data regarding the specific effects of software and hardware states on system behavior, as well as on fault propagation probabilities induced by their interactions, these probabilities are

inherently uncertain and difficult to quantify using historical data alone. Moreover, constructing CPTs purely based on expert judgment introduces subjectivity and potential bias. To address this issue, this study adopts a multi-source information fusion approach to construct the CPTs [6]. Recognizing the structural and functional similarities between intelligent ship systems and traditional ship systems, the adopted method integrates expert knowledge with fault data through a Triangular Fuzzy Number(TFN)-based Analytic Hierarchy Process (AHP) framework. By representing expert judgments as fuzzy intervals rather than crisp values and combining them with available data, this approach effectively mitigates individual scoring bias and reduces subjectivity in expert assessments, thereby enhancing the objectivity and robustness of CPT estimation under uncertainty.

The specific steps for determining the CPT of a node using a multi-source information fusion method based on expert scores and a priori information are as follows:

(1) Determination of weights of experts and historical data by AHP;

(2) The linguistic variables of TFNs and their corresponding meanings are shown in Table 5, where *m* represents the central value of the fuzzy set, and *a* and *b* are the lower and upper bounds, respectively. Experts evaluate the influence relationships using linguistic terms, which are then converted into TFNs, historical data within the corresponding ranges are likewise transformed into TFNs, then weights are assigned to the obtained TFNs；

Table 5 Linguistic variables and corresponding TFNs

Linguistic variables	TFN (a, m, b)
Very high	(0.9, 1.0, 1.0)
High	(0.7, 0.9,1.0)
Slightly high	(0.5, 0.7, 0.9)
Middle	(0.3, 0.5, 0.7)
Slightly low	(0.1, 0.3, 0.5)
Low	(0, 0.1, 0.3)
Very low	(0, 0, 0.1)

(3) The average area method is used to defuzzify the fuzzy number, which is then normalized to obtain the conditional probability when only one parent node fails;

(4) Obtaining the complete CPT of the child node based on the conditional probability calculation model.

In the DBN model developed in this study, the prior probabilities of the root nodes represent the inherent reliability and failure states of software and hardware components. It is assumed that each component can exist in one of two states: perfect (denoted as "0") or failed (denoted as "1").

For hardware components, which undergo degradation due to usage and wear, their reliability is considered time-dependent during the execution of autonomous navigation tasks. As the mission progresses, the reliability of hardware components decreases. Assuming that the current time is t, the interval between the two time slices is Δt, and the failure rate of the component is expressed as λ. The formula for calculating the relationship between node transition probability and time across two-time slices is shown in Eq. (4) [27]. Using this, the state transition matrix of the faulty nodes in the hardware device can be determined.

$$B_{\rightarrow} = \begin{bmatrix} e^{-\lambda \Delta t} & 1-e^{-\lambda \Delta t} \\ 0 & 1 \end{bmatrix} \quad (4)$$

For software components, operational behavior is generally considered to be strictly governed by predefined logic and workflows. A software module is deemed to have failed if it cannot perform its intended function due to issues such as coding errors or logical defects. Unlike hardware, the modeling of software failure rates does not typically involve physical degradation over time. Instead, non-homogeneous Poisson process (NHPP) models are commonly used to statistically characterize software failure behavior [31]. As shown in Eq. (5), the NHPP model describes the software failure rate as a function of time during the testing phase, reflecting the decreasing failure intensity as faults are identified and corrected. Here, t is the time since the start of the software system integration test; the parameters p and q denote the number of faults to be detected and the fault detection rate. However, during the operational phase, this failure rate is often treated as a constant value, representing the stabilized reliability of the

software after deployment. In this study, we estimate the software failure rates using Mean Time Between Failures (MTBF)[33], based on statistical data from software testing reports and existing literature [32]. These failure rates are then used to define the prior probabilities of software nodes in the DBN model, as given in Eq. (6).

$$\lambda_{sw}(t) = pqe^{-qt}, \quad p > 0, q > 0 \tag{5}$$

$$MTTF = \frac{1}{pqe^{-qt}} \tag{6}$$

In complex intelligent ship systems, certain failure mechanisms are difficult to fully capture through explicit structural modeling, such as latent software defects, subtle sensor drift, communication anomalies, or disturbances arising from human–machine interaction. These factors are typically unobservable, non-reproducible, or supported by insufficient data. If they are entirely neglected, the resulting model may lead to overly optimistic reliability estimates. To account for unobservable or irreproducible failures without modifying the system topology, a Noisy-OR gate is used to introduce latent failure channels. This maintains the original causal structure while capturing previously unquantified failure factors.

Subsystem nodes that perform functional aggregation are selected, as such nodes are more susceptible to the combined influence of unmodeled factors. The original deterministic logical relationships of these nodes $Y = f(X_1, X_2, \ldots, X_n)$ are therefore replaced with Noisy-OR relationships, as expressed in the corresponding Eq. (7).

$$P(Y = 1 | X_1, X_2, \ldots X_n) = 1 - \prod_i (1 - p_i)^{X_i} \times (1 - p_{leak}) \tag{7}$$

where p_i denotes the activation probability associated with the known failure path. $X_i \in \{0,1\}$ is the state of the i-th parent cause (0=absent, 1=present); p_{leak} is the leak probability, representing unknown or unmodeled failure sources.

An interval is assigned to the leak probability in the Noisy-OR formulation to represent unknown failure influences. Under varying leak values, the system reliability is recalculated to analyze the resulting variability of the model and to evaluate the system's robustness against unmodeled risks.

3.3 Integration and inference of the model

To achieve accurate reliability assessment of intelligent ship systems under software-hardware coupling, this study proposes an integrated reliability evaluation model based on a DBN that unifies the structural dependencies and coupling mechanisms of the system.

To overcome the limitations of traditional software-hardware independent models, which represent components as isolated root nodes without modeling cross-domain fault propagation, the fault-coupling PN is transformed into a DBN and embedded into the independent system model to form a unified reliability framework.

During the PN-to-DBN conversion, places representing intrinsic hardware and software states (P_{HW}, P_{SW}) are directly mapped to the root nodes of the independent DBN to maintain structural consistency and avoid duplication. In contrast, places representing interaction-induced faults (P_{SF}, P_{HF}) are preserved as intermediate nodes only when they introduce causal relationships that cannot be represented by the intrinsic state transitions alone. This ensures that each DBN node contributes unique causal information and prevents redundant modeling.

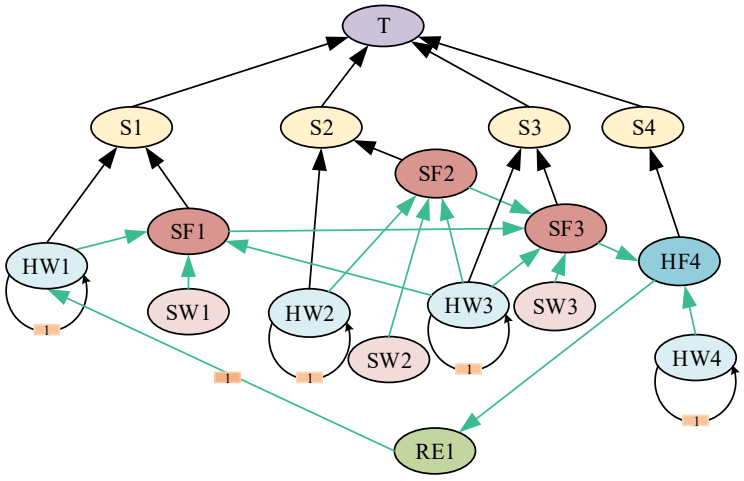


Fig. 9. Software-hardware coupled DBN system reliability model

By merging the coupling DBN with the independent system DBN through shared intrinsic state nodes and selectively introduced interaction nodes, the resulting model enables both system level reliability evaluation and cross-domain fault propagation analysis. The integrated DBN structure, as illustrated in Fig. 9, forms a complete

reliability assessment model for software-hardware coupled intelligent ship systems.

Due to the shared root nodes, the prior probabilities in the integrated model represent the inherent probability states of the software and hardware components. The CPTs are determined using the multi-source information fusion method proposed in Section 3.2.4, which captures both the influence of software and hardware component states on system-level functions and the dependency relationships within potential fault propagation paths. The system reliability is evaluated using the GeNIe software by inputting the prior probability of root nodes, the state transition matrix of the hardware device and the conditional probability of non-root nodes, enabling the calculation of the overall mission reliability of the system.

4. Evaluation procedure and case study

4.1 Evaluation procedure

To quantitatively assess the reliability of an intelligent ship's software-hardware coupled system, this study proposes a systematic reliability evaluation procedure based on the aforementioned modeling approach. The evaluation consists of the following steps:

1) Based on the system's functional architecture, identify the functional logic and the corresponding software and hardware units, and analyze the interaction and dependency paths between them.

2) Apply the Coupling-FMEA method to systematically identify interaction failure modes between software and hardware, the mechanisms of fault propagation, and their impact on mission execution.

3) Develop a PN model based on the identified coupling relationships. Define places and transitions to represent intrinsic software/hardware faults and interaction-induced faults, in order to capture the dynamic evolution process of failures in the coupled system.

4) The elements representing the fault states and trigger logic in the PN model are mapped to the nodes and causal edges in the DBN, and then embedded into the system'

s independent structure DBN model to achieve integration of the propagation and structure models.

5) Calculate prior probabilities using the relevant failure parameters, and construct CPTs by integrating expert judgments with statistical data.

6) Employ DBN inference to compute system reliability and perform sensitivity analysis to identify critical components or weak links within the system.

4.2 Comparison with existing reliability analysis Methods

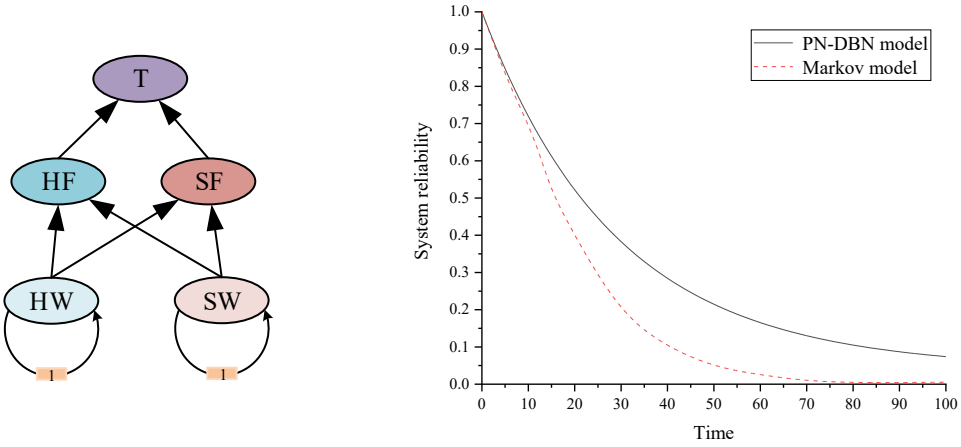
4.2.1 Comparison with existing methods

To validate the practical applicability of the proposed method in real engineering scenarios, this study selects a published work on software–hardware coupling reliability analysis of an embedded system as a comparative case [15]. In that work, a Markov process is employed to model the interaction failure states between software and hardware. The system states are categorized into software states, hardware states, hardware states induced by software, and software failures induced by hardware, and the transitions among these states are described through predefined transition probabilities.

Based on these characteristics, this paper adopts the case presented in the reference [15] and conducts further structural modeling and reliability analysis of the embedded system. It should be noted that, in the reference [15], system reliability is expressed as the product of software reliability, hardware reliability, and software–hardware interaction reliability. The Markov process is primarily used to describe the evolution of interaction states while also incorporating software-based detection and repair mechanisms for hardware. Due to differences in modeling assumptions and analysis objectives between this study and the reference, the comparison is not intended to achieve strict numerical equivalence under identical conditions. Instead, it aims to demonstrate the modeling capability and application potential of the proposed method for analyzing software–hardware coupling reliability in embedded systems.

During the comparison, the failure modeling approaches reported in the reference

are adopted, where hardware failures follow a Weibull distribution and software failures are modeled using the G-O model. Because the reference does not provide detailed joint failure data, the CPTs required by the proposed PN–DBN framework cannot be obtained directly. Therefore, the CPTs are constructed based on the reported failure models, state-transition information, and reasonable modeling assumptions to ensure consistency with the original failure mechanisms. The resulting DBN model and the corresponding reliability curves are shown in Fig. 10.



(a) DBN model for embedded systems (b) System Reliability Variation Curve

Fig. 10. Comparison of the Embedded System Case

Under the same failure mechanism assumptions, the PN–DBN approach is able to maintain an overall degradation trend consistent with that of the state-based method reported in the reference, while requiring comparatively fewer nodes for modeling. This demonstrates the effectiveness of the proposed method for embedded system modeling. Owing to differences in the prior data required for evaluation and the underlying modeling logic of the two approaches, the comparison is intended to verify that the proposed method can reproduce a consistent reliability trend.

The comparison shows that, at the early stage of operation, the reliability curves produced by the two models are nearly identical. However, as time progresses, the reliability predicted by the Markov-based model decreases more rapidly than that of the PN–DBN model. This discrepancy is mainly attributed to differences in modeling mechanisms and underlying assumptions. In the Markov-based approach, software–hardware interactions are abstracted into state transition rates, which may implicitly

amplify the impact of interaction-induced failures.

4.2.2 Comparison with traditional reliability assessment methods

Traditional intelligent ship reliability assessment methods are primarily hardware-focus, with modeling centered on the system's physical components. In such models [30], system reliability degradation is typically assumed to originate from hardware failures, while the influence of software, an essential element in intelligent ship systems, is largely neglected. In this case study, the hardware and the system are consistently characterized by a four-state model, in which "State 0", "State 1", "State 2", and "State 3" represent "perfect", "early fault", "general fault", and "serious fault", respectively.

In contrast, the proposed PN-DBN framework preserves the original system architecture while introducing an explicit representation of software–hardware coupling. Therefore, a comparison is conducted with the reliability of the autonomous navigation phase presented in the reference case [30]. On the basis of the existing structural model, functional software modules, including perception, localization, shore-based support, decision-making, and control software, are integrated, as shown in Table 6 with their failure probabilities set to 0.0001. The influence relationships during information transmission are described according to the CPT construction method. The added nodes and their corresponding meanings are listed in the table below. The software–hardware interaction relationships are first modeled using a PN, as shown in the Fig.11, and then transformed into a DBN and embedded into the original model, as illustrated in the subsequent Fig. 12.

Table 6 Additional nodes introduced in the traditional intelligent ship reliability case

Component	No.	Component	No.
Perception software	SW1	Decision-make software	SW4
Position software	SW2	Execution software	SW5
Shore-based support software	SW3	Relative flow	RE

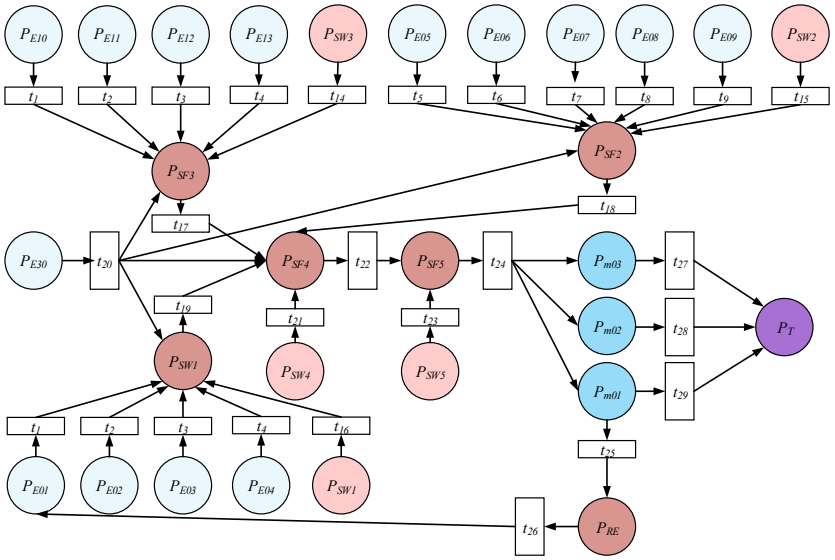


Fig. 11 PN diagram after adding software-hardware coupling

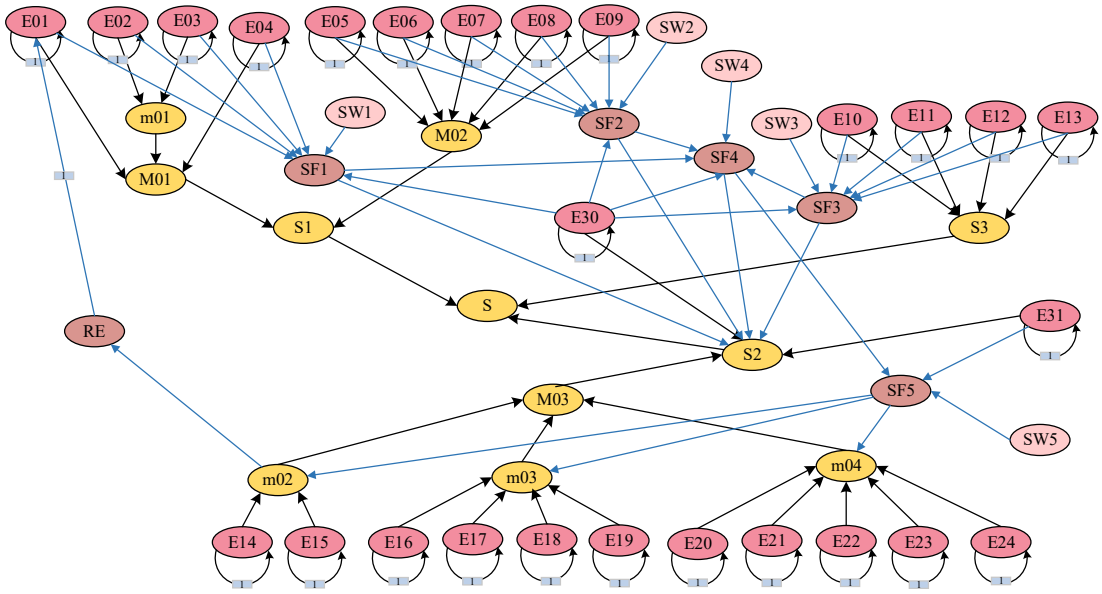


Fig. 12 DBN diagram after adding software-hardware coupling

After incorporating the additional nodes, the computed results are compared with the state distribution of the autonomous navigation phase reported in the reference [30], as summarized in the Table 7.

Table 7 Comparison of state distributions

	PN-DBN model	Hardware-only reliability model
State 0	0.4970049	0.5030495
State 1	0.1472039	0.1744195
State 2	0.1485206	0.1660893
State 3	0.2072706	0.1564416

The system reliability obtained using the hardware-only failure approach is

0.8435584. In contrast, our study incorporates the interaction between software and hardware failures through the PN representation and explicitly accounts for the impact of software failures within the reliability model, thus resulting in a system reliability of 0.7927294. Neglecting software-hardware interdependencies leads to overly optimistic reliability predictions, which could compromise safety margins in autonomous maritime operations.

4.3 Case study

4.3.1 Mission system analysis

This study takes the Unmanned Survey Vessel (USV) YLI-2500C-T1 as a case example, as shown in Fig. 13. To verify the applicability and effectiveness of the proposed software-hardware coupling reliability modeling and evaluation method, a mission scenario is defined in which the unmanned vessel continuously performs a 24-hour autonomous survey on an inland lake. During this period, the mission is carried out without human intervention and relies entirely on the core functional modules on board. These modules include perception, positioning, decision-making and execution systems, which are responsible for completing operations such as environmental perception, path planning, and course control. This mission places high demands on the stability and reliability of software-hardware cooperation across the system. In particular, for long-duration tasks, any software-hardware coupling failure in key functional modules may result in mission failure.

Fig. 13 YLI-2500C-T1unmanned survey vessel

4.3.2 Construction of model

The functional architecture of the system reveals that the successful completion of autonomous navigation tasks relies on the coordinated operation of four key subsystems:

the perception system, localization system, autonomous decision-making system, and execution system. The perception system is responsible for acquiring information about the external environment, including obstacles, water boundaries, and weather conditions. It integrates multiple sensors to construct a comprehensive environmental awareness map. The localization system provides real-time information on the vessel's position, attitude, and velocity, forming the basis for navigation and path planning. The autonomous decision-making system processes data from the perception and localization systems to perform route planning, obstacle avoidance, and generate mission control commands. Finally, the execution system receives instructions from the decision-making unit to perform vessel maneuvers such as attitude adjustment and propulsion control. The hardware and software components of the unmanned vessel are organized according to these functional subsystems, as summarized in Table 8.

Table 8 Structure of the unmanned survey vessel

Subsystem	Component	No.	Failure rate /h^{-1}
Perception system (S1)	Sonar	HW1	9.1110×10^{-5}
	Camera system	HW2	2.611945×10^{-4}
	LiDAR	HW3	1.12×10^{-3}
	Millimeter-wave radar	HW4	1.7033789×10^{-3}
Position system (S2)	Attitude sensor	HW5	9.4718×10^{-6}
	Navigation system	HW6	6.666667×10^{-4}
	Electronic chart display and information system(ECDIS)	HW7	1.6672×10^{-5}
	Automatic Identification System (AIS)	HW8	2.2614×10^{-5}
Master control system (S3)	Perception software	SW1	9.185×10^{-3}
	Positioning software	SW2	1.228×10^{-2}
	State estimation software	SW3	1.222×10^{-2}
	Target recognition software	SW4	1.017×10^{-2}
	Threat assessment software	SW5	7.5×10^{-2}
	Decision-making software	SW6	1.34×10^{-1}
	Execution software	SW7	1.8×10^{-2}
	Edge computing processor	HW9	$2.7848*10^{-4}$
	USV navigation and control core	HW10	$5.7*10^{-5}$
Actuation (S4)	Propulsion system	HW11	7.26×10^{-4}
	Braking system	HW12	1.013×10^{-5}
	Steering system	HW13	5.959×10^{-5}
	Payload system	HW14	6.22×10^{-5}

To capture the external disturbances acting on the unmanned surface vessel, two environment nodes are incorporated into the model to represent the relative wind and relative flow encountered by the USV during operation, denoted by RE1 and RE2 respectively. Since the focus of this case study is to evaluate the inherent operational reliability of the intelligent ship system rather than the random characteristics of the environment, environmental disturbances are regarded as known external conditions. Their states are set to be constantly present in the modeling process, meaning that a prior probability of 1 is assigned to state “0”. This configuration allows the model to represent the presence of environmental effects while ensuring that the reliability analysis concentrates solely on the system’s internal fault dynamics and software-hardware interaction mechanisms.

Based on the key software-hardware coupling paths and failure modes identified in Section 2.3, this section analyzes the specific software-hardware interaction failure modes, fault propagation mechanisms, and their impact on task execution for the unmanned survey vessel. The analysis further identifies the potential fault states of various software and hardware modules during the fault propagation process, as well as the corresponding triggering events, as summarized in Table 9. On this basis, a software-hardware fault propagation model is constructed using a PN framework. Different types of fault states and event nodes are systematically mapped within the model to depict the evolution paths of coupled failures throughout the system. As shown in Fig. 14, the PN model illustrates the fault propagation process of the unmanned vessel under software-hardware coupling relationships.

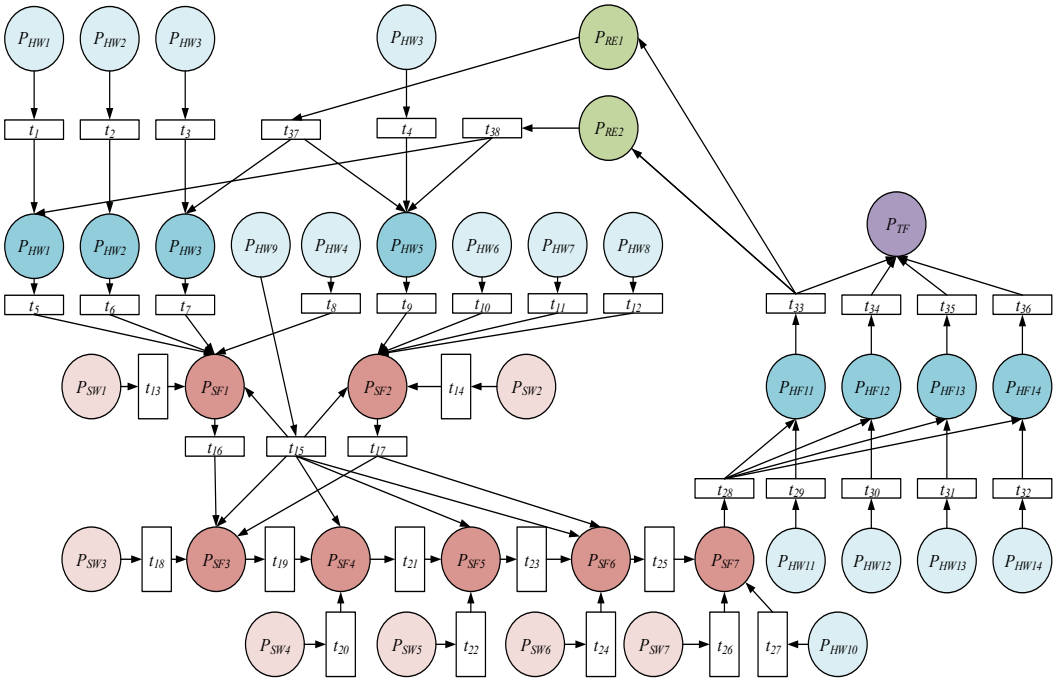


Fig. 14 Fault propagation model of the USV autonomous navigation system

The places in the PN model representing fault states are mapped to fault state nodes in the DBN, while the transitions representing state changes and triggering logic are mapped to causal directed edges. Based on this, the fault propagation model is embedded into the system's original independent structural DBN model. Through root node alignment and the integration of interaction nodes, a unified soft–hardware coupling reliability assessment model is formed. The integrated DBN structure is shown in Fig. 15.

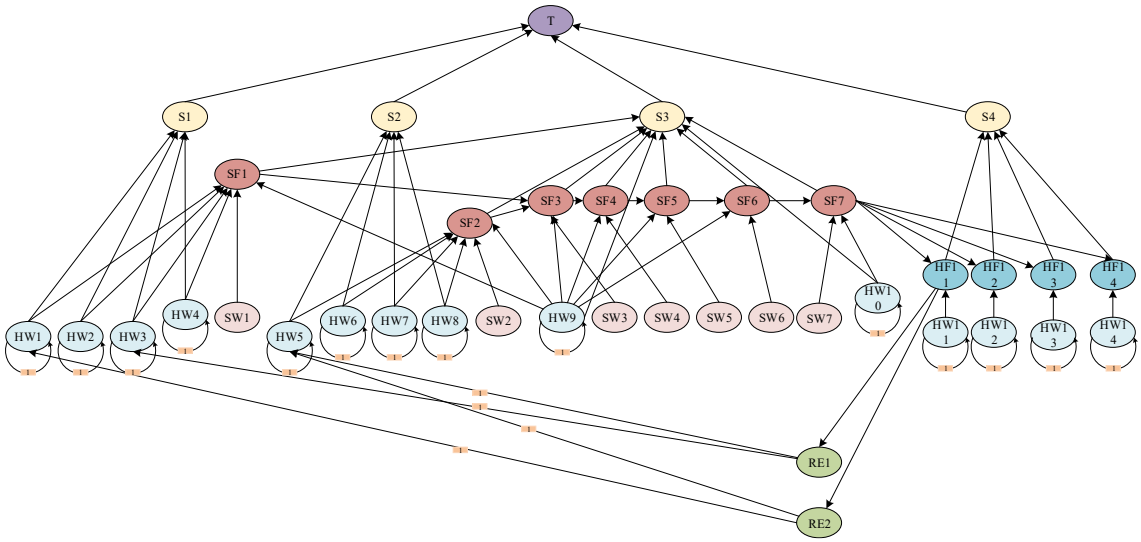


Fig. 15. DBN model for system reliability under software–hardware coupling

Based on the Nonelectronic Parts Reliability Data (NPRD), Offshore Reliability Data (OREDA), electronic equipment reliability prediction models and data handbooks, as well as data from relevant literature [30], the failure parameters of each hardware component are obtained, as shown on the right side of Table 8. Considering the degradation behavior of the hardware devices, all devices are assumed to be in a normal state at the start of the mission. A one-hour time step is adopted to construct the state transition matrix of the hardware devices. Referring to existing literature and statistical results from typical software testing reports [31], the failure rates of functional software modules are estimated and adopted as prior probabilities in the DBN model.

Table 9 Fault states and events during fault propagation

No.	Failure state	No.	Failure event
P_{HW1}	Sonar internal fault	t_1	No valid sonar data
P_{HW2}	Camera system internal fault	t_2	No valid image data
P_{HW3}	LiDAR internal fault	t_3	No valid distance data
P_{HW5}	Attitude sensor internal fault	t_4	No valid attitude data
P_{HF1}	Sonar execution fault	t_5	Delayed or incorrect obstacle detection
P_{HF2}	Camera system execution fault	t_6	Misidentification of objects
P_{HF3}	LiDAR execution fault	t_7	Inaccurate distance measurements
P_{HW4}	Millimeter-wave radar internal fault	t_8	No valid radar data
P_{HF5}	Attitude sensor execution fault	t_9	No valid attitude sensor data
P_{HW6}	Navigation system internal fault	t_{10}	No valid localization data
P_{HW7}	ECDIS internal fault	t_{11}	No valid chart data
P_{HW8}	AIS internal fault	t_{12}	Inability to detect nearby vessels
P_{SW1}	Perception software internal fault	t_{13}	Target state detection error
P_{SW2}	Localization software internal fault	t_{14}	Position, velocity, and attitude estimation error
P_{HW9}	Edge computing processor fault	t_{15}	Processing delay or failure
P_{SF1}	Perception execution fault	t_{16}	Target state monitoring failure
P_{SF2}	Localization execution fault	t_{17}	Ship localization failure
P_{SW3}	State estimation internal fault	t_{18}	Motion state prediction error
P_{SF3}	State prediction execution fault	t_{19}	Motion prediction failure
P_{SW4}	Object recognition internal fault	t_{20}	Target classification error
P_{SF4}	Object recognition execution fault	t_{21}	Target recognition failure
P_{SW5}	Threat assessment software fault	t_{22}	Threat level assessment error
P_{SF5}	Threat assessment execution fault	t_{23}	Threat assessment failure
P_{SW6}	Decision-making software fault	t_{24}	Behavioral decision and motion planning error

P_{SF6}	Decision-making execution fault	t_{25}	Behavioral decision and planning failure
P_{SW7}	Execution software fault	t_{26}	Motion and actuation control error
P_{HW10}	USV navigation and control core fault	t_{27}	Loss of navigation and control
P_{SF7}	Control execution fault	t_{28}	Erroneous command issued
P_{HW11}	Propulsion system fault	t_{29}	Ship collision or grounding
P_{HW12}	Braking system fault	t_{30}	Collision avoidance failure
P_{HW13}	Steering system fault	t_{31}	Collision or loss of control
P_{HW14}	Payload system fault	t_{32}	Structural damage or capsizing
P_{HF11}	Propulsion system execution fault	t_{33}	Motor overheating or power interruption
P_{HF12}	Braking system execution fault	t_{34}	Brake jamming or hydraulic leakage
P_{HF13}	Steering system execution fault	t_{35}	Steering gear failure
P_{HF14}	Payload system execution fault	t_{36}	Floating state imbalance
P_{RE1}	Relative wind change	t_{37}	Changes in wind direction and speed
P_{RE2}	Relative water flow change	t_{38}	Changes in water flow direction and speed

Experts are invited to evaluate the impact of individual node failures on higher-level failures. An importance judgment matrix is constructed using the AHP through pairwise comparisons among three experts and the traditional data source, as given in Eq. (8). The qualitative comparison judgments are converted into quantitative values using the fundamental 1-9 scale method, where e_{ij} denotes the relative importance of factor i compared with factor j.

$$J=\begin{pmatrix} e_{11} & e_{12} & e_{13} & e_{14} \\ e_{21} & e_{22} & e_{23} & e_{24} \\ e_{31} & e_{32} & e_{33} & e_{34} \\ e_{41} & e_{42} & e_{43} & e_{44} \end{pmatrix}=\begin{pmatrix} 1 & 2 & 1/3 & 2/5 \\ 1/2 & 1 & 3/5 & 1/3 \\ 3 & 5/3 & 1 & 4/5 \\ 5/2 & 3 & 5/4 & 1 \end{pmatrix} \tag{8}$$

In Eq. (7), e_1, e_2, e_3 represent the three experts, while e_4 denote the sample data sources. First, the eigenvector corresponding to the maximum eigenvalue of the judgment matrix is calculated. Then, the vector is normalized to obtain the final weight values, as given in Eq. (9).

$$K_i=\begin{cases} K_1=0.1564 \\ K_2=0.1117 \\ K_3=0.3240 \\ K_4=0.4078 \end{cases} \tag{9}$$

The conditional probability corresponding to a single parent node failure is derived from sample data and transformed into the associated triangular fuzzy number. The

results from both methods are then weighed and normalized. Using a multi-source information fusion approach, the conditional probability of a single parent node failure is obtained. The complete CPT is then calculated based on the principles of joint and conditional probability, as shown in Appendix.

4.3.3 Result analysis

Based on the previously described DBN modeling method for software-hardware coupling, a system-level structural model encompassing the key components of the intelligent ship's autonomous navigation system has been established. The failure probabilities of hardware components during task execution have been calculated. This study uses GeNIe software to compute the system reliability. By inputting the prior probabilities of the root nodes, state transition and the conditional probabilities of the non-root nodes, the final calculated reliability of the autonomous navigation system during task execution is 0.759444. The reliability during the task varies over time as shown in Fig. 16. Under the current software-hardware configuration and parameter settings, the system demonstrates a certain level of reliability assurance. However, some risks still remain.

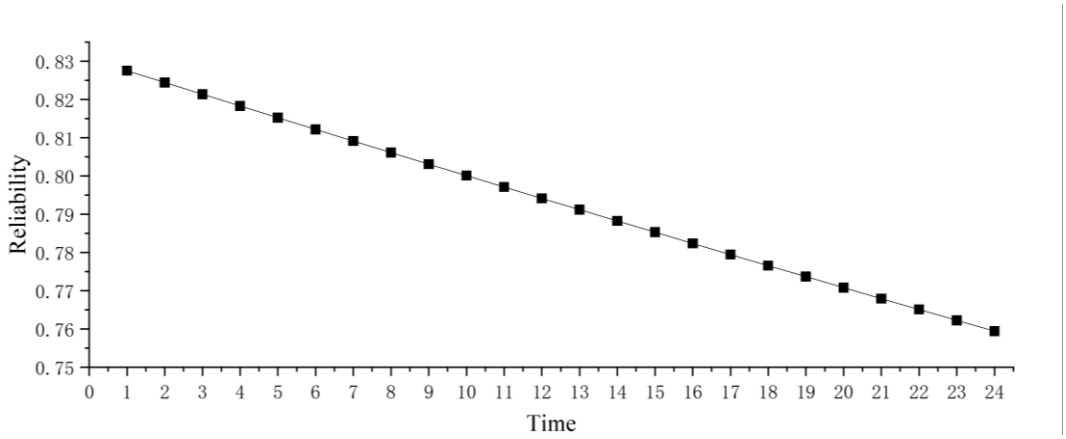


Fig. 16 Autonomous navigation system reliability

To analyze the influence of unknown failure factors, the method described in Section 3.2.4 is applied by modifying the node types of selected subsystems in the USV model and assigning leak parameters accordingly. Specifically, in GeNIe, the node types of S1, S2, S3, and S4 are changed to Noisy-OR nodes, with the leak parameter set within the interval [0.0001,0.001]. The resulting system reliability is calculated to

lie within the range [0.757365,0.759237]. The variation of the unknown parameter has only a minor effect on the overall reliability, which remains within an acceptable range, indicating that the model shows limited fluctuation in response to unmodeled influences, demonstrating a certain degree of robustness.

4.3.4 Model comparison

To verify the effectiveness of the proposed approach, a traditional software-hardware independent reliability model based on a structured BN is used for comparison. By comparing the results of these two models, the impact of software-hardware coupling and fault propagation on system reliability is quantitatively evaluated.

Table 10 Model reliability comparison

Model	Reliability Degree
BN software-hardware independent model	0.819965
Proposed software-hardware coupling model	0.759444

Under the same system structure, failure probabilities, and mission conditions, the system reliability of the intelligent ship performing a 24-hour autonomous navigation mission is calculated. The results show that the system reliability calculated using the traditional uncoupled model is 0.819965, and the reliability comparison results are presented in Table 10. The results indicate that the traditional model ignores fault propagation paths between software and hardware. For intelligent ship systems where software and hardware are highly coupled, this oversight leads to an underestimation of system risks. The proposed software-hardware coupling modeling approach more accurately reflects the actual operational mechanisms and enhances the accuracy and practical applicability of system reliability assessment.

To verify the role of feedback interactions in the model, the environmental feedback node is removed from the established PN–DBN model, thereby reducing the system to a structure that only includes unidirectional failure propagation. The system reliability is then recalculated under the same parameter settings. The results show that, without environmental feedback, the mission reliability of the unmanned vessel system is 0.762292. The comparison indicates that neglecting feedback interactions leads to a

slight overestimation of system reliability. Degradation in component performance not only directly affects current functions but can also propagate through the coupled processes of environmental perception, control decision-making, and execution response, thereby amplifying failure effects. If this feedback mechanism is ignored during modeling, the system is effectively treated as an open-loop structure, which cannot capture such cumulative effects and consequently yields an overly optimistic assessment of system resilience.

4.3.5 Sensitivity analysis

A sensitivity analysis of the system to identify the critical software and hardware components is also conducted in our study. By changing the prior probabilities of the root nodes in the DBN and observing the variation of the leaf nodes, i.e., by modifying the failure rates of components, their influence on the autonomous navigation task is analyzed. However, the failure mechanisms of software and hardware differ. Hardware failure probability is modeled using an exponential distribution, while perturbations produce a nonlinear saturation effect on the system. In contrast, software failure probability perturbations are introduced in a linear manner, making the system response more sensitive. In the sensitivity analysis, applying a uniform disturbance amplitude inevitably causes software components to exhibit a higher impact rate. To avoid this bias, this paper separately identifies critical components for software and hardware. Additionally, by estimating the actual change in hardware reliability resulting from increased hardware failure rates, the perturbation amplitude for software is adjusted to be approximately equivalent to that of hardware.

Each hardware component's failure rate is increased to 120% of its original value, and the resulting change in the autonomous navigation devices' reliability of mission duration is calculated. The average change in hardware component reliability due to such failure rate disturbance is estimated to be 0.0017. The software failure probability disturbance is then adjusted to produce an equivalent effect. The resulting change in system reliability caused by software failure rate perturbations is calculated accordingly.

From this, the sensitivity of software and hardware components are obtained, enabling the identification of the most critical components affecting the task system. Fig. 17 presents the sensitivity analysis results of the proposed model and the uncoupled BN model.

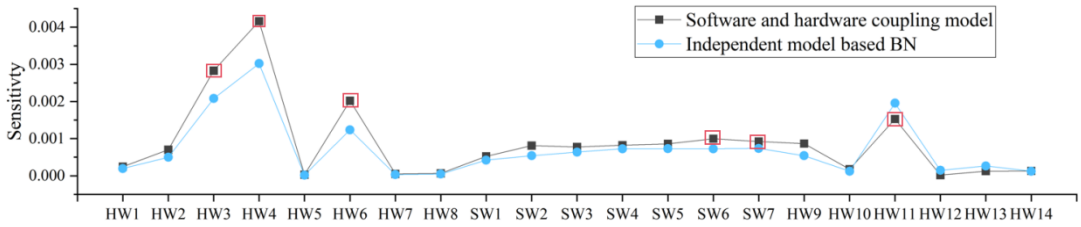


Fig. 17 Sensitivity analysis

According to the sensitivity analysis results, HW4 (millimeter-wave radar), HW3 (LiDAR), HW6 (navigation system), HW11 (propulsion system), SW6 (Decision-making software), SW7 (Execution software) exhibit the highest sensitivity, making them the critical hardware and software components for the autonomous navigation task to consider. Enhancing the reliability of these components, introducing redundancy, or implementing preventive maintenance can significantly improve reliability of these components, and hence the overall system reliability.

The coupling between hardware and software substantially changes the impact weights of individual components on system reliability. Traditional independent models, due to the lack of interaction consideration, fail to capture the actual risk amplification associated with components like HW3 and HW4, which play critical roles in the system's fault propagation. This limitation may lead to suboptimal system reinforcement strategies. In contrast, the proposed coupling model explicitly characterizes the fault interactions between software and hardware, enabling precise identification of high-sensitivity modules and offering more realistic decision support for reliability optimization in highly coupled intelligent ship systems. Moreover, components such as HW11, located at the back end of the fault propagation path, primarily affect local subsystems or exert direct influence. Their marginal impact tends to be partially offset within the propagation chain, resulting in lower observed sensitivity.

4.3.6 Fault tolerance analysis

In the original DBN model, the impact of component failures on higher-level system functions is typically described directly by conditional probabilities derived from multi-source information. This modeling approach implicitly incorporates redundancy design, software recovery, and fault isolation into a single probability value, without explicitly distinguishing between two different mechanisms: "fault occurrence" and "fault handling". Consequently, it becomes difficult to assess the system's inherent fault-tolerance capability.

To explicitly characterize this resilience mechanism, a Fault-Tolerant Control (FTC) node can be introduced between each component node and its parent node to represent whether the fault-handling strategy is successfully executed. Accordingly, the original dependency relationship $X_i \to S_j$ is restructured to $X_i \to FTC_{ij} \to S_j$ .The FTC_{ij} node represent whether the fault-handling mechanism for component X_i is effective, and its states are defined as $FTC_{ij} \in \{Effective, Failed\}$, serving as an abstract description of the fault-tolerance behavior. On this basis, a parameter δ is introduced to denote the probability that the fault-tolerance mechanism successfully mitigates the fault, while γ represents the probability that residual effects still propagate to the system even after successful fault handling. The resulting influence relationship between the FTC node and the component node is expressed as shown in the Eq. (10), and its effect on the child node is formulated as given in the Eq. (11).

$$\begin{cases} P(FTC_{ij} = Effective \mid X_i = Failure) = \delta \\ P(FTC_{ij} = Failure \mid X_i = Failure) = 1 - \delta \\ P(FTC_{ij} = Effective \mid X_i = Normal) = 1 \end{cases} \tag{10}$$

$$\begin{cases} P(S_j = Failure \mid FTC_{ij} = Failed) = 1 \\ P(S_j = Failure \mid FTC_{ij} = Effective) = \gamma \end{cases} \tag{11}$$

$$P(S_j = Failure \mid X_i = Failure) = (1 - \delta) + \delta\gamma \tag{12}$$

Under this mechanism, the influence of a component node on its child node is reformulated as expressed in Eq. (12). The original conditional probability is thus redefined to simultaneously capture fault occurrence, the effectiveness of fault-

tolerance mechanisms, and the residual system vulnerability following mitigation.

Focusing on the critical component HW4 in the model, a further investigation is conducted to examine how fault-tolerance capability affects overall system reliability. In the model, $P(S1=1|HW4=1)=0.7124$, if the system fault-tolerance capability δ for the component is set to 0.4, 0.6, and 0.8, the corresponding residual vulnerabilities γ are 0.2811, 0.5207, and 0.6405, representing weak, moderate, and strong fault-tolerance capabilities, respectively. By keeping parameter γ unchanged, the effects of varying the fault-tolerance capability δ on the propagation of component failures to subsystem failures, as well as on overall system reliability, are examined. The results are summarized in Table 11. As can be observed from the table, enhancing the system's fault-tolerance capability for critical components, alone with its ability to maintain functional operation following a fault, can effectively improve overall system reliability.

Table 11 Fault-Tolerance Analysis of HW4

γ \ δ	0.4	0.6	0.8
0.2811	0.759444	0.761175	0.762908
0.5207	0.758289	0.759444	0.760598
0.6405	0.757712	0.758578	0.759444

4.3.7 Uncertainty analysis

4.3.7.1 Conditional probability uncertainty analysis

Due to the high complexity of software–hardware coupled systems in intelligent ships, many nodes lack complete and sufficient statistical data. In this study, CPTs are constructed using a multi-source information fusion approach that integrates both operational statistics and expert knowledge. However, discrepancies in accuracy and reliability among different data sources inevitably introduce uncertainty into the model. On the one hand, the limited availability of operational data makes it difficult to fully capture the dynamic behavior of software–hardware interactions under diverse operating scenarios. On the other hand, expert knowledge, while valuable in compensating for data scarcity, is inherently influenced by individual experience and cognitive biases. Consequently, the CPTs become the primary source of uncertainty in the DBN model and constitute a key factor affecting the confidence level of the system reliability assessment.

By conducting a sensitivity analysis on the parameters of the conditional probability, applying a 10% perturbation to each conditional probability, the posterior interval of the system task reliability is [0.74571,0.77318]. The maximum and minimum values are both caused by the perturbation of the conditional probability P(T=1|S3=1,S4=1), with a sensitivity value of 0.01374. Other conditional probabilities with relatively high impact are P(SF6=1|SW6=1) (0.00893), P(SF7=1|SF6=1) (0.00552) and P(T=1|S3=1) (0.00538). This result indicates that conditional probability parameters directly associated with the task node, and corresponding to key nodes located at the downstream end of the fault propagation process, exert the greatest influence on system reliability.

Furthermore, a structured global perturbation analysis was performed to assess the impact of correlated uncertainty across CPTs. A ±2% perturbation factor was applied to the occurrence probabilities of failure states of individual child nodes conditioned on their parent states. The resulting changes were then propagated through joint probability relationships to derive the adjusted CPTs, while all other model parameters were kept unchanged. Based on the two perturbed CPT sets, the system task reliability was recalculated, yielding values of 0.75000 under the +2% perturbation and 0.768874 under the -2% perturbation.

Although both local and structured global perturbations of CPT parameters lead to quantitative changes in the estimated reliability, the overall reliability level remains within a relatively narrow and acceptable range, and no qualitative change in system reliability is observed. This indicates that, while expert elicitation inevitably introduces epistemic uncertainty into the CPTs, the proposed PN-DBN reliability model demonstrates satisfactory robustness against reasonable deviations in expert judgment, thereby supporting the credibility and practical applicability of the assessment results.

4.3.7.2 Software failure uncertainty analysis

In this study, software failures are modeled based on a NHPP, which has been widely used to characterize software failure behavior. In the reliability analysis of the

operational phase conducted in this paper, the software failure probability is approximated as a constant value representing the average number of residual faults during system operation. This treatment is consistent with common practice in software reliability engineering and allows the analysis to focus on the interaction mechanisms between software and hardware rather than on the software reliability growth process itself. However, given that NHPP assumes failures are observable and reproducible, it cannot fully account for unknown faults that are difficult to detect or non-reproducible. To address this limitation, a Beta distribution is introduced on top of the original fixed failure probability to represent uncertainty in software failures.

The Beta distribution is naturally defined on the interval [0,1], making it suitable for modeling bounded uncertainty and expressing incomplete knowledge of the true software failure rate. The failure probability estimated from the NHPP is treated as the mean value, and an uncertainty band of $\pm x$ around this mean is introduced using the Beta distribution. Specifically, the shape parameters α and β are derived using the method of moments, calculated based on the mean $\mu = \alpha / (\alpha + \beta)$ and the variance σ^2 determined by the 30% fluctuation range, thereby defining the specific profile of the probability density function. In this way, unknown software faults are captured through probabilistic uncertainty and propagated to the system level via the DBN model.

Assuming a 30% epistemic uncertainty in all software failures, that is, the true software failure probability may fluctuate by ±30% around the NHPP-based mean, the resulting variation interval of the software failure probability is calculated and incorporated into the reliability model. The corresponding system reliability fluctuates within the range [0.716603,0.804212], which quantifies the impact of non-deterministic and non-reproducible faults on overall system performance. By employing interval estimation to compensate for the constraints in processing non-deterministic failures, the analysis reveals that despite the inherent impact of software uncertainty, the system's reliability performance stays consistently at an acceptable level.

4.3.8 Scalability and computational efficiency analysis

During model computation, the CPTs are derived using the joint probability distribution of conditional probabilities. The probability distribution of a child node under the influence of a single parent-node failure is first determined, and then combined through conditional probability relationships to obtain the probability distribution of the child node under multiple parent-node failures. The computational burden of the model is mainly governed by the number of nodes and the maximum dimensionality of the CPTs. In this case study, the model contains a total of 39 nodes, of which 21 are child nodes. The child node with the largest number of parents is S4, which has nine parent nodes, resulting in a CPT with 2^9 entries. Consequently, the computational cost is constrained at the node level and does not grow exponentially with the total number of system components.

To further demonstrate the scalability of the proposed method, the plug-and-play capability of the framework is validated through local node refinement based on the original model. In the initial structure, as represented in Fig.15, the camera HW2 serves as an input to the SF1, thereby influencing other components in the system. An additional node HF15 is introduced to model failures occurring at the software-hardware interface. Its prior failure probability is set to 0.0001, and its effect is combined through the node “Inter” to represent the combined impact of HW2 and interface failures on the SF1 input. The original structural relationship $HW2 \to SF1$ is thus locally refined to $\{HW2, HF15\} \to inter \to SF1$, and the recalculated system reliability is 0.759421. Compared with the original model results, the system reliability changes only slightly, indicating that the addition of new nodes does not disrupt the stability of the existing structure.

These results demonstrate that the proposed PN-DBN modeling framework supports capability expansion through localized insertion without requiring complete model reconstruction or re-identification of all parameters. New failure mechanisms can therefore be incorporated in a plug-and-play manner, without restructuring the overall model, thereby demonstrating the method’s scalability and engineering applicability for complex ship system modeling.

4.4 Discussion

In intelligent systems, there is typically a high degree of coupling and complex interaction between software and hardware components. Conventional reliability analysis models often treat software and hardware as independent entities, evaluating their reliability in isolation, which fails to capture the intricate dependency and propagation mechanisms within the system. As a result, they often lead to an underestimation of the system's actual reliability.

This paper proposes a method that models software-hardware interactions based on the system's structural architecture, thereby improving the accuracy of system-level reliability assessments. Although this study focuses on autonomous navigation systems, the proposed approach is not limited to the autonomous navigation. Its modeling procedure, fault propagation extraction via PN and probabilistic reasoning via DBN, can be directly extended to other intelligent ship subsystems or to an integrated multi-subsystem model, provided that the system architecture and interaction logic are available. This proposed method is well-suited for complex systems characterized by high software-hardware coupling, task-oriented architecture, and clearly defined system structures, such as intelligent ships, unmanned systems, and intelligent manufacturing equipment.

However, several limitations remain when applying this approach to large-scale intelligent systems with highly complex architectures. First, as system scale increases, structural complexity inevitably grows due to the dense interactions among software and hardware components. By using PN to extract cross-domain failure propagation, the proposed framework avoids exhaustive enumeration of coupling relationships within the DBN, thereby mitigating the risk of combinatorial explosion. The computational cost is primarily governed by the number of nodes per time slice, state cardinalities, and the size of local conditional probability tables, meaning that inference complexity is constrained by localized dependency structures rather than the full system state space. This modular representation also enables incremental model extension

without global restructuring, which is essential for analyzing evolving intelligent systems.

Second, failure modeling in complex intelligent ship systems is inherently constrained by limited observability and insufficient data, making it difficult to comprehensively characterize all failure modes. In this study, a Noisy-OR formulation is incorporated at selected subsystem nodes as an engineering approximation, allowing implicit failure channels to be represented while preserving the explicit causal structure of the model. However, this treatment remains a phenomenological representation rather than a fully data-driven identification of underlying mechanisms. The framework still relies on predefined relationships and parameter assumptions, and future work will require data integration and parameter analysis so that these implicit failure representations can gradually evolve from expert-based approximations to evidence-based characterizations, thereby improving model fidelity during long-term operation.

Moreover, system components are represented using binary fault state (normal/faulty). This modeling choice is primarily made to ensure tractability and clarity of the proposed PN-DBN coupling framework, focusing on capturing the fault propagation logic and software-hardware interaction mechanisms at the system level. However, the PN-DBN structure can be naturally extended to multi-state models by introducing intermediate performance states in the DBN. For example, component degradation processes can be incorporated using multi-state Markov approaches, as demonstrated in Reference [30].

Despite these limitations, the primary value of the proposed methodology lies in its integration of software-hardware coupling into the reliability modeling process. This shifts the focus from identifying isolated hardware failures to capturing complex interaction-induced failures, providing a more robust and integrated basis for system maintenance. To further enhance the reliability of software-hardware coupled systems, it is essential not only to improve the inherent reliability of individual software and hardware components but also to predict potential fault propagation paths and optimize

fault-tolerant logic at their interfaces. This helps prevent interaction-induced failures and enhances the system's robustness against abnormal interactions.

5. Conclusion

This study addresses the challenge of strong software and hardware coupling in intelligent ship autonomous systems by analyzing coupling characteristics and fault propagation mechanisms. To overcome the limitations of traditional reliability models that assume independence, a novel modeling and evaluation method is proposed, integrating PN for fault propagation representation with DBN for probabilistic reasoning and dynamic evolution. This PN-DBN framework unifies dynamic fault propagation and system structural models, enabling comprehensive reliability assessment under software–hardware coupling.

By comparing the proposed approach with other embedded software-hardware coupling models and traditional reliability models, its practical value is demonstrated in that the approach can simultaneously represent the interaction mechanisms between software and hardware and perform quantitative reliability reasoning for complex systems.

A case study on the autonomous navigation task of a USV validates the effectiveness of the proposed method. Compared with the traditional model (system reliability: 0.8200), the proposed coupled method yielded a slightly lower value (0.7594), reflecting more realistic risks arising from software–hardware interactions. This result demonstrates that the method not only improves accuracy but also facilitates the early identification of weak links and critical coupling relationships, providing valuable guidance for reliability design and system optimization.

The proposed approach is applicable to a wide range of complex, mission-driven CPSs characterized by high levels of software–hardware integration, such as intelligent ships, unmanned systems, and smart manufacturing platforms. By optimizing fault-tolerant mechanisms at software–hardware interfaces and introducing functional redundancy, the propagation of local anomalies can be mitigated, preventing them from

escalating into mission-level failures.

Future work will focus on integrating data-driven learning mechanisms to reduce the subjectivity of parameter estimation and on developing standardized software tools to support automated modeling and inference. Moreover, the proposed method will be integrated into Model-Based Systems Engineering (MBSE) workflows to support system operation and maintenance. These efforts aim to enhance the scalability and practical applicability of the proposed framework for reliability analysis of complex systems.

CRediT authorship contribution statement

Xiaofang Luo: Conceptualization, Validation, Supervision, Writing-Original draft, Funding acquisition. **Linghui Guo:** Methodology, Data curation, Writing-Original draft. **Xiangdong Ma:** Writing-Review & Editing, Visualization. **Xu Bai:** Layout design, Resources, Writing-Review & Editing. **Jingling Li**: Methodology, Supervision.

Declaration of competing interest

The authors declare that they have no known completing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential.

Acknowledgment

The research was supported by the following funding program: National Natural Science Foundation of China (72571119) and National Key R&D Program of China (No. 2024YFC2816400). The authors wish to express their sincere gratitude to the reviewers for their valuable and constructive comments.

References

[1] Zhang W J, Zhang Y J,Zhang C. Research on risk assessment of maritime autonomous surface ships based on catastrophe theory[J].Reliability Engineering and System Safety

2024,244:109946.

[2] Li H Q, Meng X K, Zhang W J, et al. A real-time reliability assessment framework for marine mechanical equipment integrating machine learning and physical knowledge: Toward applications in maritime autonomous surface ships[J].Reliability Engineering and System Safety 2026; 271: 112233.

[3] Lu H, Zhang Y, Zhang C, et al. A multi-sensor fusion approach for maritime autonomous surface ships berthing navigation perception[J]. Ocean Engineering 2025; 316: 119965.

[4] Wang S, Zhang Y, Zhang X, et al. A novel maritime autonomous navigation decision-making system: Modeling, integration, and real ship trial[J]. Expert Systems with Applications 2023; 222: 119825.

[5] Yousaf A, Amro A, Kwa P T H, et al. Cyber risk assessment of cyber-enabled autonomous cargo vessel[J]. International Journal of Critical Infrastructure Protection 2024; 46: 100695.

[6] Luo X F, Li Y S, Bai X, et al. A novel approach based on fault tree analysis and Bayesian network for multi-state reliability analysis of complex equipment systems [J]. Institution of Mech Eng Part O: J Risk Reliab 2023:1-27.

[7] Lyu H, Li Z, Qiao X, et al. Reliability analysis for multi-component system considering failure propagation and dependent competing failure process[J]. Reliability Engineering & System Safety 2025, 259: 110930.

[8] Yang J F, Chen J, and Wang X B. EM algorithm for estimating reliability of multi-release open source software based on general masked data[J]. IEEE Access 2021; 9:18890–18903.

[9] Song J Y, Zhao H D, Li X L, Yang Y, et al. A new software failure analysis method based on the system reliability modeling[C] // IEEE. 2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC). Chongqing: IEEE\ 2019: 1143 - 1149.

[10] Li Z H, Di Z, Bing H, et al. Risk and reliability analysis for maritime autonomous surface ship: A bibliometric review of literature from 2015 to 2022[J]. Accident Analysis and Prevention 2023;187:107090-107090.

[11] Kanoun K, Ortalo - Borrel M. Fault - tolerant system dependability - explicit modeling of

hardware and software component‐interactions[J]. IEEE Transactions on Reliability 2000 49(4):363‐376.

[12] Yang X, Zhou T, Utne B I, et al. A framework for quantification of coupling risk in the transition between operational modes of MASS[J]. Reliability Engineering and System Safety 2025,264(PB):111436.

[13] Varuvel A G, Prasath R. Quantification of effects of degraded hardware on the execution of self-healing software using Continuous Time Markov Chain and Reliability[J]. Computers and Electrical Engineering 2024;120: 109598.

[14] Andrade T N C, Lima G, Lima V M C, et al. On the impact of hardware-related events on the execution of real-time programs[J]. Design Automation for Embedded Systems 2023;27(4): 275-302.

[15] Zhu M, Pham H. A novel system reliability modeling of hardware, software, and interactions of hardware and software[J]. Mathematics 2019; 7(11): 1049.

[16] Zheng Z T, Yang J F, Huang J Y. Software-hardware embedded system reliability modeling with failure dependency and masked data[J]. Computers & Industrial Engineering 2023; 186: 109746.

[17] John Y M, Sanusi A, Yusuf I, et al. Reliability analysis of multi-hardware‐software system with failure interaction[J]. Journal of Computational and Cognitive Engineering 2023; 2(1): 38-46.

[18] Huang P, Tang C W, Sun Y, et al. A coupled failure mode and effects analysis method for hardware-software failure interactions in embedded systems of intelligent equipment[J]. Reliability Engineering and System Safety 2026,270:112184.

[19] Zeng Y H, Xing L D, Zhang Q, et al. An analytical method for reliability analysis of hardware‐software co‐design system[J]. Quality and Reliability Engineering International 2019; 35(1): 165-178.

[20] Seo J, Kang H G, Lee E C, et al. Experimental approach to evaluate software reliability in hardware-software integrated environment[J]. Nuclear Engineering and Technology 2020; 52(7): 1462-1470.

[21] Stewart D, Liu J, Cofer D, et al. AADL-Based safety analysis using formal methods applied to aircraft digital systems[J].Reliability Engineering and System Safety 2021,213:107649.

[22] Zheng X H, Yao W, Xu Y C, et al. Algorithms for Bayesian network modeling and reliability inference of complex multistate systems with common cause failure[J].Reliability Engineering and System Safety 2024,241:109663.

[23] Byun S, Papaelias M, MÃ¡rquez F P G, et al. Fault-tree-analysis-based health monitoring for autonomous underwater vehicle[J]. Journal of Marine Science and Engineering 2022; 10(12): 1855.

[24] Abaei M M, Hekkenberg R, BahooToroody A. A multinomial process tree for reliability assessment of machinery in autonomous ships[J]. Reliability Engineering & System Safety 2021; 210: 107484.

[25] Li J K, Wang X, Wang N, et al. A novel closed-loop feedback system reliability analysis methodology based on GO methodology and modified Bayesian networks[J].Reliability Engineering and System Safety 2026,270:112174.

[26] Sinha S, Goyal N K, Mall R. Early prediction of reliability and availability of combined hardware-software systems based on functional failures[J]. Journal of Systems Architecture 2019; 92: 23-38.

[27] Shorthill T, Bao H, Zhang H, et al. A novel approach for software reliability analysis of digital instrumentation and control systems in nuclear power plants[J]. Annals of Nuclear Energy 2021; 158: 108260.

[28] Xu Y X, Li K P, Liu Y Y. Quantitative analysis of risk propagation in urban rail transit: A novel ensemble learning method based on the structure of Bayesian Network[J]. Reliability Engineering & System Safety 2025: 111280.

[29] Kumari R, Naick B K, Ghosh D. Reliability assessment of distribution system using Petri net for enhancement of situational awareness[J]. Electric Power Systems Research 2023; 224: 109739.

[30] Luo X F, Guo L H, Bai X, et al. A multi-phase mission success evaluation approach for maritime autonomous surface ships considering equipment performance degradation and

system composition changes[J]. Reliability Engineering & System Safety 2025; 254: 110604.

[31] Goel A L. Time-dependent error-detection rate model for software and other performance measures[J]. IEEE Transactions on Reliability 1979; 28(3): 465-484.

[32] Yang C, Liu H, Wang Y, et al. Task reliability evaluation method for autonomous intelligent system based on multi-agent approach[J]. Chinese Journal of Ship Research 2025; 20(2): 335-349.

[33] Wang Y, Li W, Lu J. Reliability Analysis of Phasor Measurement Unit Using Hierarchical Markov Modeling[J]. Electric Power Components and Systems 2009;37(5):517-532.

Appendix

Appendix Table 1 CPT of SF1 and SF2

Child node						SF1		Child node						SF2	
HW9	HW1	HW2	HW3	HW4	SW1	P(X)=0	P(X)=1	HW9	HW5	HW6	HW7	HW8	SW2	P(X)=0	P(X)=1
0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
0	0	0	0	0	1	0.3895	0.6105	0	0	0	0	0	1	0.2001	0.7999
0	0	0	0	1	0	0.3856	0.6144	0	0	0	0	1	0	0.2542	0.7458
0	0	0	0	1	1	0.1502	0.8498	0	0	0	0	1	1	0.0509	0.9491
0	0	0	1	0	0	0.3580	0.6420	0	0	0	1	0	0	0.1341	0.8659
0	0	0	1	0	1	0.1394	0.8606	0	0	0	1	0	1	0.0268	0.9732
0	0	0	1	1	0	0.1380	0.8620	0	0	0	1	1	0	0.0341	0.9659
0	0	0	1	1	1	0.0538	0.9462	0	0	0	1	1	1	0.0068	0.9932
0	0	1	0	0	0	0.2570	0.7430	0	0	1	0	0	0	0.1125	0.8875
0	0	1	0	0	1	0.1001	0.8999	0	0	1	0	0	1	0.0225	0.9775
0	0	1	0	1	0	0.0991	0.9009	0	0	1	0	1	0	0.0286	0.9714
0	0	1	0	1	1	0.0386	0.9614	0	0	1	0	1	1	0.0057	0.9943
0	0	1	1	0	0	0.0920	0.9080	0	0	1	1	0	0	0.0151	0.9849
0	0	1	1	0	1	0.0358	0.9642	0	0	1	1	0	1	0.0030	0.9970
0	0	1	1	1	0	0.0355	0.9645	0	0	1	1	1	0	0.0038	0.9962
0	0	1	1	1	1	0.0138	0.9862	0	0	1	1	1	1	0.0008	0.9992
0	1	0	0	0	0	0.3884	0.6116	0	1	0	0	0	0	0.1955	0.8045
0	1	0	0	0	1	0.1513	0.8487	0	1	0	0	0	1	0.0391	0.9609
0	1	0	0	1	0	0.1498	0.8502	0	1	0	0	1	0	0.0497	0.9503
0	1	0	0	1	1	0.0583	0.9417	0	1	0	0	1	1	0.0099	0.9901
0	1	0	1	0	0	0.1391	0.8609	0	1	0	1	0	0	0.0262	0.9738
0	1	0	1	0	1	0.0542	0.9458	0	1	0	1	0	1	0.0052	0.9948
0	1	0	1	1	0	0.0536	0.9464	0	1	0	1	1	0	0.0067	0.9933
0	1	0	1	1	1	0.0209	0.9791	0	1	0	1	1	1	0.0013	0.9987
0	1	1	0	0	0	0.0998	0.9002	0	1	1	0	0	0	0.0220	0.9780

0	1	1	0	0	1	0.0389	0.9611	0	1	1	0	0	1	0.0044	0.9956
0	1	1	0	1	0	0.0385	0.9615	0	1	1	0	1	0	0.0056	0.9944
0	1	1	0	1	1	0.0150	0.9850	0	1	1	0	1	1	0.0011	0.9989
0	1	1	1	0	0	0.0357	0.9643	0	1	1	1	0	0	0.0030	0.9970
0	1	1	1	0	1	0.0139	0.9861	0	1	1	1	0	1	0.0006	0.9994
0	1	1	1	1	0	0.0138	0.9862	0	1	1	1	1	0	0.0008	0.9992
0	1	1	1	1	1	0.0054	0.9946	0	1	1	1	1	1	0.0002	0.9998
1	0	0	0	0	0	0.1058	0.8942	1	0	0	0	0	0	0.1058	0.8942
1	0	0	0	0	1	0.0412	0.9588	1	0	0	0	0	1	0.0212	0.9788
1	0	0	0	1	0	0.0408	0.9592	1	0	0	0	1	0	0.0269	0.9731
1	0	0	0	1	1	0.0159	0.9841	1	0	0	0	1	1	0.0054	0.9946
1	0	0	1	0	0	0.0379	0.9621	1	0	0	1	0	0	0.0142	0.9858
1	0	0	1	0	1	0.0148	0.9852	1	0	0	1	0	1	0.0028	0.9972
1	0	0	1	1	0	0.0146	0.9854	1	0	0	1	1	0	0.0036	0.9964
1	0	0	1	1	1	0.0057	0.9943	1	0	0	1	1	1	0.0007	0.9993
1	0	1	0	0	0	0.0272	0.9728	1	0	1	0	0	0	0.0119	0.9881
1	0	1	0	0	1	0.0106	0.9894	1	0	1	0	0	1	0.0024	0.9976
1	0	1	0	1	0	0.0105	0.9895	1	0	1	0	1	0	0.0030	0.9970
1	0	1	0	1	1	0.0041	0.9959	1	0	1	0	1	1	0.0006	0.9994
1	0	1	1	0	0	0.0097	0.9903	1	0	1	1	0	0	0.0016	0.9984
1	0	1	1	0	1	0.0038	0.9962	1	0	1	1	0	1	0.0003	0.9997
1	0	1	1	1	0	0.0038	0.9962	1	0	1	1	1	0	0.0004	0.9996
1	0	1	1	1	1	0.0015	0.9985	1	0	1	1	1	1	0.0001	0.9999
1	1	0	0	0	0	0.0411	0.9589	1	1	0	0	0	0	0.0207	0.9793
1	1	0	0	0	1	0.0160	0.9840	1	1	0	0	0	1	0.0041	0.9959
1	1	0	0	1	0	0.0159	0.9841	1	1	0	0	1	0	0.0053	0.9947
1	1	0	0	1	1	0.0062	0.9938	1	1	0	0	1	1	0.0011	0.9989
1	1	0	1	0	0	0.0147	0.9853	1	1	0	1	0	0	0.0028	0.9972
1	1	0	1	0	1	0.0057	0.9943	1	1	0	1	0	1	0.0006	0.9994
1	1	0	1	1	0	0.0057	0.9943	1	1	0	1	1	0	0.0007	0.9993
1	1	0	1	1	1	0.0022	0.9978	1	1	0	1	1	1	0.0001	0.9999
1	1	1	0	0	0	0.0106	0.9894	1	1	1	0	0	0	0.0023	0.9977
1	1	1	0	0	1	0.0041	0.9959	1	1	1	0	0	1	0.0005	0.9995
1	1	1	0	1	0	0.0041	0.9959	1	1	1	0	1	0	0.0006	0.9994
1	1	1	0	1	1	0.0016	0.9984	1	1	1	0	1	1	0.0001	0.9999
1	1	1	1	0	0	0.0038	0.9962	1	1	1	1	0	0	0.0003	0.9997
1	1	1	1	0	1	0.0015	0.9985	1	1	1	1	0	1	0.0001	0.9999
1	1	1	1	1	0	0.0015	0.9985	1	1	1	1	1	0	0.0001	0.9999
1	1	1	1	1	1	0.0006	0.9994	1	1	1	1	1	1	0.0000	1.0000

Appendix Table 2 CPT of S1 and S2

Child node				S1		Child node				S2	
HW5	HW6	HW7	HW8	P(X)=0	P(X)=1	HW5	HW6	HW7	HW8	P(X)=0	P(X)=1

0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	1	0.2876	0.7124	0	0	0	1	0.2588	0.7412
0	0	1	0	0.2588	0.7412	0	0	1	0	0.3478	0.6522
0	0	1	1	0.0744	0.9256	0	0	1	1	0.0900	0.9100
0	1	0	0	0.2458	0.7542	0	1	0	0	0.3146	0.6854
0	1	0	1	0.0707	0.9293	0	1	0	1	0.0814	0.9186
0	1	1	0	0.0636	0.9364	0	1	1	0	0.1094	0.8906
0	1	1	1	0.0183	0.9817	0	1	1	1	0.0283	0.9717
1	0	0	0	0.1425	0.8575	1	0	0	0	0.2889	0.7111
1	0	0	1	0.0410	0.9590	1	0	0	1	0.0748	0.9252
1	0	1	0	0.0369	0.9631	1	0	1	0	0.1005	0.8995
1	0	1	1	0.0106	0.9894	1	0	1	1	0.0260	0.9740
1	1	0	0	0.0350	0.9650	1	1	0	0	0.0909	0.9091
1	1	0	1	0.0101	0.9899	1	1	0	1	0.0235	0.9765
1	1	1	0	0.0091	0.9909	1	1	1	0	0.0316	0.9684
1	1	1	1	0.0026	0.9974	1	1	1	1	0.0082	0.9918

Appendix Table 3 CPT of SF3 and SF6

Child node				SF3		Child node				SF6	
HW9	SF1	SF2	SW3	P(X)=0	P(X)=1	HW9	SF2	SF5	SW6	P(X)=0	P(X)=1
0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	1	0.1414	0.8586	0	0	0	1	0.1446	0.8554
0	0	1	0	0.3414	0.6586	0	0	1	0	0.2941	0.7059
0	0	1	1	0.0483	0.9517	0	0	1	1	0.0425	0.9575
0	1	0	0	0.1437	0.8563	0	1	0	0	0.3187	0.6813
0	1	0	1	0.0203	0.9797	0	1	0	1	0.0461	0.9539
0	1	1	0	0.0491	0.9509	0	1	1	0	0.0938	0.9062
0	1	1	1	0.0069	0.9931	0	1	1	1	0.0136	0.9864
1	0	0	0	0.1058	0.8942	1	0	0	0	0.1058	0.8942
1	0	0	1	0.0150	0.9850	1	0	0	1	0.0153	0.9847
1	0	1	0	0.0361	0.9639	1	0	1	0	0.0311	0.9689
1	0	1	1	0.0051	0.9949	1	0	1	1	0.0045	0.9955
1	1	0	0	0.0152	0.9848	1	1	0	0	0.0337	0.9663
1	1	0	1	0.0021	0.9979	1	1	0	1	0.0049	0.9951
1	1	1	0	0.0052	0.9948	1	1	1	0	0.0099	0.9901
1	1	1	1	0.0007	0.9993	1	1	1	1	0.0014	0.9986

Appendix Table 4 CPT of S4 and T

Child node				S4		Child node				T	
HF11	HF12	HF13	HF14	P(X)=0	P(X)=1	S1	S2	S3	S4	P(X)=0	P(X)=1
0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	1	0.1362	0.8638	0	0	0	1	0.2919	0.7081
0	0	1	0	0.1411	0.8589	0	0	1	0	0.2643	0.7357

0	0	1	1	0.0192	0.9808	0	0	1	1	0.0771	0.9229
0	1	0	0	0.1524	0.8476	0	1	0	0	0.3058	0.6942
0	1	0	1	0.0208	0.9792	0	1	0	1	0.0893	0.9107
0	1	1	0	0.0215	0.9785	0	1	1	0	0.0808	0.9192
0	1	1	1	0.0029	0.9971	0	1	1	1	0.0236	0.9764
1	0	0	0	0.0876	0.9124	1	0	0	0	0.3448	0.6552
1	0	0	1	0.0119	0.9881	1	0	0	1	0.1007	0.8993
1	0	1	0	0.0124	0.9876	1	0	1	0	0.0911	0.9089
1	0	1	1	0.0017	0.9983	1	0	1	1	0.0266	0.9734
1	1	0	0	0.0134	0.9866	1	1	0	0	0.1054	0.8946
1	1	0	1	0.0018	0.9982	1	1	0	1	0.0308	0.9692
1	1	1	0	0.0019	0.9981	1	1	1	0	0.0279	0.9721
1	1	1	1	0.0003	0.9997	1	1	1	1	0.0081	0.9919

Appendix Table 5 CPT of SF4, SF5, SF7, HF11, HF12, HF13, HF14, RE1, RE2, HW1, HW3, HW5

Child node			SF4		Child node			SF5	
HW9	SF3	SW4	P(X)=0	P(X)=1	HW10	SF4	SW5	P(X)=0	P(X)=1
0	0	0	1	0	0	0	0	1	0
0	0	1	0.1379	0.8621	0	0	1	0.1549	0.8451
0	1	0	0.2545	0.7455	0	1	0	0.2345	0.7655
0	1	1	0.0351	0.9649	0	1	1	0.0363	0.9637
1	0	0	0.1058	0.8942	1	0	0	0.1058	0.8942
1	0	1	0.0146	0.9854	1	0	1	0.0164	0.9836
1	1	0	0.0269	0.9731	1	1	0	0.0248	0.9752
1	1	1	0.0037	0.9963	1	1	1	0.0038	0.9962

Child node			SF5		Child node			HF11	
HW10	SF6	SW7	P(X)=0	P(X)=1	SF7		HW11	P(X)=0	P(X)=1
0	0	0	1	0	0		0	1	0
0	0	1	0.1877	0.8123	0		1	0.0876	0.9124
0	1	0	0.2104	0.7896	1		0	0.1045	0.8955
0	1	1	0.0395	0.9605	1		1	0.0092	0.9908
1	0	0	0.0985	0.9015	Child nod			HF12	
1	0	1	0.0185	0.9815	SF7		HW12	P(X)=0	P(X)=1
1	1	0	0.0207	0.9793	0		0	1	0
1	1	1	0.0039	0.9961	0		1	0.1521	0.8479
Child node			HF13		1		0	0.1045	0.8955
SF7		HW13	P(X)=0	P(X)=1	1		1	0.0159	0.9841
0		0	1	0	Child node			HF14	
0		1	0.0215	0.9785	SF7		HW14	P(X)=0	P(X)=1
1		0	0.1045	.0.8955	0		0	1	0
1		1	0.0023	0.9977	0		1	0.0363	0.9637
Child node			RE1		1		0	0.1045	0.8955

HW11	0.9780	0.0298	1	1	0.0038	0.9962
Child node	RE2		Child node		HW5	
HW11	0.9414	0.0586	RE1(t-1)	RE2(t-1)	P(X)=0	P(X)=1
Child node	HW3		0	0	1	0
RE1(t-1)	0.9433	0.0597	0	1	0.9974	0.0026
Child node	HW1		1	0	0.9983	0.0017
RE2(t-1)	0.9968	0.0032	1	1	0.9957	0.0043

Appendix Table 6 CPT of S4

Child node									S4	
SF1	SF2	SF3	SF4	SF5	SF6	SF7	HW9	HW10	P(X)=0	P(X)=1
0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0.2897	0.7103
0	0	0	0	0	0	0	1	0	0.2137	0.7863
0	0	0	0	0	0	0	1	1	0.0619	0.9381
0	0	0	0	0	0	1	0	0	0.2481	0.7519
0	0	0	0	0	0	1	0	1	0.0719	0.9281
0	0	0	0	0	0	1	1	0	0.0530	0.9470
0	0	0	0	0	0	1	1	1	0.0154	0.9846
0	0	0	0	0	1	0	0	0	0.3167	0.6833
0	0	0	0	0	1	0	0	1	0.0918	0.9082
0	0	0	0	0	1	0	1	0	0.0677	0.9323
0	0	0	0	0	1	0	1	1	0.0196	0.9804
0	0	0	0	0	1	1	0	0	0.0786	0.9214
0	0	0	0	0	1	1	0	1	0.0228	0.9772
0	0	0	0	0	1	1	1	0	0.0168	0.9832
0	0	0	0	0	1	1	1	1	0.0049	0.9951
0	0	0	0	1	0	0	0	0	0.2875	0.7125
0	0	0	0	1	0	0	0	1	0.0833	0.9167
0	0	0	0	1	0	0	1	0	0.0614	0.9386
0	0	0	0	1	0	0	1	1	0.0178	0.9822
0	0	0	0	1	0	1	0	0	0.0713	0.9287
0	0	0	0	1	0	1	0	1	0.0207	0.9793
0	0	0	0	1	0	1	1	0	0.0152	0.9848
0	0	0	0	1	0	1	1	1	0.0044	0.9956
0	0	0	0	1	1	0	0	0	0.0910	0.9090
0	0	0	0	1	1	0	0	1	0.0264	0.9736
0	0	0	0	1	1	0	1	0	0.0195	0.9805
0	0	0	0	1	1	0	1	1	0.0056	0.9944
0	0	0	0	1	1	1	0	0	0.0226	0.9774
0	0	0	0	1	1	1	0	1	0.0065	0.9935
0	0	0	0	1	1	1	1	0	0.0048	0.9952
0	0	0	0	1	1	1	1	1	0.0014	0.9986

0	0	0	1	0	0	0	0	0	0.2548	0.7452
0	0	0	1	0	0	0	0	1	0.0738	0.9262
0	0	0	1	0	0	0	1	0	0.0544	0.9456
0	0	0	1	0	0	0	1	1	0.0158	0.9842
0	0	0	1	0	0	1	0	0	0.0632	0.9368
0	0	0	1	0	0	1	0	1	0.0183	0.9817
0	0	0	1	0	0	1	1	0	0.0135	0.9865
0	0	0	1	0	0	1	1	1	0.0039	0.9961
0	0	0	1	0	1	0	0	0	0.0807	0.9193
0	0	0	1	0	1	0	0	1	0.0234	0.9766
0	0	0	1	0	1	0	1	0	0.0172	0.9828
0	0	0	1	0	1	0	1	1	0.0050	0.9950
0	0	0	1	0	1	1	0	0	0.0200	0.9800
0	0	0	1	0	1	1	0	1	0.0058	0.9942
0	0	0	1	0	1	1	1	0	0.0043	0.9957
0	0	0	1	0	1	1	1	1	0.0012	0.9988
0	0	0	1	1	0	0	0	0	0.0732	0.9268
0	0	0	1	1	0	0	0	1	0.0212	0.9788
0	0	0	1	1	0	0	1	0	0.0157	0.9843
0	0	0	1	1	0	0	1	1	0.0045	0.9955
0	0	0	1	1	0	1	0	0	0.0182	0.9818
0	0	0	1	1	0	1	0	1	0.0053	0.9947
0	0	0	1	1	0	1	1	0	0.0039	0.9961
0	0	0	1	1	0	1	1	1	0.0011	0.9989
0	0	0	1	1	1	0	0	0	0.0232	0.9768
0	0	0	1	1	1	0	0	1	0.0067	0.9933
0	0	0	1	1	1	0	1	0	0.0050	0.9950
0	0	0	1	1	1	0	1	1	0.0014	0.9986
0	0	0	1	1	1	1	0	0	0.0058	0.9942
0	0	0	1	1	1	1	0	1	0.0017	0.9983
0	0	0	1	1	1	1	1	0	0.0012	0.9988
0	0	0	1	1	1	1	1	1	0.0004	0.9996
0	0	1	0	0	0	0	0	0	0.3459	0.6541
0	0	1	0	0	0	0	0	1	0.1002	0.8998
0	0	1	0	0	0	0	1	0	0.0739	0.9261
0	0	1	0	0	0	0	1	1	0.0214	0.9786
0	0	1	0	0	0	1	0	0	0.0858	0.9142
0	0	1	0	0	0	1	0	1	0.0249	0.9751
0	0	1	0	0	0	1	1	0	0.0183	0.9817
0	0	1	0	0	0	1	1	1	0.0053	0.9947
0	0	1	0	0	1	0	0	0	0.1096	0.8904
0	0	1	0	0	1	0	0	1	0.0317	0.9683

0	0	1	0	0	1	0	1	0	0.0234	0.9766
0	0	1	0	0	1	0	1	1	0.0068	0.9932
0	0	1	0	0	1	1	0	0	0.0272	0.9728
0	0	1	0	0	1	1	0	1	0.0079	0.9921
0	0	1	0	0	1	1	1	0	0.0058	0.9942
0	0	1	0	0	1	1	1	1	0.0017	0.9983
0	0	1	0	1	0	0	0	0	0.0994	0.9006
0	0	1	0	1	0	0	0	1	0.0288	0.9712
0	0	1	0	1	0	0	1	0	0.0212	0.9788
0	0	1	0	1	0	0	1	1	0.0062	0.9938
0	0	1	0	1	0	1	0	0	0.0247	0.9753
0	0	1	0	1	0	1	0	1	0.0071	0.9929
0	0	1	0	1	0	1	1	0	0.0053	0.9947
0	0	1	0	1	0	1	1	1	0.0015	0.9985
0	0	1	0	1	1	0	0	0	0.0315	0.9685
0	0	1	0	1	1	0	0	1	0.0091	0.9909
0	0	1	0	1	1	0	1	0	0.0067	0.9933
0	0	1	0	1	1	0	1	1	0.0019	0.9981
0	0	1	0	1	1	1	0	0	0.0078	0.9922
0	0	1	0	1	1	1	0	1	0.0023	0.9977
0	0	1	0	1	1	1	1	0	0.0017	0.9983
0	0	1	0	1	1	1	1	1	0.0005	0.9995
0	0	1	1	0	0	0	0	0	0.0881	0.9119
0	0	1	1	0	0	0	0	1	0.0255	0.9745
0	0	1	1	0	0	0	1	0	0.0188	0.9812
0	0	1	1	0	0	0	1	1	0.0055	0.9945
0	0	1	1	0	0	1	0	0	0.0219	0.9781
0	0	1	1	0	0	1	0	1	0.0063	0.9937
0	0	1	1	0	0	1	1	0	0.0047	0.9953
0	0	1	1	0	0	1	1	1	0.0014	0.9986
0	0	1	1	0	1	0	0	0	0.0279	0.9721
0	0	1	1	0	1	0	0	1	0.0081	0.9919
0	0	1	1	0	1	0	1	0	0.0060	0.9940
0	0	1	1	0	1	0	1	1	0.0017	0.9983
0	0	1	1	0	1	1	0	0	0.0069	0.9931
0	0	1	1	0	1	1	0	1	0.0020	0.9980
0	0	1	1	0	1	1	1	0	0.0015	0.9985
0	0	1	1	0	1	1	1	1	0.0004	0.9996
0	0	1	1	1	0	0	0	0	0.0253	0.9747
0	0	1	1	1	0	0	0	1	0.0073	0.9927
0	0	1	1	1	0	0	1	0	0.0054	0.9946
0	0	1	1	1	0	0	1	1	0.0016	0.9984

0	0	1	1	1	0	1	0	0	0.0063	0.9937
0	0	1	1	1	0	1	0	1	0.0018	0.9982
0	0	1	1	1	0	1	1	0	0.0013	0.9987
0	0	1	1	1	0	1	1	1	0.0004	0.9996
0	0	1	1	1	1	0	0	0	0.0080	0.9920
0	0	1	1	1	1	0	0	1	0.0023	0.9977
0	0	1	1	1	1	0	1	0	0.0017	0.9983
0	0	1	1	1	1	0	1	1	0.0005	0.9995
0	0	1	1	1	1	1	0	0	0.0020	0.9980
0	0	1	1	1	1	1	0	1	0.0006	0.9994
0	0	1	1	1	1	1	1	0	0.0004	0.9996
0	0	1	1	1	1	1	1	1	0.0001	0.9999
0	1	0	0	0	0	0	0	0	0.4450	0.5550
0	1	0	0	0	0	0	0	1	0.1289	0.8711
0	1	0	0	0	0	0	1	0	0.0951	0.9049
0	1	0	0	0	0	0	1	1	0.0276	0.9724
0	1	0	0	0	0	1	0	0	0.1104	0.8896
0	1	0	0	0	0	1	0	1	0.0320	0.9680
0	1	0	0	0	0	1	1	0	0.0236	0.9764
0	1	0	0	0	0	1	1	1	0.0068	0.9932
0	1	0	0	0	1	0	0	0	0.1410	0.8590
0	1	0	0	0	1	0	0	1	0.0408	0.9592
0	1	0	0	0	1	0	1	0	0.0301	0.9699
0	1	0	0	0	1	0	1	1	0.0087	0.9913
0	1	0	0	0	1	1	0	0	0.0350	0.9650
0	1	0	0	0	1	1	0	1	0.0101	0.9899
0	1	0	0	0	1	1	1	0	0.0075	0.9925
0	1	0	0	0	1	1	1	1	0.0022	0.9978
0	1	0	0	1	0	0	0	0	0.1279	0.8721
0	1	0	0	1	0	0	0	1	0.0371	0.9629
0	1	0	0	1	0	0	1	0	0.0273	0.9727
0	1	0	0	1	0	0	1	1	0.0079	0.9921
0	1	0	0	1	0	1	0	0	0.0317	0.9683
0	1	0	0	1	0	1	0	1	0.0092	0.9908
0	1	0	0	1	0	1	1	0	0.0068	0.9932
0	1	0	0	1	0	1	1	1	0.0020	0.9980
0	1	0	0	1	1	0	0	0	0.0405	0.9595
0	1	0	0	1	1	0	0	1	0.0117	0.9883
0	1	0	0	1	1	0	1	0	0.0087	0.9913
0	1	0	0	1	1	0	1	1	0.0025	0.9975
0	1	0	0	1	1	1	0	0	0.0101	0.9899
0	1	0	0	1	1	1	0	1	0.0029	0.9971

0	1	0	0	1	1	1	1	0	0.0021	0.9979
0	1	0	0	1	1	1	1	1	0.0006	0.9994
0	1	0	1	0	0	0	0	0	0.1134	0.8866
0	1	0	1	0	0	0	0	1	0.0328	0.9672
0	1	0	1	0	0	0	1	0	0.0242	0.9758
0	1	0	1	0	0	0	1	1	0.0070	0.9930
0	1	0	1	0	0	1	0	0	0.0281	0.9719
0	1	0	1	0	0	1	0	1	0.0081	0.9919
0	1	0	1	0	0	1	1	0	0.0060	0.9940
0	1	0	1	0	0	1	1	1	0.0017	0.9983
0	1	0	1	0	1	0	0	0	0.0359	0.9641
0	1	0	1	0	1	0	0	1	0.0104	0.9896
0	1	0	1	0	1	0	1	0	0.0077	0.9923
0	1	0	1	0	1	0	1	1	0.0022	0.9978
0	1	0	1	0	1	1	0	0	0.0089	0.9911
0	1	0	1	0	1	1	0	1	0.0026	0.9974
0	1	0	1	0	1	1	1	0	0.0019	0.9981
0	1	0	1	0	1	1	1	1	0.0006	0.9994
0	1	0	1	1	0	0	0	0	0.0326	0.9674
0	1	0	1	1	0	0	0	1	0.0094	0.9906
0	1	0	1	1	0	0	1	0	0.0070	0.9930
0	1	0	1	1	0	0	1	1	0.0020	0.9980
0	1	0	1	1	0	1	0	0	0.0081	0.9919
0	1	0	1	1	0	1	0	1	0.0023	0.9977
0	1	0	1	1	0	1	1	0	0.0017	0.9983
0	1	0	1	1	0	1	1	1	0.0005	0.9995
0	1	0	1	1	1	0	0	0	0.0103	0.9897
0	1	0	1	1	1	0	0	1	0.0030	0.9970
0	1	0	1	1	1	0	1	0	0.0022	0.9978
0	1	0	1	1	1	0	1	1	0.0006	0.9994
0	1	0	1	1	1	1	0	0	0.0026	0.9974
0	1	0	1	1	1	1	0	1	0.0007	0.9993
0	1	0	1	1	1	1	1	0	0.0005	0.9995
0	1	0	1	1	1	1	1	1	0.0002	0.9998
0	1	1	0	0	0	0	0	0	0.1539	0.8461
0	1	1	0	0	0	0	0	1	0.0446	0.9554
0	1	1	0	0	0	0	1	0	0.0329	0.9671
0	1	1	0	0	0	0	1	1	0.0095	0.9905
0	1	1	0	0	0	1	0	0	0.0382	0.9618
0	1	1	0	0	0	1	0	1	0.0111	0.9889
0	1	1	0	0	0	1	1	0	0.0082	0.9918
0	1	1	0	0	0	1	1	1	0.0024	0.9976

0	1	1	0	0	1	0	0	0	0.0488	0.9512
0	1	1	0	0	1	0	0	1	0.0141	0.9859
0	1	1	0	0	1	0	1	0	0.0104	0.9896
0	1	1	0	0	1	0	1	1	0.0030	0.9970
0	1	1	0	0	1	1	0	0	0.0121	0.9879
0	1	1	0	0	1	1	0	1	0.0035	0.9965
0	1	1	0	0	1	1	1	0	0.0026	0.9974
0	1	1	0	0	1	1	1	1	0.0007	0.9993
0	1	1	0	1	0	0	0	0	0.0442	0.9558
0	1	1	0	1	0	0	0	1	0.0128	0.9872
0	1	1	0	1	0	0	1	0	0.0095	0.9905
0	1	1	0	1	0	0	1	1	0.0027	0.9973
0	1	1	0	1	0	1	0	0	0.0110	0.9890
0	1	1	0	1	0	1	0	1	0.0032	0.9968
0	1	1	0	1	0	1	1	0	0.0023	0.9977
0	1	1	0	1	0	1	1	1	0.0007	0.9993
0	1	1	0	1	1	0	0	0	0.0140	0.9860
0	1	1	0	1	1	0	0	1	0.0041	0.9959
0	1	1	0	1	1	0	1	0	0.0030	0.9970
0	1	1	0	1	1	0	1	1	0.0009	0.9991
0	1	1	0	1	1	1	0	0	0.0035	0.9965
0	1	1	0	1	1	1	0	1	0.0010	0.9990
0	1	1	0	1	1	1	1	0	0.0007	0.9993
0	1	1	0	1	1	1	1	1	0.0002	0.9998
0	1	1	1	0	0	0	0	0	0.0392	0.9608
0	1	1	1	0	0	0	0	1	0.0114	0.9886
0	1	1	1	0	0	0	1	0	0.0084	0.9916
0	1	1	1	0	0	0	1	1	0.0024	0.9976
0	1	1	1	0	0	1	0	0	0.0097	0.9903
0	1	1	1	0	0	1	0	1	0.0028	0.9972
0	1	1	1	0	0	1	1	0	0.0021	0.9979
0	1	1	1	0	0	1	1	1	0.0006	0.9994
0	1	1	1	0	1	0	0	0	0.0124	0.9876
0	1	1	1	0	1	0	0	1	0.0036	0.9964
0	1	1	1	0	1	0	1	0	0.0027	0.9973
0	1	1	1	0	1	0	1	1	0.0008	0.9992
0	1	1	1	0	1	1	0	0	0.0031	0.9969
0	1	1	1	0	1	1	0	1	0.0009	0.9991
0	1	1	1	0	1	1	1	0	0.0007	0.9993
0	1	1	1	0	1	1	1	1	0.0002	0.9998
0	1	1	1	1	0	0	0	0	0.0113	0.9887
0	1	1	1	1	0	0	0	1	0.0033	0.9967

0	1	1	1	1	0	0	1	0	0.0024	0.9976
0	1	1	1	1	0	0	1	1	0.0007	0.9993
0	1	1	1	1	0	1	0	0	0.0028	0.9972
0	1	1	1	1	0	1	0	1	0.0008	0.9992
0	1	1	1	1	0	1	1	0	0.0006	0.9994
0	1	1	1	1	0	1	1	1	0.0002	0.9998
0	1	1	1	1	1	0	0	0	0.0036	0.9964
0	1	1	1	1	1	0	0	1	0.0010	0.9990
0	1	1	1	1	1	0	1	0	0.0008	0.9992
0	1	1	1	1	1	0	1	1	0.0002	0.9998
0	1	1	1	1	1	1	0	0	0.0009	0.9991
0	1	1	1	1	1	1	0	1	0.0003	0.9997
0	1	1	1	1	1	1	1	0	0.0002	0.9998
0	1	1	1	1	1	1	1	1	0.0001	0.9999
1	0	0	0	0	0	0	0	0	0.5643	0.4357
1	0	0	0	0	0	0	0	1	0.1635	0.8365
1	0	0	0	0	0	0	1	0	0.1206	0.8794
1	0	0	0	0	0	0	1	1	0.0349	0.9651
1	0	0	0	0	0	1	0	0	0.1400	0.8600
1	0	0	0	0	0	1	0	1	0.0406	0.9594
1	0	0	0	0	0	1	1	0	0.0299	0.9701
1	0	0	0	0	0	1	1	1	0.0087	0.9913
1	0	0	0	0	1	0	0	0	0.1787	0.8213
1	0	0	0	0	1	0	0	1	0.0518	0.9482
1	0	0	0	0	1	0	1	0	0.0382	0.9618
1	0	0	0	0	1	0	1	1	0.0111	0.9889
1	0	0	0	0	1	1	0	0	0.0443	0.9557
1	0	0	0	0	1	1	0	1	0.0128	0.9872
1	0	0	0	0	1	1	1	0	0.0095	0.9905
1	0	0	0	0	1	1	1	1	0.0027	0.9973
1	0	0	0	1	0	0	0	0	0.1622	0.8378
1	0	0	0	1	0	0	0	1	0.0470	0.9530
1	0	0	0	1	0	0	1	0	0.0347	0.9653
1	0	0	0	1	0	0	1	1	0.0100	0.9900
1	0	0	0	1	0	1	0	0	0.0402	0.9598
1	0	0	0	1	0	1	0	1	0.0117	0.9883
1	0	0	0	1	0	1	1	0	0.0086	0.9914
1	0	0	0	1	0	1	1	1	0.0025	0.9975
1	0	0	0	1	1	0	0	0	0.0514	0.9486
1	0	0	0	1	1	0	0	1	0.0149	0.9851
1	0	0	0	1	1	0	1	0	0.0110	0.9890
1	0	0	0	1	1	0	1	1	0.0032	0.9968

1	0	0	0	1	1	1	0	0	0.0127	0.9873
1	0	0	0	1	1	1	0	1	0.0037	0.9963
1	0	0	0	1	1	1	1	0	0.0027	0.9973
1	0	0	0	1	1	1	1	1	0.0008	0.9992
1	0	0	1	0	0	0	0	0	0.1438	0.8562
1	0	0	1	0	0	0	0	1	0.0416	0.9584
1	0	0	1	0	0	0	1	0	0.0307	0.9693
1	0	0	1	0	0	0	1	1	0.0089	0.9911
1	0	0	1	0	0	1	0	0	0.0357	0.9643
1	0	0	1	0	0	1	0	1	0.0103	0.9897
1	0	0	1	0	0	1	1	0	0.0076	0.9924
1	0	0	1	0	0	1	1	1	0.0022	0.9978
1	0	0	1	0	1	0	0	0	0.0455	0.9545
1	0	0	1	0	1	0	0	1	0.0132	0.9868
1	0	0	1	0	1	0	1	0	0.0097	0.9903
1	0	0	1	0	1	0	1	1	0.0028	0.9972
1	0	0	1	0	1	1	0	0	0.0113	0.9887
1	0	0	1	0	1	1	0	1	0.0033	0.9967
1	0	0	1	0	1	1	1	0	0.0024	0.9976
1	0	0	1	0	1	1	1	1	0.0007	0.9993
1	0	0	1	1	0	0	0	0	0.0413	0.9587
1	0	0	1	1	0	0	0	1	0.0120	0.9880
1	0	0	1	1	0	0	1	0	0.0088	0.9912
1	0	0	1	1	0	0	1	1	0.0026	0.9974
1	0	0	1	1	0	1	0	0	0.0103	0.9897
1	0	0	1	1	0	1	0	1	0.0030	0.9970
1	0	0	1	1	0	1	1	0	0.0022	0.9978
1	0	0	1	1	0	1	1	1	0.0006	0.9994
1	0	0	1	1	1	0	0	0	0.0131	0.9869
1	0	0	1	1	1	0	0	1	0.0038	0.9962
1	0	0	1	1	1	0	1	0	0.0028	0.9972
1	0	0	1	1	1	0	1	1	0.0008	0.9992
1	0	0	1	1	1	1	0	0	0.0032	0.9968
1	0	0	1	1	1	1	0	1	0.0009	0.9991
1	0	0	1	1	1	1	1	0	0.0007	0.9993
1	0	0	1	1	1	1	1	1	0.0002	0.9998
1	0	1	0	0	0	0	0	0	0.1952	0.8048
1	0	1	0	0	0	0	0	1	0.0565	0.9435
1	0	1	0	0	0	0	1	0	0.0417	0.9583
1	0	1	0	0	0	0	1	1	0.0121	0.9879
1	0	1	0	0	0	1	0	0	0.0484	0.9516
1	0	1	0	0	0	1	0	1	0.0140	0.9860

1	0	1	0	0	0	1	1	0	0.0103	0.9897
1	0	1	0	0	0	1	1	1	0.0030	0.9970
1	0	1	0	0	1	0	0	0	0.0618	0.9382
1	0	1	0	0	1	0	0	1	0.0179	0.9821
1	0	1	0	0	1	0	1	0	0.0132	0.9868
1	0	1	0	0	1	0	1	1	0.0038	0.9962
1	0	1	0	0	1	1	0	0	0.0153	0.9847
1	0	1	0	0	1	1	0	1	0.0044	0.9956
1	0	1	0	0	1	1	1	0	0.0033	0.9967
1	0	1	0	0	1	1	1	1	0.0009	0.9991
1	0	1	0	1	0	0	0	0	0.0561	0.9439
1	0	1	0	1	0	0	0	1	0.0163	0.9837
1	0	1	0	1	0	0	1	0	0.0120	0.9880
1	0	1	0	1	0	0	1	1	0.0035	0.9965
1	0	1	0	1	0	1	0	0	0.0139	0.9861
1	0	1	0	1	0	1	0	1	0.0040	0.9960
1	0	1	0	1	0	1	1	0	0.0030	0.9970
1	0	1	0	1	0	1	1	1	0.0009	0.9991
1	0	1	0	1	1	0	0	0	0.0178	0.9822
1	0	1	0	1	1	0	0	1	0.0051	0.9949
1	0	1	0	1	1	0	1	0	0.0038	0.9962
1	0	1	0	1	1	0	1	1	0.0011	0.9989
1	0	1	0	1	1	1	0	0	0.0044	0.9956
1	0	1	0	1	1	1	0	1	0.0013	0.9987
1	0	1	0	1	1	1	1	0	0.0009	0.9991
1	0	1	0	1	1	1	1	1	0.0003	0.9997
1	0	1	1	0	0	0	0	0	0.0497	0.9503
1	0	1	1	0	0	0	0	1	0.0144	0.9856
1	0	1	1	0	0	0	1	0	0.0106	0.9894
1	0	1	1	0	0	0	1	1	0.0031	0.9969
1	0	1	1	0	0	1	0	0	0.0123	0.9877
1	0	1	1	0	0	1	0	1	0.0036	0.9964
1	0	1	1	0	0	1	1	0	0.0026	0.9974
1	0	1	1	0	0	1	1	1	0.0008	0.9992
1	0	1	1	0	1	0	0	0	0.0158	0.9842
1	0	1	1	0	1	0	0	1	0.0046	0.9954
1	0	1	1	0	1	0	1	0	0.0034	0.9966
1	0	1	1	0	1	0	1	1	0.0010	0.9990
1	0	1	1	0	1	1	0	0	0.0039	0.9961
1	0	1	1	0	1	1	0	1	0.0011	0.9989
1	0	1	1	0	1	1	1	0	0.0008	0.9992
1	0	1	1	0	1	1	1	1	0.0002	0.9998

1	0	1	1	1	0	0	0	0	0.0143	0.9857
1	0	1	1	1	0	0	0	1	0.0041	0.9959
1	0	1	1	1	0	0	1	0	0.0031	0.9969
1	0	1	1	1	0	0	1	1	0.0009	0.9991
1	0	1	1	1	0	1	0	0	0.0035	0.9965
1	0	1	1	1	0	1	0	1	0.0010	0.9990
1	0	1	1	1	0	1	1	0	0.0008	0.9992
1	0	1	1	1	0	1	1	1	0.0002	0.9998
1	0	1	1	1	1	0	0	0	0.0045	0.9955
1	0	1	1	1	1	0	0	1	0.0013	0.9987
1	0	1	1	1	1	0	1	0	0.0010	0.9990
1	0	1	1	1	1	0	1	1	0.0003	0.9997
1	0	1	1	1	1	1	0	0	0.0011	0.9989
1	0	1	1	1	1	1	0	1	0.0003	0.9997
1	0	1	1	1	1	1	1	0	0.0002	0.9998
1	0	1	1	1	1	1	1	1	0.0001	0.9999
1	1	0	0	0	0	0	0	0	0.2511	0.7489
1	1	0	0	0	0	0	0	1	0.0727	0.9273
1	1	0	0	0	0	0	1	0	0.0537	0.9463
1	1	0	0	0	0	0	1	1	0.0155	0.9845
1	1	0	0	0	0	1	0	0	0.0623	0.9377
1	1	0	0	0	0	1	0	1	0.0180	0.9820
1	1	0	0	0	0	1	1	0	0.0133	0.9867
1	1	0	0	0	0	1	1	1	0.0039	0.9961
1	1	0	0	0	1	0	0	0	0.0795	0.9205
1	1	0	0	0	1	0	0	1	0.0230	0.9770
1	1	0	0	0	1	0	1	0	0.0170	0.9830
1	1	0	0	0	1	0	1	1	0.0049	0.9951
1	1	0	0	0	1	1	0	0	0.0197	0.9803
1	1	0	0	0	1	1	0	1	0.0057	0.9943
1	1	0	0	0	1	1	1	0	0.0042	0.9958
1	1	0	0	0	1	1	1	1	0.0012	0.9988
1	1	0	0	1	0	0	0	0	0.0722	0.9278
1	1	0	0	1	0	0	0	1	0.0209	0.9791
1	1	0	0	1	0	0	1	0	0.0154	0.9846
1	1	0	0	1	0	0	1	1	0.0045	0.9955
1	1	0	0	1	0	1	0	0	0.0179	0.9821
1	1	0	0	1	0	1	0	1	0.0052	0.9948
1	1	0	0	1	0	1	1	0	0.0038	0.9962
1	1	0	0	1	0	1	1	1	0.0011	0.9989
1	1	0	0	1	1	0	0	0	0.0229	0.9771
1	1	0	0	1	1	0	0	1	0.0066	0.9934

1	1	0	0	1	1	0	1	0	0.0049	0.9951
1	1	0	0	1	1	0	1	1	0.0014	0.9986
1	1	0	0	1	1	1	0	0	0.0057	0.9943
1	1	0	0	1	1	1	0	1	0.0016	0.9984
1	1	0	0	1	1	1	1	0	0.0012	0.9988
1	1	0	0	1	1	1	1	1	0.0004	0.9996
1	1	0	1	0	0	0	0	0	0.0640	0.9360
1	1	0	1	0	0	0	0	1	0.0185	0.9815
1	1	0	1	0	0	0	1	0	0.0137	0.9863
1	1	0	1	0	0	0	1	1	0.0040	0.9960
1	1	0	1	0	0	1	0	0	0.0159	0.9841
1	1	0	1	0	0	1	0	1	0.0046	0.9954
1	1	0	1	0	0	1	1	0	0.0034	0.9966
1	1	0	1	0	0	1	1	1	0.0010	0.9990
1	1	0	1	0	1	0	0	0	0.0203	0.9797
1	1	0	1	0	1	0	0	1	0.0059	0.9941
1	1	0	1	0	1	0	1	0	0.0043	0.9957
1	1	0	1	0	1	0	1	1	0.0013	0.9987
1	1	0	1	0	1	1	0	0	0.0050	0.9950
1	1	0	1	0	1	1	0	1	0.0015	0.9985
1	1	0	1	0	1	1	1	0	0.0011	0.9989
1	1	0	1	0	1	1	1	1	0.0003	0.9997
1	1	0	1	1	0	0	0	0	0.0184	0.9816
1	1	0	1	1	0	0	0	1	0.0053	0.9947
1	1	0	1	1	0	0	1	0	0.0039	0.9961
1	1	0	1	1	0	0	1	1	0.0011	0.9989
1	1	0	1	1	0	1	0	0	0.0046	0.9954
1	1	0	1	1	0	1	0	1	0.0013	0.9987
1	1	0	1	1	0	1	1	0	0.0010	0.9990
1	1	0	1	1	0	1	1	1	0.0003	0.9997
1	1	0	1	1	1	0	0	0	0.0058	0.9942
1	1	0	1	1	1	0	0	1	0.0017	0.9983
1	1	0	1	1	1	0	1	0	0.0012	0.9988
1	1	0	1	1	1	0	1	1	0.0004	0.9996
1	1	0	1	1	1	1	0	0	0.0014	0.9986
1	1	0	1	1	1	1	0	1	0.0004	0.9996
1	1	0	1	1	1	1	1	0	0.0003	0.9997
1	1	0	1	1	1	1	1	1	0.0001	0.9999
1	1	1	0	0	0	0	0	0	0.0869	0.9131
1	1	1	0	0	0	0	0	1	0.0252	0.9748
1	1	1	0	0	0	0	1	0	0.0186	0.9814
1	1	1	0	0	0	0	1	1	0.0054	0.9946

1	1	1	0	0	0	1	0	0	0.0215	0.9785
1	1	1	0	0	0	1	0	1	0.0062	0.9938
1	1	1	0	0	0	1	1	0	0.0046	0.9954
1	1	1	0	0	0	1	1	1	0.0013	0.9987
1	1	1	0	0	1	0	0	0	0.0275	0.9725
1	1	1	0	0	1	0	0	1	0.0080	0.9920
1	1	1	0	0	1	0	1	0	0.0059	0.9941
1	1	1	0	0	1	0	1	1	0.0017	0.9983
1	1	1	0	0	1	1	0	0	0.0068	0.9932
1	1	1	0	0	1	1	0	1	0.0020	0.9980
1	1	1	0	0	1	1	1	0	0.0015	0.9985
1	1	1	0	0	1	1	1	1	0.0004	0.9996
1	1	1	0	1	0	0	0	0	0.0250	0.9750
1	1	1	0	1	0	0	0	1	0.0072	0.9928
1	1	1	0	1	0	0	1	0	0.0053	0.9947
1	1	1	0	1	0	0	1	1	0.0015	0.9985
1	1	1	0	1	0	1	0	0	0.0062	0.9938
1	1	1	0	1	0	1	0	1	0.0018	0.9982
1	1	1	0	1	0	1	1	0	0.0013	0.9987
1	1	1	0	1	0	1	1	1	0.0004	0.9996
1	1	1	0	1	1	0	0	0	0.0079	0.9921
1	1	1	0	1	1	0	0	1	0.0023	0.9977
1	1	1	0	1	1	0	1	0	0.0017	0.9983
1	1	1	0	1	1	0	1	1	0.0005	0.9995
1	1	1	0	1	1	1	0	0	0.0020	0.9980
1	1	1	0	1	1	1	0	1	0.0006	0.9994
1	1	1	0	1	1	1	1	0	0.0004	0.9996
1	1	1	0	1	1	1	1	1	0.0001	0.9999
1	1	1	1	0	0	0	0	0	0.0221	0.9779
1	1	1	1	0	0	0	0	1	0.0064	0.9936
1	1	1	1	0	0	0	1	0	0.0047	0.9953
1	1	1	1	0	0	0	1	1	0.0014	0.9986
1	1	1	1	0	0	1	0	0	0.0055	0.9945
1	1	1	1	0	0	1	0	1	0.0016	0.9984
1	1	1	1	0	0	1	1	0	0.0012	0.9988
1	1	1	1	0	0	1	1	1	0.0003	0.9997
1	1	1	1	0	1	0	0	0	0.0070	0.9930
1	1	1	1	0	1	0	0	1	0.0020	0.9980
1	1	1	1	0	1	0	1	0	0.0015	0.9985
1	1	1	1	0	1	0	1	1	0.0004	0.9996
1	1	1	1	0	1	1	0	0	0.0017	0.9983
1	1	1	1	0	1	1	0	1	0.0005	0.9995

1	1	1	1	0	1	1	1	0	0.0004	0.9996
1	1	1	1	0	1	1	1	1	0.0001	0.9999
1	1	1	1	1	0	0	0	0	0.0064	0.9936
1	1	1	1	1	0	0	0	1	0.0018	0.9982
1	1	1	1	1	0	0	1	0	0.0014	0.9986
1	1	1	1	1	0	0	1	1	0.0004	0.9996
1	1	1	1	1	0	1	0	0	0.0016	0.9984
1	1	1	1	1	0	1	0	1	0.0005	0.9995
1	1	1	1	1	0	1	1	0	0.0003	0.9997
1	1	1	1	1	0	1	1	1	0.0001	0.9999
1	1	1	1	1	1	0	0	0	0.0020	0.9980
1	1	1	1	1	1	0	0	1	0.0006	0.9994
1	1	1	1	1	1	0	1	0	0.0004	0.9996
1	1	1	1	1	1	0	1	1	0.0001	0.9999
1	1	1	1	1	1	1	0	0	0.0005	0.9995
1	1	1	1	1	1	1	0	1	0.0001	0.9999
1	1	1	1	1	1	1	1	0	0.0001	0.9999
1	1	1	1	1	1	1	1	1	0.0000	1.0000