Human-Interaction-aware Adaptive Functional Safety Processing for Multi-Functional Automotive Cyber-Physical Systems

GUOQI XIE, YANG BAI, and WEI WU, Hunan University
YANWEN LI, China Automotive Technology and Research Center
RENFA LI, Hunan University
KEQIN LI, State University of New York

The functional safety research for automotive cyber-physical systems (ACPS) has been studied in recent years; however, these studies merely consider the change in the exposure of the functional safety classification and assume that the driver's controllability in the functional safety classification is always fixed and uncontrollable. In fact, the driver's controllability is variable during the runtime phase, such that the execution process of safety-critical automotive functions is a human-interaction-aware process between the driver and ACPS. To adapt to the changes in the driver's controllability, this article studies the human-interaction-aware adaptive functional safety processing for multi-functional ACPS in two main phases. In the design phase, where the driver's controllability is fixed at the highest level (i.e., C3), we obtain the approximate optimal priority sequence of safety-critical functions without exhausting all sequences by proposing the refined exploration method. In the runtime phase, where the driver's controllability level is variable (i.e., C0, C1, C2, or C3), we propose the human-interaction-aware task remapping method to autonomously respond to the change of the driver's controllability. Examples and experiments confirm that the proposed adaptive functional safety processing can reduce overall task redundancy of safety-critical automotive functions while meeting their functional safety requirements, shorten the overall response time of safety-critical automotive functions, and increase the slack time for non-safety-critical automotive functions.

CCS Concepts: • **Computer systems organization → Embedded and cyber-physical systems**;

Additional Key Words and Phrases: Automotive cyber-physical systems (ACPS), functional safety, human-interaction-aware

This work was supported in part by the National Natural Science Foundation of China under Grant 61702172, Grant 61672217, the National Technical Committee of Auto Standardization Research Foundation of China under Grant BZ201908, the Natural Science Foundation of Hunan Province under Grant 2018JJ3076, the Open Research Project of the State Key Laboratory of Synthetical Automation for Process Industries (SAPI), Northeastern University, China under Grant PAL-N201803, and the Fundamental Research Funds for the Central Universities, Hunan University, China.

Authors' addresses: G. Xie, Y. Bai, W. Wu, and R. Li, Key Laboratory for Embedded and Cyber-Physical Systems of Hunan Province, College of Computer Science and Electronic Engineering, Hunan University, China; emails: {xgqman, baiyang, wu96, lirenfa}@hnu.edu.cn; Y. Li, Automotive Engineering Research Institute, China Automotive Technology and Research Center, China; email: vliyanwen@catarc.ac.cn; K. Li, Department of Computer Science, State University of New York, New Paltz, NY; email: blik@newpaltz.edu.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2019 Association for Computing Machinery.

2378-962X/2019/08-ART39 $15.00
https://doi.org/10.1145/3337931

ACM Reference format:
Guoqi Xie, Yang Bai, Wei Wu, Yanwen Li, Renfa Li, and Keqin Li. 2019. Human-Interaction-aware Adaptive Functional Safety Processing for Multi-Functional Automotive Cyber-Physical Systems. *ACM Trans. Cyber-Phys. Syst.* 3, 4, Article 39 (August 2019), 25 pages.
https://doi.org/10.1145/3337931

1 INTRODUCTION

1.1 Background

An electronic control unit (ECU) can support multiple automotive functions, and one automotive function is distributed to multiple ECUs for execution in the integrated automotive electrical and electronic (E/E) architecture [19, 21, 34]. In this architecture, collecting physical data from 360° sensors and performing control command to actuators is an end-to-end computation and communication process for some safety-critical automotive functions [19, 21, 34]. Examples of such safety-critical automotive functions include active cruise control, lane departure warning, and collision avoidance [19, 20, 34]. Considering the heterogeneity, interaction, and diverse nature of such an architecture [11, 15], the joint and tight interaction between the cyber part and physical part are required, such that multi-functional automotive cyber-physical-systems (ACPS) have been studied [34].

Automotive functions are classified into three types in multi-functional ACPS: active safety, passive safety, and non-safety automotive functions. The first edition of the automotive functional safety standard ISO 26262 was released in November 2011 [2]. Currently, the second edition of ISO 26262 was released December 2018 [1]. As stated in ISO 26262, safety refers to the absence of unreasonable risk, and risk refers to the combination of the probability of harm and the severity of that harm [1, 2]. To visually reflect the risk degree of an automotive function, the automotive safety integrity level (ASIL) proposed by the ISO 26262 has been adopted for a risk classification scheme.

ISO 26262 provides four ASILs (i.e., ASIL A, ASIL B, ASIL C, and ASIL D), where ASIL A and ASIL D are the lowest and highest ASILs, respectively. In general, a high-ASIL automotive function is more prone to be harmed and the severity of the harm is more threatening than a low-ASIL automotive function. ISO 26262 defines three mutually orthotropic functional safety attributes (i.e., severity, exposure, and controllability) to construct the four ASILs [2, 18]. Severity means the extent of harm to an individual in a specific situation, exposure means the relative expected frequency of the operational conditions, and controllability means the avoidance of the specified harm or damage through the timely reactions of the driver involved. Different from severity and exposure, which are related to the system (i.e., ACPS), controllability is related to the driver's driving state. Another interpretation is that controllability is a characteristic of the function, when statistically considering a typical driver (i.e., controllability does not vary with the driver status). However, the definitions in ISO 26262 are rather vague and there may be different interpretations for them especially in the industry [12]. Overall, an ASIL is the combination of severity, exposure, and controllability [2, 18] (refer to Section 4.1 for more details about ASIL determination based on severity, exposure, and controllability).

1.2 Motivation

To ensure that the risk of a high-ASIL automotive function is within the acceptable range, a feasible approach is to reduce its ASIL by reducing its exposure, thereby enhancing its reliability. For instance, some works have been studied for the functional safety enhancement [31] or functional safety assurance [30] by changing the exposure of safety-critical automotive functions. However, these works merely consider the change in exposure and assume that the driver's controllability is always fixed and uncontrollable. This is reasonable during the design phase, because the concrete controllability value is known only during the runtime phase. However, the driver's controllability is related to his/her current physical and mental state, such that it is variable during the runtime phase. Hence, setting the controllability to be fixed and uncontrollable is quite conservative during the runtime phase, and such conservativeness will result in a large waste of resources. For instance, when the driver's controllability is strong, the required system resources can be reduced accordingly while the required ASIL can still be maintained. Therefore, for the cost-sensitive automotive industry, the driver's controllability needs to be given enough attention during the runtime phase.

Considering that the driver's controllability has an impact on the system safety, this forms a human-interaction process between the driver and ACPS. In addition, multi-functional ACPS also contain some non-safety-critical automotive functions, which also require the corresponding system resources to support their execution. To achieve human-interaction-aware functional safety assurance, as well as optimizing the utilization of system resources during the runtime phase, it is quite necessary for the human-interaction-aware adaptive functional safety processing for multi-functional ACPS. Adaptive processing for ACPS should autonomously respond to environmental or internal changes at runtime [11, 34].

1.3 Related Work

Human-interaction is one of the important aspects for CPS. In Reference [5], the authors described the interactions between humans and CPS developed by the national institute of standards and technology (NIST). In Reference [23], the authors elaborated potential solutions to human factors challenges in driving automation of automobiles. Human factors in automotive engineering and technology were widely discussed in Reference [26], which seeks to bridge the gap among automobiles, engineers, and human factors. In Reference [16], the authors provided a review and outlook of human-interaction for automobiles. Reference [16] pointed out that it is a critical issue to create safe automotive interaction that assists the driver to complete the driving task and various non-driving tasks. In Reference [22], the authors reviewed current human vehicle interaction design challenges and pointed out that the interaction has become a primary consideration in meeting automotive user experience. In Reference [13], the authors proposed the driver, environment, software, hardware, and goal (DESH-G) model as a framework to calculate the task demand from the situation-scenario matrix. In Reference [14], the authors showed a controllability classification evaluation method without deviation and presented examples of the evaluation of three hazardous events in actual vehicle tests.

Although safety issues are mentioned in the above literatures, no in-depth research has been conducted. The entire development lifecycle of ACPS includes analysis, design, implementation, testing, runtime, and maintenance phases, and all of these phases involve functional safety management issues. The design and the runtime phases are the key phases of functional safety management research. The functional safety guarantee during the design phase generally includes functional safety verification [35], functional safety enhancement [31], and functional safety assurance [30] by task scheduling. A union fast functional safety requirement verification (UFFSV) method for a safety-critical automotive function was proposed in Reference [35] during the early design phase. If UFFSV is passed, then the design burden can be reduced for designers based on the verification results; otherwise, abandon the late design phase or adopt a functional safety enhancement method to make the functional safety verification be passed as much as possible, as proposed in Reference [31]. Related functional safety assurance methods were proposed in Reference [30] based on geometric mean by assuring the reliability requirement.

Besides the functional safety management during the design phase, an adaptive dynamic scheduling method for mixed-criticality multi-functional ACPS during the runtime phase was proposed in Reference [34], which refers to the relevant requirements of the newly released AUTOSAR adaptive platform standard [8, 9]. However, all the aforementioned works merely consider the change in the exposure of the functional safety classification and assume that the driver's controllability of the functional safety classification is always fixed and uncontrollable. In fact, the driver's controllability is variable during the runtime phase, such that the execution process of safety-critical automotive functions is a human-interaction-aware process between the driver and ACPS as pointed earlier.

1.4 Contributions

To adapt to the change in the driver's controllability, this article proposes a human-interaction-aware adaptive functional safety processing methodology for multi-functional ACPS based on the ISO 26262 standard, which is specifically developed for non-autonomous vehicles. Since the functional safety standard for autonomous vehicles is still in the process of development, the research in this article is only applicable to non-autonomous vehicles. The methodology includes the proposed methods during the design and runtime phases. The contributions of this article are outlined below.

(1) In the design phase, where the driver's controllability is fixed at the highest level (i.e., C3), we obtain the approximate optimal priority sequence of safety-critical functions without exhausting all sequences by proposing the refined exploration method. A heuristic method proposed in this phase is a novel contribution, which determines the mapping order of functions.

(2) In the runtime phase, where the driver's controllability level is variable (i.e., C0, C1, C2, or C3), we propose the human-interaction-aware task remapping method to autonomously respond to the change of the driver's controllability.

2 MODELS

2.1 Multi-Functional ACPS

Current ACPS E/E architecture is an integrated architecture, where multiple subsystems (e.g., the engine control subsystem, the powertrain subsystem, the body subsystem, and the entertainment subsystem, etc.) are intergraded in the same system by a central gateway [34]. In the above subsystems, the engine control subsystem and the powertrain subsystem are safety-critical subsystems, whereas the body subsystem and the entertainment subsystem are non-safety-critical subsystems. Considering that this article focuses on functional safety issue in ACPS, we only consider a safety-critical subsystem (e.g., the engine control subsystem or the powertrain subsystem). In addition, this article does not consider connected vehicles; therefore, ACPS can ignore cyber security problems caused by hackers' attacks in this situation.

A safety-critical subsystem contains not only multiple safety-critical automotive functions but also multiple non-safety-critical automotive functions. This article assumes that all safety-critical automotive functions are only executed inside the subsystem in one CAN bus and do not cross two or more subsystems, namely, all safety-critical automotive functions are not cross-domain automotive functions, as shown in Figure 1.

An automotive function must perform its execution by interacting with the physical world through sensors and actuators. Therefore, these sensors and actuators are connected to the corresponding ECUs, as shown in Figure 1. Let $U = \{u_1, u_2, \ldots, u_k, \ldots, u_{|U|}\}$ be the ECU set in ACPS and these ECUs are heterogeneous (see Figure 1), where $|U|$ is the size of U. Notice that the sensors and actuators are redundant, because physical processes are compositions of many parallel processes [34]. An automotive function typically involves end-to-end computing and communication; therefore, such function is also known as an end-to-end automotive function [34]. That is, an end-to-end automotive function is released to collect the sensor data, then goes through the intermediate computation and communication process, and, finally, is ended by sending the executive command to the actuators.

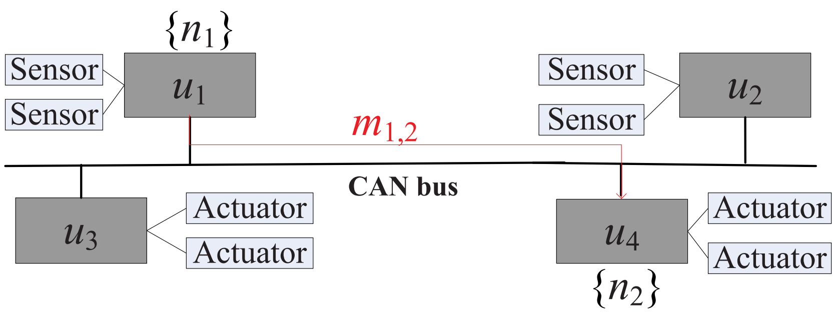

Fig. 1. ACPS E/E architecture in this article.

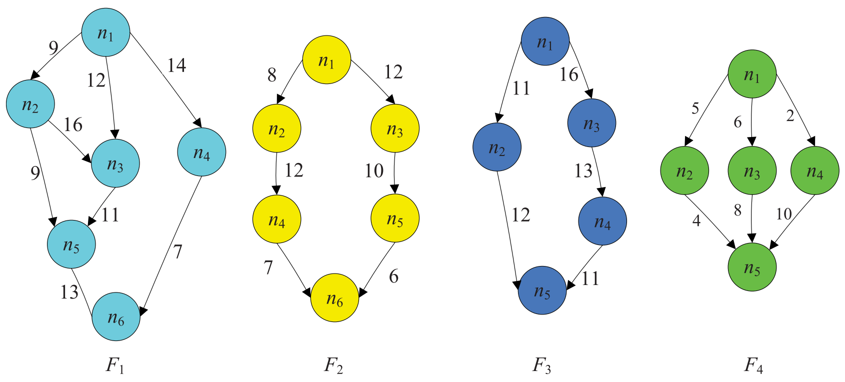

Fig. 2. Example of four automotive functions in ACPS.

Let $S = \{F_1, F_2, \ldots, F_m, \ldots, F_{|S|}\}$ be the safety-critical automotive function set in ACPS, where F_m represents the m*th* automotive function. Figure 2 shows four safety-critical automotive functions (i.e., F_1, F_2, F_3, and F_4) in ACPS. The functional safety attribute contains not only reliability and real-time but also robustness and stability, and to on. In this article, we mainly focus on reliability and real-time and assume that other functional safety attributes such as robustness and stability are all meeting the corresponding requirements. Therefore, to meet functional safety requirements, both reliability and real-time requirements must be simultaneously met from a perspective of functional safety assurance in this article. We assume that these safety-critical automotive functions are released simultaneously. In other words, this article considers multi-functional static scheduling rather than multi-functional dynamic scheduling. Such cases can be found in ACPS. For instance, integrated safety systems include the automotive functions of anti-lock braking system (ABS), acceleration slip regulation (ASR), and electronic stability program (ESP). To avoid possible collisions in an emergent state, these safety-critical automotive functions will be released simultaneously.

2.2 Automotive Function Model

A safety-critical automotive function F_m is represented by a DAG $F_m = (N, W, E, C)$. $F_m.N$ represents the task set of F_m, $F_m.W$ represents the worst-case execution time (WCET) matrix of tasks in F_m, $F_m.E$ represents the message set of F_m, and $F_m.C$ represents the worst-case response time (WCRT) set of messages in F_m.

(1) Let $F_m.n_i$ be a task (i.e., a node in DAG) in $F_m.N$. Considering that there are precedence constrains among tasks, we let $pred(F_m.n_i)$ and $succ(F_m.n_i)$ be the immediate predecessor task set and immediate successor task set, respectively, of $F_m.n_i$. For example, we have $pred(F_1.n_3) = \{F_1.n_1, F_1.n_2\}$ and $succ(F_1.n_3) = \{F_1.n_5\}$ in Figure 2, where $F_1.n_1$ is the entry task and is called

Table 1. WCET Matrixes of Four Automotive Functions in Figure 2

(a) WCET matrix of F_1

Tasks	$F_1.n_1$	$F_1.n_2$	$F_1.n_3$	$F_1.n_4$	$F_1.n_5$	$F_1.n_6$
u_1	8	14	9	8	18	5
u_2	11	13	12	15	16	10
u_3	9	8	16	14	9	7

(b) WCET matrix of F_2

Tasks	$F_2.n_1$	$F_2.n_2$	$F_2.n_3$	$F_2.n_4$	$F_2.n_5$	$F_2.n_6$
u_1	12	9	7	13	18	15
u_2	8	15	12	15	10	10
u_3	9	11	16	18	20	8

(c) WCET matrix of F_3

Tasks	$F_3.n_1$	$F_3.n_2$	$F_3.n_3$	$F_3.n_4$	$F_3.n_5$
u_1	15	13	11	14	8
u_2	9	16	10	6	10
u_3	12	9	6	11	17

(d) WCET matrix of F_4

Tasks	$F_4.n_1$	$F_4.n_2$	$F_4.n_3$	$F_3.n_4$	$F_4.n_5$
u_1	4	9	10	11	7
u_2	10	10	7	15	10
u_3	6	7	6	9	5

$F_1.n_{\text{entry}}$, whereas $F_1.n_6$ is the exit task and is called $F_1.n_{\text{exit}}$. Notice that a dummy entry or exit task with zero-weight dependencies is added to the DAG, because there may be multiple entry or exit tasks. For example, an automotive function (e.g., brake-by-wire) may be released in multiple ECUs by receiving collected data from multiple sensors and is completed in multiple ECUs by sending the performing action to multiple actuators.

(2) Let $F_m.w_{i,k}$ be the WCET of $F_m.n_i$ in u_k. Notice that $F_m.n_i$ has different WCET values in different ECUs, because the ECUs are heterogeneous. In this article, we assume that all the WCET values have been obtained by the WCET analysis method [4]. Table 1 shows the WCET matrixes of four automotive functions in Figure 2. For example, $F_1.w_{2,3} = 8$ represents the WCET of $F_1.n_2$ in u_3, as shown in Table 1(a).

(3) Let $F_m.e_{i,j}$ be a communication message (i.e., an edge in DAG) from $F_m.n_i$ to $F_m.n_j$ in $F_m.E$. As shown in Figure 2, there are communication messages of $F_1.e_{1,2}$, $F_1.e_{1,3}$, $F_1.e_{1,4}$, $F_1.e_{2,3}$, $F_1.e_{2,5}$, $F_1.e_{3,5}$, $F_1.e_{4,6}$, and $F_1.e_{5,6}$.

(4) Let $F_m.c_{i,j} \in C$ be the WCRT of $F_m.e_{i,j}$. In this article, we assume that all the WCRT values have been obtained by the WCRT analysis method [30]. As shown in Figure 2, the WCRT values of messages $F_1.e_{1,2}$, $F_1.e_{1,3}$, $F_1.e_{1,4}$, $F_1.e_{2,3}$, $F_1.e_{2,5}$, $F_1.e_{3,5}$, $F_1.e_{4,6}$, and $F_1.e_{5,6}$ are 9, 12, 14, 9, 16, 11, 7, and 13, respectively. Notice that if $F_m.n_i$ and $F_m.n_j$ are allocated to the same ECU, then the communication time from $F_m.n_i$ to $F_m.n_j$ is negligible, because the shared memory scheme is employed.

(5) Both preemptive and non-preemptive scheduling are supported in ACPS [6, 7]. Considering message scheduling in CAN bus is non-preemptive, we assume that the task scheduling in ECUs is also non-preemptive [30, 34].

Table 2. Classes of Probability of Exposure Regarding Duration/Probability of Exposure in ISO 26262 [2]

Exposure		Probability of exposure	Reliability requirement	Specified Reliability requirement in this article
E1	Very low probability	Not specified	At least exceeds 0.99	0.9999
E2	Low probability	<1%	0.99	0.99
E3	Medium probability	[1%, 10%]	>0.9	0.95
E4	High probability	>10%	<=0.9	0.9

2.3 Reliability Model

The reliability issue is actually the core foundation of the automotive functional safety. The duration/probability of exposure was provided in Table B.2, Annex B of Part 3 of the first edition of ISO 26262, as shown in Table 2 [1, 2]. Exposure is explained as the relative expected frequency of the operational conditions, in which hazardous events may occur and cause hazards or even injuries according to ISO 26262. Exposure is one of the functional safety attributes to construct the ASIL. Exposure is related to random hardware failures and can be understood by the inverse expression of reliability (i.e., exposure = 1 - reliability) [30, 31, 35]. There are two types of random hardware faults, namely, transient faults and permanent faults. Transient faults, such as soft errors (like bit flips), refer to failures that occur and subsequently disappear [17, 28, 36, 37]. Permanent faults, such as a broken connection or a short connection, refer to faults that occur and remain all the time. This article considers the transient faults of ECUs. Random hardware failures occur unpredictably during the lifecycle of a hardware, but random hardware failure rates can be predicated based on the probability distribution of random hardware faults as specified in ISO 26262 [2]. Besides, this article assumes that transient failures of ECUs follow the Poisson distribution, which has been adopted by a lot of research works [30, 31, 35].

Let λ_k be the failure rate of ECU u_k. The reliability value of task $F_m.n_i$ executed in u_k is

$$R\left(F_m.n_i, u_k\right) = e^{-\lambda_k \times F_m.w_{i,k}}. \tag{1}$$

Then, the exposure of n_i is

$$exposure\left(F_m.n_i, u_k\right) = 1 - R\left(F_m.n_i, u_k\right) = 1 - e^{-\lambda_k \times F_m.w_{i,k}}. \tag{2}$$

We can obtain reliability requirements through the given probabilities of exposures. Table 2 lists the reliability requirements under different exposures, where E2 has a fixed reliability requirement value of 0.99, whereas the reliability requirements of other exposures (i.e., E1, E3, and E4) are only given the reliability ranges, as shown in Table 2. In other words, the reliability requirements of E1, E3, and E4 are flexible. The advantage of such flexibility is that the automakers can specify specific reliability values according to their own needs, as long as the reliability values falls within the scopes required in ISO 26262. Notice that the classes of the probability of exposure in ISO 26262 shown in Table 2 are informative, but not prescriptive, and leave a great deal of discretion to designers [12]. In this article, we assume that the reliability requirement values are 0.9999, 0.95, and 0.9 for exposures E1, E3, and E4, respectively.

Passive replication (i.e., backup/restart) and active replication are two primary-backup replication paradigms. The detailed explanations for these paradigms can be found in Reference [30, 32]. Considering that when the reliability requirement reaches as much as 0.9999, passive replication is very difficult or almost impossible to meet such high reliability requirement. The second edition of ISO 26262 formally introduces the concept of fault-tolerance, which means that the capability to deliver a specified application for at least a limited time after a fault (i.e., the specified application

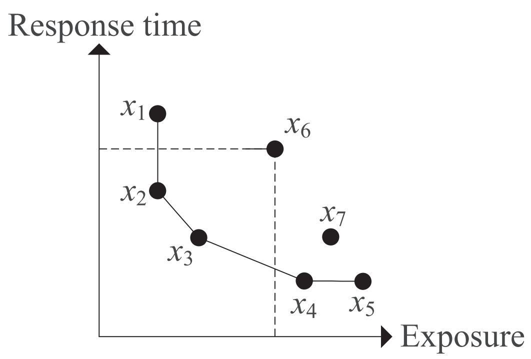

Fig. 3. A bi-criteria between response time minimization and exposure minimization [35].

still works after one or more faults have occurred in the system) [1]. Thus, similarly to most works [30, 32], this article adopts active replication to implement fault-tolerance.

Let num_i ($num_i \leqslant |U|$) be the number of replicas of $F_m.n_i$, and the replica set of $F_m.n_i$ is $\{F_m.n_i^1, F_m.n_i^2, \ldots, F_m.n_i^{num_i}\}$, where $F_m.n_i^1$ is the primary and the others are the backups. By using active replication, the reliability value of $F_m.n_i$ with num_i replicas is

$$R\left(F_m.n_i\right) = 1 - \prod_{\beta=1}^{num_i} \left(1 - R\left(F_m.n_i^{\beta}, u_{ecu(F_m.n_i^{\beta})}\right)\right), \tag{3}$$

where $u_{ecu(F_m.n_i^{\beta})}$ represents the allocated ECU of the βth replica $F_m.n_i^{\beta}$. The reliability value of automotive function F_m is the product of those of all tasks [30, 32], namely,

$$R(F_m) = \prod_{F_m.n_i \in F_m.|N|} R(F_m.n_i). \tag{4}$$

3 REFINED EXPLORATION DURING DESIGN PHASE

3.1 Reliability Requirement Assurance for a Function

Both reliability requirement and real-time requirement must be simultaneously assured toward functional safety assurance for a safety-critical automotive function [35]. Figure 3 shows the bi-criteria between response time minimization and exposure minimization (i.e., reliability maximization) for an end-to-end automotive function [35]. In other words, gaining higher reliability means longer response time in general.

Considering that scheduling tasks under quality of service (QoS) requirement for optimality in multiprocessors (ECUs) is known to be NP-hard [27, 38], the shortest response time under reliability requirement is hard to know in pseudo-polynomial time [30, 35]. Therefore, lower bound on response time (i.e., approximate shortest response time) will be adopted in this article. Considering there are four exposure levels in ISO 26262, the lower bound on response time of automotive function F_m also contains four results, namely, $LB(F_m, E1)$, $LB(F_m, E2)$, $LB(F_m, E3)$, and $LB(F_m, E4)$, where $LB(F_m, E1)$ represents the lower bound on response time of automotive function F_m in exposure E1. Considering that gaining higher reliability means longer response time according to the trend in Figure 3, $LB(F_m, E1)$ is the longest lower bound on response time, because E1 has the maximum reliability requirement of 0.9999, as shown in Table 2. For all safety-critical automotive functions, we set their exposure to E1 (i.e., the reliability requirement is 0.9999) during the design phase from a conservative and cautious perspective in this article.

Notice that $LB(F_m, E1)$ must be shorter than or equal to the real-time requirement $D(F_m)$ (i.e., deadline); otherwise, the automotive function cannot be completed correctly within the correct time, resulting in a systemic failure of the automotive function. Therefore, the first step of this article is to obtain $LB(F_m, E1)$, namely, the lower bound on response time of each automotive function under the reliability requirement of 0.9999.

The verifying functional safety requirement (VFSR) method was proposed in Reference [32] to obtain the lower bound on response time under a given reliability requirement. The idea of VFSR is that it transfers the reliability requirement of the automotive function to the reliability requirement of each task and then downward iteratively allocates tasks (i.e., from the entry task to the exit task) to ECUs. In this article, we propose a new method to implement the aforementioned objective by upward iteratively allocating tasks (i.e., from the exit task to the entry task) to ECUs. The VFSR method is called downward fault-tolerance toward reliability assurance (DFRA), and the proposed method in this section is called upward fault-tolerance toward reliability assurance (UFRA). UFRA aims to optimize the response time of the automotive function while meeting the reliability requirement. UFRA is suitable for functional safety assurance in multi-functional ACPS. According to our practical experience, UFRA can provide enough slack time at the early time interval compared to DFRA. Let us explain the UFRA method below.

(1) UFRA lets $F_m.n_{s(y)}$ be the yth downward task. $n_{s(y)}$ is considered the current task to be allocated. Then, two groups are separated by $F_m.n_{s(y)}$. The first group is $\{F_m.n_{s(1)}, F_m.n_{s(2)}, \ldots, F_m.n_{s(y-1)}\}$, where the tasks have not been allocated to ECUs, whereas the second group is $\{F_m.n_{s(y+1)}, F_m.n_{s(y+2)}, \ldots, F_m.n_{s(|N|)}\}$, where the tasks have been allocated to ECUs. Notice that all tasks of the automotive function are un-allocated in the initial state. In other words, y is changed from $F_m.|N|$ to 1 with 1 decrement until all tasks are allocated. Notice that we use the words "$F_m.n_{s(y)}$" and "$F_m.n_i$" interchangeably.

(2) The task priority (i.e., task allocation order) is based on the ascending order of the upward rank value (i.e., upward iteration) [25, 39]. The upward rank value is calculated by

$$rank_{\mathrm{u}}(F_m.n_i) = F_m.\overline{w_i} + \max_{F_m.n_j \in succ(F_m.n_i)} \{F_m.c_{i,j} + rank_{\mathrm{u}}(F_m.n_j)\}. \tag{5}$$

$F_m.\overline{w_i}$ represents the average WCET of task $F_m.n_i$ and is calculated by

$$F_m.\overline{w_i} = \left(\sum_{k=1}^{|U|} F_m.w_{i,k}\right) / |U|.$$

Notice that the $rank_{\mathrm{u}}(F_m.n_i)$ calculation is from the exit to entry tasks. $rank_{\mathrm{u}}(F_m.n_i)$ is a well-studied task allocation order strategy and is widely employed in various reliability requirement assurance approaches [30, 32, 35].

(3) Considering that the tasks in $\{F_m.n_{s(y+1)}, F_m.n_{s(y+2)}, \ldots, F_m.n_{s(|N|)}\}$ have been allocated to specific ECUs, the actual reliability values of these tasks are known when allocating $F_m.n_{s(y)}$. However, the tasks in $\{F_m.n_{s(1)}, F_m.n_{s(2)}, \ldots, F_m.n_{s(y-1)}\}$ have not been allocated to specific ECUs. For these un-allocated tasks, VFSR gives the upper bound on reliability [32]:

$$R_{\mathrm{up}}(F_m.n_{s(x)}) = \sqrt[F_m.|N|]{R_{\mathrm{req}}(F_m)}. \tag{6}$$

$R_{\mathrm{up}}(F_m.n_{s(x)})$ is then pre-allocated to these un-allocated tasks. Notice that $F_m.n_{s(x)}$'s actual reliability value $R(F_m.n_{s(x)})$ should be close to (can be larger than, equal to, or less than) $R_{\mathrm{up}}(F_m.n_i)$.

(4) Considering that the reliability value of $R_{\text{up}}(F_m.n_{s(x)})$ is pre-allocated to the un-allocated tasks, the reliability of F_m is calculated by

$$R(F_m) = \prod_{x=1}^{y-1} R_{\text{up}}(F_m.n_{s(x)}) \times R(F_m.n_{s(y)}) \times \prod_{z=y+1}^{|N|} R_{\text{UFRA}}(F_m.n_{s(z)}).$$

(5) To assure the reliability requirement $R_{\text{req}}(F_m)$, the final reliability $R(F_m)$ must be larger than or equal to $R_{\text{req}}(F_m)$. That is,

$$R(F_m) = \prod_{x=1}^{y-1} R_{\text{up}}(F_m.n_{s(x)}) \times R(F_m.n_{s(y)}) \times \prod_{z=y+1}^{|N|} R_{\text{UFRA}}(F_m.n_{s(z)}) \geqslant R_{\text{req}}(F_m). \tag{7}$$

Based on Equation (7), the final reliability value of task $n_{s(y)}$ must meet

$$R(F_m.n_{s(y)}) \geqslant \frac{R_{\text{req}}(F_m)}{\prod\limits_{x=1}^{y-1} R_{\text{up}}(F_m.n_{s(x)}) \times \prod\limits_{z=y+1}^{|N|} R_{\text{UFRA}}(F_m.n_{s(z)})}. \tag{8}$$

(6) UFRA lets the reliability requirement of task $F_m.n_{s(y)}$ be the right half of Equation (8), namely,

$$R_{\text{req}}(F_m.n_{s(y)}) = \frac{R_{\text{req}}(F_m)}{\prod\limits_{x=1}^{y-1} R_{\text{up}}(F_m.n_{s(x)}) \times \prod\limits_{z=y+1}^{|N|} R_{\text{UFRA}}(F_m.n_{s(z)})}. \tag{9}$$

Following the above steps, the reliability requirement of the automotive function is transformed to that of each task. In other words, the Equation (4), namely,

$$R(F_m) = \prod_{F_m.n_i \in F_m.N} (R\,(F_m.n_i)) \geqslant R_{\text{req}}(F_m)$$

is not the concern for reliability assurance and should be replaced by

$$R(F_m.n_{s(y)}) \geqslant R_{\text{req}}(F_m.n_{s(y)}) \tag{10}$$

to simplify the solution. Through mathematical induction, we can prove that the function's reliability is still assured when using approximated reliability values of the tasks that have not been allocated. The similar proof to assure the function's reliability through mathematical induction has been provided in Reference [29]. Due to space limitation, this article no longer lists the proof process, and the readers can refer to the Appendix of Reference [29].

Let $LFT(F_m.n_i^{\beta}, u_k)$ and $LST(F_m.n_i^{\beta}, u_k)$ be the latest finish time (LFT) and the latest start time (LST), respectively, of the replica n_i^{β} in the ECU u_k, namely,

$$\begin{cases} LFT\left(F_m.n_{\text{exit}}^{\beta}, u_k\right) = D(F_m) \\ LFT\left(F_m.n_i^{\beta}, u_k\right) = \min\left\{ LAT[k], \min\limits_{F_m.n_j \in succ(n_i), \gamma \in [1, num_j)]} \left\{ AST(F_m.n_j^{\gamma}) - F_m.c'_{i,j} \right\} \right\} \end{cases} \tag{11}$$

and

$$LST\left(F_m.n_i^{\beta}, u_k\right) = LFT\left(F_m.n_i^{\beta}, u_k\right) - F_m.w_{i,k}. \tag{12}$$

$LAT[k]$ is the latest available time (LAT) of u_k in the current state and can be understood as the end time of the last slack on u_k. $AST(F_m.n_j^{\gamma})$ is the actual start time (AST) of the replica $F_m.n_j^{\gamma}$ and is calculated by

$$AST\left(F_m.n_j^{\gamma}\right) = LST\left(F_m.n_j^{\gamma}, u_{ecu(n_j^{\gamma})}\right). \tag{13}$$

In other words, $F_m.n_j^\gamma$ has different LSTs on each ECU. When $F_m.n_j^\gamma$ has been assigned to the ECU $u_{ecu(n_j^\gamma)}$, the AST of $F_m.n_j^\gamma$ is equal to the LST of $F_m.n_j^\gamma$ on $u_{ecu(n_h^\gamma)}$. $F_m.c_{i,j}^{'}$ represents the communication from $F_m.n_i^\gamma$ to $F_m.n_j^\beta$. Notice that $F_m.c_{i,j}^{'}$ is the WCRT $F_m.c_{i,j}$ from $F_m.n_i^\gamma$ to $F_m.n_j^\beta$ if $F_m.n_i^\gamma$ and $F_m.n_j^\beta$ are not allocated to the same ECU; otherwise, $F_m.c_{i,j}^{'}$ is 0, because the shared memory scheme is employed as explained in Section 2.2.

The final response time of F_m (e.g., $RT(F_m)$) is the deadline of the automotive function (e.g., $D(F_m)$) subtracting AST of the entry task $F_m.n_{\text{entry}}$ (i.e., $AST(F_m.n_{\text{entry}})$), and $AST(F_m.n_{\text{exit}})$ is the minimum AST among all the replicas of $F_m.n_{\text{entry}}$. Therefore, we have

$$RT(F_m) = D(F_m) - AST(F_m.n_{\text{entry}}) = D(F_m) - \min_{\beta \in [1, num_{\text{entry}}]} \{AST(F_m.n_{\text{entry}}^\beta)\}. \tag{14}$$

The strategy of UFRA is as follows: UFRA iteratively allocates the replica of current task n_i to an available ECU with the maximum LST (Equation (12)) until the reliability requirement of n_i (Equation (9)) is met. The allocated ECU $u_{\max}$ and corresponding $LST(F_m.n_i, u_{\max})$ value for $F_m.n_i$ are determined by the following:

$$LST(F_m.n_i, u_{\max}) = \max_{u_k \in U, u_k \textit{ is available}} \{LST(F_m.n_i, u_k)\}. \tag{15}$$

"u_k is available" means that no other replicas of $F_m.n_i$ have been allocated to u_k. The reason is that allocating multiple replicas of the same task to the same ECU is not allowed in active replication. The final number of replicas of the function is the sum of those of tasks, namely,

$$NR(F_m) = \sum_{i=1}^{F_m.|N|} NR(F_m.n_i). \tag{16}$$

$NR(F_m.n_i)$ can be obtained by the while loop in the UFRA algorithm. The detailed description of the UFRA algorithm is shown in Algorithm 1.

ALGORITHM 1: The UFRA Algorithm

```
Input:  $F_m = (N, W, E, C), U, R_{\text{req}}(F_m)$
Output: $RT(G)$, $R(G)$ and related values
 1: The task priority is ordered based on the increasing order of the upward rank value using Equation (5);
 2: for ($y \leftarrow 1; y \leqslant F_m.|N|; y{+}{+}$) do
 3:     $i \leftarrow s(y)$;
 4:     Calculate $R_{\text{up}}(F_m.n_i)$ using Equation (6);
 5:     Calculate $R_{\text{req}}(F_m.n_i)$ using Equation (9);
 6:     for ($k \leftarrow 1; k \leqslant |U|; k{+}{+}$) do
 7:         Calculate $R(F_m.n_i, u_k)$ for the task $F_m.n_i$ using Equation (1);
 8:         Calculate $LST(F_m.n_i, u_k)$ for the task $F_m.n_i$ using Equation (15);
 9:     end for
10:     while ($R(F_m.n_i) < R_{\text{req}}(F_m.n_i)$) do
11:         Select available replica $n_i^\varepsilon$ and ECU $u_{ecu(F_m.n_i^\varepsilon)}$ with the maximum LST;
12:         Calculate $R(F_m.n_i)$ using Equation (3);
13:     end while
14: end for
15: Calculate $RT(F_m)$ using Equation (14);
16: Calculate $NR(F_m)$ using Equation (16);
17: Calculate $R(F_m)$ using Equation (4);
```

Table 3. Task Allocations of F_1 Using UFRA

n_i	$R_{\text{req}}(F_1.n_i)$	$LST(F_1.n_i, u_1)$	$LST(F_1.n_i, u_2)$	$LST(F_1.n_i, u_3)$	num_i	$R(F_1.n_i)$
n_6	0.99998333	$\boxed{245}$	240	$\boxed{243}$	2	0.99999686
n_4	0.99996980	$\boxed{228}$	221	$\boxed{222}$	2	0.99998999
n_5	0.99996315	$\boxed{210}$	$\boxed{214}$	$\boxed{213}$	3	0.99999986
n_3	0.99994662	$\boxed{190}$	$\boxed{187}$	183	2	0.99999355
n_2	0.99993641	157	$\boxed{158}$	$\boxed{162}$	2	0.99994426
n_1	0.99997548	$\boxed{141}$	138	$\boxed{140}$	2	0.99999355
$RT(F_1) = 110$, $NR(F_1) = 13$, $R(F_1) = 0.99991807$						

Table 4. Properties of Four Automotive Functions in Figure 2

Function	F_1	F_2	F_3	F_4
$D(F_m)$	250	250	250	250
$LB(F_m)$	110	92	86	41
$L(F_m)$	140	158	164	209

Table 3 shows the task allocations of F_1 using UFRA. We assume that the failure rates for three ECUs are as follows: $\lambda_1 = 0.0001$, $\lambda_2 = 0.0006$, and $\lambda_3 = 0.0009$. The reliability requirement of F_1 is $R_{\text{req}}(F_1) = 0.9999$, which is the reliability requirement for E1 as listed in Table 2. The deadline of F_1 is $D(F_m) = 250$. The task priority order is $F_1.n_6$, $F_1.n_4$, $F_1.n_5$, $F_1.n_3$, $F_1.n_2$, and $F_1.n_1$.

Each row shows the reliability requirement $R_{\text{req}}(F_1.n_i)$, allocated ECUs with LSTs (in boxed), number of replicas num_i, and final reliability value $R(F_1.n_i)$ of each task.

(1) The reliability requirement of $F_1.n_6$ is $R_{\text{req}}(F_1.n_6) = 0.99998333$ calculated by Equation (9). UFRA iteratively allocates the replica of current task $F_1.n_6$ to u_1 and u_3 with the maximum LSTs of 245 and 243 until the reliability requirement of $F_1.n_6$ is met. The final reliability value of $F_1.n_6$ is 0.99999686.

(2) The reliability requirement of $F_1.n_4$ is $R_{\text{req}}(F_1.n_4) = 0.99996980$ calculated by Equation (9). UFRA iteratively allocates the replica of current task $F_1.n_4$ to u_1 and u_3 with the maximum LSTs of 228 and 222 until the reliability requirement of $F_1.n_4$ is met. The final reliability value of $F_1.n_4$ is 0.99998999.

(3) After that $F_1.n_5$, $F_1.n_3$, $F_1.n_2$, and $F_1.n_1$ adopt the same strategy as $F_1.n_6$ and $F_1.n_4$, the final response time of the automotive function F_1 is $RT(F_1) = 110$ calculated by Equation (14). The final number of replicas and reliability value of the automotive function F_1 are $NR(F_1) = 13$ and $R(F_1) = 0.99991807$, calculated by Equations (16) and (4), respectively.

Table 4 lists the lower bounds of four automotive functions in Figure 2, where we have $LB(F_1) = 110$, $LB(F_2) = 92$, $LB(F_3) = 86$, and $LB(F_4) = 41$. In this example, the deadlines of all automotive functions are 250, namely, $D(F_1) = D(F_2) = D(F_3) = D(F_4) = 250$.

3.2 Refined Exploration for Priority Sequence

Considering that ACPS contains multiple safety-critical automotive functions, we should make as many as automotive functions be finished under their functional safety requirement in the shared ACPS platform. Using a suitable strategy to determine the priority sequence of safety-critical functions will directly affect the above objectives. Priority sequence of functions means to allocate a priority, which is a positive integer, to each automotive function, and then sort these automotive functions according to their priority values. The higher the priority of an automotive function, the more forward its position in the sequence. Two objectives need to be considered in multi-functional static scheduling: (1) make as many as automotive functions be finished under their functional safety requirements, and (2) shorten the overall response time of ACPS.

Currently, there are two priority sequence strategies widely used, namely, earliest deadline first (EDF), which means that shorter deadline has higher priority [3], and least laxity first (LLT), which means that less laxity has higher priority [24]. The laxity means the span value that the deadline subtract the lower bound of the automotive function, namely,

$$L(F_m) = D(F_m) - LB(F_m). \tag{17}$$

Table 4 shows the laxity of each automotive function, namely, $L(F_1) = 140$, $L(F_2) = 158$, $L(F_3) = 164$, and $L(F_4) = 209$. According to the summary in Reference [33], the priorities of automotive functions are based on LLT, namely, the less the laxity value, the higher the priority. When two automotive functions have the same LLT, then the priorities of automotive function are based on EDF, namely, the shorter the deadline value, the higher the priority. We name such priority sequence strategy as LLT. Therefore, the priority sequence using LLT in Figure 2 is (F_1, F_2, F_3, F_4).

The overall response time of four automotive functions using LLT is 214. However, LLT is not the best choice in general. For example, there are a total of 4! = 24 priority sequences for four automotive functions. Table 5 shows the overall response time values of four automotive functions for different priority sequences. By exhausting all sequences to obtain the overall response time values in these 24 cases, we can find that (F_1, F_3, F_2, F_4) is the optimal priority sequence, which has the minimum overall response time of 210. For ACPS with only a few automotive functions, exhausting all priority sequences (i.e., the exhaustive exploration method) may be feasible. However, ACPS usually contains dozens of automotive functions in a subsystem, and it is very unrealistic to still use the exhaustive exploration method. For example, for a subsystem containing 10 safety-critical automotive functions, there are a total of 10! = 3,628,800 priority sequences; however, our experiments report that exhausting these sequences must need 1,260 days.

In summary, we have the following basic conclusions: (1) The LLT method is a simple and fast method, but the performance is not very satisfactory; (2) The exhaustive exploration method can find the optimal solution, but can not be applied to large-scale ACPS. To cope with the problems when using the above methods, this article proposes a non-exhaustive exploration method to find the approximate optimal priority sequence. "Non-exhaustive exploration" means that we skip most of the priority sequences and only choose the approximate optimal sequence from a small number of priority sequences. The sequence is approximately optimal, because the real optimal sequence may be skipped.

Rule 1. *Index forward from the individual last positions of the two sequences until different application identifiers are indexed at the same index location by comparing two sequences. The sequence with longer response time will be discarded, and the subsequence starting from the first different indexed identifier in the discarded sequence will be added to the skipped subsequence set. The following sequences ending with any such subsequence will be skipped.*

In the following, we propose how to skip most of the priority sequences and explain this method with the motivational functions in Figure 2 by Table 5.

(1) We consider the first sequence (F_1, F_2, F_3, F_4), which has the overall response time of 214.
(2) We consider the second sequence (F_1, F_2, F_4, F_3), which has the overall response time of 244. In this case, we conduct Rule 1 that the following sequences ending with the subsequence (F_3) will be skipped.

Table 5. Discarding and Skipping Priority Sequences for the Motivational Functions in Figure 2

	function priority sequence	Overall response time	The sequences ending with the following subsequences will be skipped	skipping sequence?
1	**(F_1, F_2, F_3, F_4)**	**214**		**no**
2	**(F_1, F_2, F_4, F_3)**	**244**	**(F_3)**	**no**
3	**(F_1, F_3, F_2, F_4)**	**210**	**$(F_3), (F_3, F_4)$**	**no**
4	**(F_1, F_3, F_4, F_2)**	**228**	**$(F_3), (F_3, F_4), (F_2)$**	**no**
5	(F_1, F_4, F_3, F_2)	247	$(F_3), (F_3, F_4), (F_2)$	yes
6	(F_1, F_4, F_2, F_3)	247	$(F_3), (F_3, F_4), (F_2)$	yes
7	(F_2, F_1, F_3, F_4)	217	$(F_3), (F_3, F_4), (F_2)$	yes
8	(F_2, F_1, F_4, F_3)	235	$(F_3), (F_3, F_4), (F_2)$	yes
9	**(F_2, F_3, F_1, F_4)**	**232**	**$(F_3), (F_3, F_4), (F_2), (F_1, F_4)$**	**no**
10	**(F_2, F_3, F_4, F_1)**	**-**	**$(F_3), (F_3, F_4), (F_2), (F_1, F_4), (F_1)$**	**no**
11	(F_2, F_4, F_3, F_1)	-	$(F_3), (F_3, F_4), (F_2), (F_1, F_4), (F_1)$	yes
12	(F_2, F_4, F_1, F_3)	231	$(F_3), (F_3, F_4), (F_2), (F_1, F_4), (F_1)$	yes
13	(F_3, F_2, F_1, F_4)	221	$(F_3), (F_3, F_4), (F_2), (F_1, F_4), (F_1)$	yes
14	(F_3, F_2, F_4, F_4)	247	$(F_3), (F_3, F_4), (F_2), (F_2, F_4), (F_1)$	yes
15	**(F_3, F_1, F_2, F_4)**	**221**	**$(F_3), (F_3, F_4), (F_2), (F_2, F_4), (F_1), (F_1, F_2, F_4)$**	**no**
16	(F_3, F_1, F_4, F_2)	-	$(F_3), (F_3, F_4), (F_2), (F_2, F_4), (F_1), (F_1, F_2, F_4)$	yes
17	(F_3, F_4, F_1, F_2)	242	$(F_3), (F_3, F_4), (F_2), (F_2, F_4), (F_1), (F_1, F_2, F_4)$	yes
18	(F_3, F_4, F_3, F_1)	244	$(F_3), (F_3, F_4), (F_2), (F_2, F_4), (F_1), (F_1, F_2, F_4)$	yes
19	(F_4, F_2, F_3, F_1)	-	$(F_3), (F_3, F_4), (F_2), (F_2, F_4), (F_1), (F_1, F_2, F_4)$	yes
20	(F_4, F_2, F_1, F_3)	224	$(F_3), (F_3, F_4), (F_2), (F_2, F_4), (F_1), (F_1, F_2, F_4)$	yes
21	(F_4, F_3, F_2, F_1)	-	$(F_3), (F_3, F_4), (F_2), (F_2, F_4), (F_1), (F_1, F_2, F_4)$	yes
22	(F_4, F_3, F_1, F_2)	-	$(F_3), (F_3, F_4), (F_2), (F_2, F_4), (F_1), (F_1, F_2, F_4)$	yes
23	(F_4, F_1, F_3, F_4)	238	$(F_3), (F_3, F_4), (F_2), (F_2, F_4), (F_1), (F_1, F_2, F_4)$	yes
24	(F_4, F_1, F_2, F_3)	245	$(F_3), (F_3, F_4), (F_2), (F_2, F_4), (F_1), (F_1, F_2, F_4)$	yes

For instance, the indexed location for two sequences (F_1, F_2, F_3, F_4) and (F_1, F_2, F_4, F_3) is the last position and the different application identifiers are F_4 and F_3. As the overall response time value of sequences (F_1, F_2, F_3, F_4) and (F_1, F_2, F_4, F_3) are 214 and 244, respectively, sequence (F_1, F_2, F_4, F_3) must be discarded, because it has longer response time than sequence (F_1, F_2, F_3, F_4). Therefore, F_3 will be added to the skipped subsequence set, namely, *skipped_subsequence_set* = $\{(F_3)\}$. The following sequences ending with (F_3) will be skipped.

(3) We consider the third sequence (F_1, F_3, F_2, F_4), which has the overall response time of 210. As 210 is shorter than 214 of sequence (F_1, F_2, F_3, F_4), sequence (F_1, F_2, F_3, F_4) muse be discarded. According to Rule 1, the subsequence (F_3, F_4) will be added to the skipped subsequence set, namely, *skipped_subsequence_set* = $\{(F_3), (F_3, F_4)\}$. In this case, the following sequences ending with subsequences (F_3) and (F_3, F_4) will be skipped.

(4) We consider the fourth sequence (F_1, F_3, F_4, F_2), which has the overall response time of 228. As 228 is longer than 210 of (F_1, F_3, F_2, F_4), sequence (F_1, F_3, F_4, F_2) must be discarded. According to Rule 1, the subsequence (F_2) will be added to the skipped subsequence set, namely, *skipped_subsequence_set* = $\{(F_3), (F_3, F_4), (F_2)\}$.

(5) Considering that the fifth sequence (F_1, F_4, F_3, F_2) ends with subsequence (F_2), this sequence is skipped. Similarity, sixth sequence (F_1, F_4, F_2, F_3) ending with subsequence (F_3),

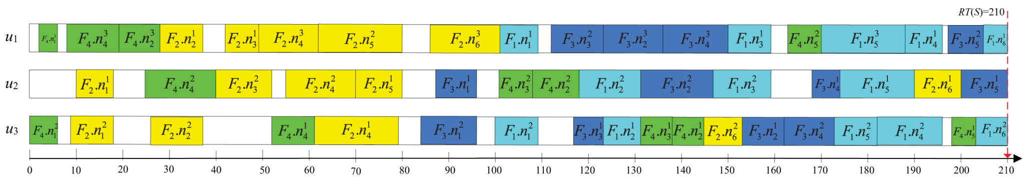

Fig. 4. Task mapping of four functions in three ECUs during the design phase.

and the seventh sequence (F_2, F_1, F_3, F_4) ending with subsequence (F_3, F_4), and the eighth (F_2, F_1, F_4, F_3) ending with subsequence (F_3) will be skipped.

(6) We consider the ninth sequence (F_2, F_3, F_1, F_4), which has the overall response time of 232. As 232 is longer than 210 of sequence (F_1, F_3, F_2, F_4), sequence (F_2, F_3, F_1, F_4) must be discarded. According to Rule 1, the subsequence (F_1, F_4) will be added to the skipped subsequence set, namely, *skipped_subsequence_set* = {(F_3), (F_3, F_4), (F_2), (F_1, F_4)}.

(7) We consider the 10th sequence (F_2, F_3, F_4, F_1), the overall response time of which exceeds the deadline of 250. Therefore, sequence (F_2, F_3, F_4, F_1) must be discarded. According to Rule 1, the subsequence (F_1) will be added to the skipped subsequence set, namely, *skipped_subsequence_set* = {(F_3), (F_3, F_4), (F_2), (F_1, F_4), (F_1)}.

(8) Similarity, the following four sequences are skipped until (F_3, F_1, F_2, F_4), which has the overall response time of 221. As 221 is longer than 210 of (F_1, F_3, F_2, F_4), sequence (F_3, F_1, F_2, F_4) must be discarded. According to Rule 1, the subsequence (F_1, F_2, F_4) will be added to the skipped subsequence set, namely, *skipped_subsequence_set* = {(F_3), (F_3, F_4), (F_2), (F_1, F_4), (F_1), (F_1, F_2, F_4)}.

(9) All the remaining sequences are discarded, because these sequences end with one of the subsequences of *skipped_subsequence_set* = {(F_3), (F_3, F_4), (F_2), (F_1, F_4), (F_1), (F_1, F_2, F_4)}.

Through the above process, only seven sequences are reserved (denoted with red color), and the remaining 17 sequences are discarded. The discarded rate is 70.83%. (F_1, F_3, F_2, F_4) is chosen as the final priority sequence, because it has the least overall response time among the seven reserved sequences. Therefore, the above heuristic method proposed in this phase to determine the mapping order of functions is a novel contribution.

We propose the refined exploration method shown in Algorithm 2 (i.e., the RE algorithm) to discard most of the sequences and only selected the approximate optimal priority sequence from the reserved small number of sequences.

Figure 4 shows the task mapping of four functions in three ECUs during the design phase using RE when the exposure is E1 (i.e., the reliability requirement is 0.9999). As can be seen from Figure 4, the overall response time of four functions is $RT(S) = 210$.

4 HUMAN-INTERACTION DURING RUNTIME PHASE

4.1 Human-Interaction Process between Driver and ACPS

Table 6 shows the severity, exposure, and controllability classifications provided by ISO 26262. There are four controllability levels of C0 (controllable in general), C1 (simply controllable), C2 (normally controllable), and C3 (difficult to control or uncontrollable). Severity is an inherent safety attribute of an automotive function that has been identified during the hazard analysis and risk assessment phase, such that it will not be changed throughout the development lifecycle. Exposure can be changed for an automotive function. In general, exposure can be reduced by increasing redundancy to increase reliability. For example, when the reliability value is increased from 0.9 to

Table 6. Severity, Exposure, and Controllability Classifications Provided by ISO 26262 [2]

Severity		Exposure		Controllability	
S0	No injuries	E0	Incredibly unlikely	C0	Controllable in general
S1	Light to moderate injuries	E1	Very low probability	C1	Simply controllable
S2	Severe to life-threatening injuries	E2	Low probability	C2	Normally controllable
S3	Life-threatening to fatal injuries	E3	Medium probability	C3	Difficult to control or uncontrollable
		E4	High probability		

ALGORITHM 2: The RE Algorithm

```
Input: $U = \{u_1, u_2, \ldots, u_{|U|}\}$, $S = \{F_1, F_2, \ldots, F_{|S|}\}$
Output: Approximate optimal priority sequence approximate_optimal_sequence and related values
1: reserved_sequence_set ← NULL;
2: skipped_subsequence_set ← NULL;
3: approximate_optimal_sequence ← $(F_1, F_2, \ldots, F_{|S|-1}, F_{|S|})$;
4: reserved_sequence_set. add( approximate_optimal_sequence);
5: while (there is sequence has not been considered in all sequences) do
6:     Obtain a currently un-considered sequence current_sequence;
7:     if (current_sequence ends with the any subsequence in skipped_subsequence_set) then
8:         continue; //skip current_sequence
9:     end if
10:    if ($RT_{\text{current\_sequence}}(S) < RT_{\text{approximate\_optimal\_sequence}}(S)$) then
11:        reserved_sequence_set. add( current_sequence);
12:        Obtain the subsequence of approximate_optimal_sequence based on Rule 1;
13:        skipped_subsequence_set. add(subsequence);
14:        approximate_optimal_sequence ← current_sequence;
15:        $RT_{\text{approximate\_optimal\_sequence}}(S) \leftarrow RT_{\text{current\_sequence}}(S)$;
16:    else
17:        Obtain the subsequence of current_sequence based on Rule 1;
18:        skipped_subsequence_set. add(subsequence);
19:    end if
20: end while
```

0.9999, and the exposure can be reduced from E4 (high probability) to E1 (very low probability) according to Table 2.

Table 7 lists the ASIL determination based on severity, exposure, and controllability provided by ISO 26262 [2]. Besides ASIL A, ASIL B, ASIL C, and ASIL D, there is another level of quality management (QM), which does not involve safety requirement design as explained in ISO 26262. In other words, only QM is enough to develop a function. As can be seen from Table 7, once the driver's controllability changes, the ASIL of the application will change accordingly. Therefore, there is a human-interaction process between the driver and the system as pointed out in Section 1. However, this change brings instability and variables to the safe operation of safety-critical applications.

For example, assume that an automotive function has severity S3 and require the level of ASIL A. when the driver's controllability is C3, its exposure must reach E1 according to Table 7; otherwise, the ASIL of automotive function will be promoted to ASIL B, ASIL C, or ASIL D, where the risk of harm to the automotive function will increase significantly.

Table 7. ASIL Determination Based on Severity, Exposure, and Controllability Provided by ISO 26262 [2]

Severity	Exposure	Controllability		
		C1	C2	C3
S1	E1	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	ASIL A
	E4	QM	ASIL A	ASIL B
S2	E1	QM	QM	QM
	E2	QM	QM	ASIL A
	E3	QM	ASIL A	ASIL B
	E4	ASIL A	ASIL B	ASIL C
S3	E1	QM	QM	ASIL A
	E2	QM	ASIL A	ASIL B
	E3	ASIL A	ASIL B	ASIL C
	E4	ASIL B	ASIL C	ASIL D

4.2 Human-Interaction-Aware Task Remapping

To maintain the safe operations of safety-critical functions and implement system optimization during the runtime phase, a human-interaction-aware adaptive functional safety processing is required. This subsection achieves the above objective by proposing the human-interaction-aware task remapping method.

Figure 4 has shown the task mapping of four automotive functions during the design phase. We still assume that the severity is S3 for four safety-critical functions. Since the driver's state during the design phase is pessimistically set to C3 (difficult to control or uncontrollable), the reliability requirements of automotive functions must reach 0.9999 (E1) to meet the required level of ASIL A according to the ASIL determination in Table 7. A possible scenario is that the driver's state is changed to C1 (simply controllable) at time instant of 100 during the runtime phase. At this time instant, it is also feasible that the reliability requirements of four automotive functions can be changed to 0.95 (E3) while still meeting the required level of ASIL A.

In response to the above scenario, the task remapping is conducted at the time instant of 100. It should be noted that the tasks can be divided into three categories.

(1) Tasks that have been finished (e.g., $F_3.n_1^1$, $F_3.n_1^2$, and preceding replicas in Figure 4).
(2) Tasks that are being executed (e.g., $F_2.n_6^3$ in Figure 4).
(3) Tasks that have not been started (e.g., $F_1.n_1^1$, $F_4.n_3^2$, $F_1.n_1^2$, and subsequent replicas in Figure 4).

The tasks that have been finished certainly cannot be remapped. Tasks that have not been started can definitely be remapped. For tasks that are being executed (e.g., $F_2.n_6^3$ in Figure 4), there are two possible strategies. The first strategy is to continue to perform these tasks according to the previous mapping. The second strategy is that only the un-finished task segments can be re-allocated, whereas the already finished task segments cannot be re-allocated. Considering that the time is required for ECU flashing, re-mapping when the tasks are getting executed is unrealistic. Therefore, to reduce the complexity of the analysis and design, this article adopts the first strategy.

Figure 5 shows the human-interaction-aware task remapping of four functions in three ECUs during the runtime phase. From Figure 5, the task remapping begins at time instant of 100. On the

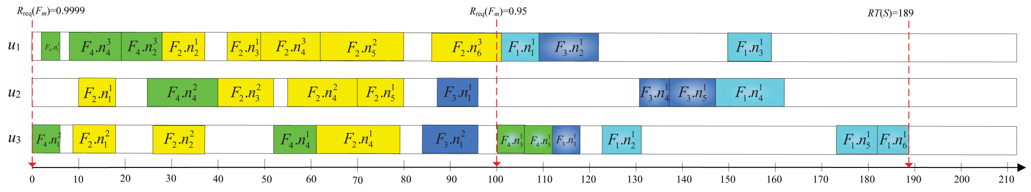

Fig. 5. Human-interaction-aware Task remapping of four functions in three ECUs during the runtime phase.

one hand, since the reliability requirements of four automotive functions have been reduced to 0.95, the reliability requirements of the tasks that require remapping could also be correspondingly reduced. The upper bound on reliability of these tasks is calculated by

$$R_{\text{up}}(F_m.n_{s(z)}) = \sqrt[F_m.|N| - F_m.|N(f)|]{R_{\text{req}}(F_m)}, \tag{18}$$

where $F_m.|N(f)|$ represents the finished task number in F_m, such that $F_m.|N| - F_m.|N(f)|$ represents the number of tasks that require remapping in F_m.

On the other hand, since some tasks have been finished, these tasks are no longer considered. Therefore, only the tasks that have not been started at current time need to be considered in reliability requirement calculation during the runtime phase. The reliability requirement of $F_m.n_{s(y)}$ is calculated by

$$R_{\text{req}}(F_m.n_{s(y)}) = \frac{R_{\text{req}}(F_m)}{\prod\limits_{x=(F_m.|N(f)|+1)}^{y-1} R_{\text{HR}}(F_m.n_{s(x)}) \times \prod\limits_{z=y+1}^{|N|} R_{\text{up}}(F_m.n_{s(z)})}. \tag{19}$$

Notice that the task priority is based on the ascending order of the AST values before remapping. For example, the task priority for the tasks that require remapping in F_3 is $F_3.n_3$, $F_3.n_2$, $F_3.n_4$, and $F_3.n_5$.

To reduce the remapping overhead during the runtime phase, the priority sequence of functions still adopts what has been obtained during the design phase. For the motivational functions in Figure 2, the priority sequence of functions is still (F_1, F_3, F_2, F_4).

We have the following observations based on the human-interaction-aware task remapping in Figure 5.

(1) The number of redundancy for function F_1 is reduced from 13 to 6, for function F_2 is reduced from 14 to 12, for function F_4 is reduced from 12 to 8.
(2) The overall response time of four functions was shortened from 210 to 189.
(3) The total slack time of three ECUs has increased dramatically from 230 to 420.

Through human-interaction-aware task remapping, we implemented a human-interaction-aware adaptive functional safety processing to reach the following system optimization: (1) reduce overall task redundancy, (2) shorten the overall response time of safety-critical automotive functions, and (3) increase the slack time for non-safety-critical automotive functions. Although the proposed methodology only demonstrates the increase in exposure (i.e., decrease in reliability) because the driver's controllability is changed from C3 to C1 by an example, this methodology is also adapted to the scenario where the driver's controllability is changed from C1 to C3 and the exposure will be reduced. Due to space limitation, no longer description is provided here.

Finally, we propose the human-interaction-aware task remapping method shown in Algorithm 3 (i.e., the HR algorithm) to achieve human-interaction-aware adaptive functional safety processing.

ALGORITHM 3: The HR algorithm

```
Input: Task mapping generated by RE method,
Output: Task remapping
1: $current\_time \leftarrow 0$;
2: $deadline \leftarrow \max\{D(F_1), D(F_1), \ldots, D(F_{|S|})\}$;
3: while ($current\_time < deadline$) do
4:   if (driver's current state is changed in $current\_time$ ) then
5:     Obtain the driver's controllability level belonging C1, C2, or C3;
6:     if (the driver's controllability level is C1) then
7:       Set the exposure level of all safety-critical functions to E3 according to Table 7;
8:       Set the reliability requirement of all safety-critical functions to 0.95 according to Table 2;
9:     end if
10:    if (the driver's controllability level is C2) then
11:      Set the exposure level of all safety-critical functions to E2 according to Table 7.
12:      Set the reliability requirement of all safety-critical functions to 0.99 according to Table 2;
13:    end if
14:    if (the driver's controllability level is C3) then
15:      Set the exposure level of all safety-critical functions to E1 according to Table 7;
16:      Set the reliability requirement of all safety-critical functions to 0.9999 according to Table 2;
17:    end if
18:    Calculate the new upper bound on reliability $R_{\text{up}}(F_m.n_{s(z)})$ by Equation (18) of the tasks that have not been started;
19:    Calculate new reliability requirement $R_{\text{up}}(F_m.n_{s(z)})$ by Equation (19) of the tasks that have not been started;
20:    Perform task remapping for the tasks that have not been started.
21:  end if
22: end while
```

Notice that the required ASILs for all safety-critical functions are ASIL A, and the severity levels for all safety-critical functions are S3 in this article.

5 EXPERIENTIAL EVALUATION

5.1 Methods and Metrics for Comparison

Four methods are adopted for comparison in this article.

(1) The first method is adopting the LLT strategy proposed in Reference [33] to obtain the priority sequence of safety-critical functions and perform the task mapping during the design phase; we name this method as LLT in this experiment.
(2) The second method is the exhaustive exploration method during the design phase, and we name it as EE in this experiment.
(3) The third method is the refined exploration method proposed in this article to obtain the approximate optimal priority sequence of safety-critical functions and perform the task mapping during the design phase; we name this method as RE in this experiment.
(4) The fourth method is the human-interaction-aware task remapping method proposed in this article to perform the task remapping during the runtime phase; we name this method as HR in this experiment.

The performance metrics are overall task redundancy of safety-critical automotive functions while meeting their functional safety requirements (expressed by $NR(S)$), overall response time of safety-critical automotive functions (expressed by $RT(S)$), and slack time for non-safety-critical

Table 8. Result of LLT, Exhaustive Exploration, and RE during the Design Phase When the Total Number of Safety-critical Functions Is 6

Method	LLT	EE		RE	
Metric	$RT(S)$	$RT(S)$	iteration count	$RT(S)$	iteration count
Test 1	12,489ms	11,729ms	720	11,729ms	16
Test 2	8,374ms	7,808ms	720	7,808ms	16
Test 3	11,924ms	11,039ms	720	11,326ms	17
Test 4	6,300ms	5,855ms	720	5,855ms	18
Test 5	9,894ms	9,283ms	720	9,283ms	16
Test 6	9,708ms	9,130ms	720	9,130ms	17
Test 7	11,245ms	10,213ms	720	10,213ms	16
Test 8	11,459ms	10,322ms	720	10,322ms	16
Test 9	10,090ms	9,453ms	720	9,453ms	16
Test 10	9,497ms	8,783ms	720	8,783ms	17

automotive functions (expressed by $Slack(S)$). The task and message parameters refer to the realistic values in Reference [34] and are generated by uniform distribution: $100\mu s \leqslant w_{i,k} \leqslant 400\mu s$, $100\mu s \leqslant c_{i,j} \leqslant 400\mu s$. The number of tasks for a function is from 8 to 29 generated by uniform distribution. The ECU number of the subsystem is 12. All the functions are synthetic and are generated by using the task graph generator from Reference [10].

5.2 Experiential Results and Analysis

Experiment 1. In this experiment, we consider 6 safety-critical functions in a subsystem of ACPS. Therefore, the total sequence number of safety-critical functions is 6! = 720. The severity of all these safety-critical functions is S3. The driver's controllability level is fixed at C3 during the design phase and is changed to C1 at time instant of 2,000ms during the runtime phase. Ten tests are conducted based on the randomly generated parameters.

Table 8 shows the results of LLT, EE, and RE during the design phase when the total number of safety-critical functions is 6. As the total sequence number is 720, the three methods can return the results within 5s.

(1) LLT always generates longer overall response time than ER and RE in 10 tests. The minimum and maximum differences are 445ms (Test 4) and 1,137ms (Test 8). These results confirm that LLT is not a good choice to be a priority sequences method, especially for ACPS with extremely high time accuracy.
(2) Except for Test 3, EE and RE get equal overall response time values in the other 9 tests. Due to the unequalness in Test 3, RE is not an optimal solution in finding the optimal sequence toward overall response time minimization. In other words, real optimal sequence may be skipped when using RE. Considering that RE obtains the optimal overall response time values in 9 of 10 tests, it is shown that using RE is an approximate optimal solution.
(3) The iteration count is from 16 to 18 in total count of 720. That is, the refinement ratio is from 97.5% to 97.8%.

In summary, the proposed RE generates approximate optimal response time with small iteration count, whereas LLT-generated response time is not approximate optimal and EE-generated iteration count is exhaustive.

Table 9. Result of RE during the Design Phase and HR during the Runtime Phase When the Total Number of Safety-critical Functions Is 6

Method	RE start at 0ms			HR controllability changes at 2,000ms		
Metric	$NR(S)$	$RT(S)$	$Slack(S)$	$NR(S)$	$RT(S)$	$Slack(S)$
Test 1	236	11,729ms	132,375ms	119	9,003ms	145,037ms
Test 2	158	7,808ms	144,749ms	80	5,306ms	152,721ms
Test 3	274	11,326ms	126,896ms	139	8,016ms	141,489ms
Test 4	150	5,855ms	146,410ms	76	4,052ms	154,070ms
Test 5	210	9,283ms	138,761ms	107	6,870ms	147,900ms
Test 6	218	9,130ms	138,190ms	111	6,523ms	147,465ms
Test 7	250	10,213ms	133,889ms	126	7,621ms	144,629ms
Test 8	238	10,322ms	134,281ms	120	7,428ms	144,380ms
Test 9	242	9,453ms	134379.0ms	122	6990.0ms	145947.0ms
Test 10	192	8,783ms	140,897ms	97	6,922ms	149,138ms

Table 9 shows the result of RE during the design phase and HR during the runtime phase when the total number sequences safety-critical functions is 6. The driver's controllability changes from C3 to C1 at 2000ms.

(1) HR generates less overall task redundancy than RE in all the tests, where the overall task redundancy generated by HR is almost only half of that by RE. The reason is that when the driver's controllability changes from C3 to C1, its reliability requirement is changed from 0,9999 to 0.95. Lower reliability requirements are naturally assured with less overall task redundancy.
(2) Similarly to the reduction in overall task redundancy, HR generates shorter overall response time than RE in all the tests. The reduction in overall response time is also significant for HR. These results confirm the bi-criteria between response time minimization and reliability maximization, as stated in Figure 3.
(3) The slack time values increase after HR is used because of the overall task redundancy reduction. This is useful, because non-safety-critical automotive functions have more slack time to execute.

Experiment 2. In this experiment, we consider 10 safety-critical functions in a subsystem of ACPS. Therefore, the total sequence number of safety-critical functions is 10! = 3,628,800. According to the iteration count and iteration time generated by RE of this experiment, one iteration takes about 0.5s, and exhausting 3,628,800 sequences takes about 1,814,400s (i.e., 1,260 days). Therefore, the results of exhaustive exploration method are no longer listed. Similarly to Experiment 1, the severity of all these safety-critical functions is still S3. The driver's controllability level is still fixed at C3 during the design phase and is changed to C1 at time instant of 2,000ms during the runtime phase. Ten tests are conducted based on the randomly generated parameters.

Table 10 shows the result of LLT and RE during the design phase when the total number of safety-critical functions is 10. Although the results of 10 tests are not the same, they generally show some stability.

(1) The overall task redundancy of safety-critical automotive functions generated by LLT and RE are basically the same in most tests except for the Test 9 that LLT has one replica less than RE.

Table 10. Result of LLT and RE during the Design Phase When the Total Number of Safety-critical Functions is 10

Method	LLT			RE				
Metric	$NR(S)$	$RT(S)$	$Slack(S)$	$NR(S)$	$RT(S)$	$Slack(S)$	Iteration count	Iteration time
Test 1	360	12,836ms	114,646ms	360	11,267ms	115,693ms	48	23s
Test 2	432	13,493ms	104,355ms	432	12,549ms	103,950ms	56	34s
Test 3	396	12,719ms	109,278ms	396	12,596ms	108,773ms	55	24s
Test 4	340	11,424ms	118,915ms	340	9,718ms	119,723ms	51	21s
Test 5	324	12,354ms	121,234ms	324	11,554ms	119,723ms	47	20s
Test 6	362	12,719ms	115,412ms	362	12,671ms	114,283ms	64	22s
Test 7	370	12,087ms	113,853ms	370	11,221ms	113,405ms	46	21s
Test 8	420	13,149ms	108,559ms	420	12,475ms	106,658ms	63	25s
Test 9	316	10,864ms	122,631ms	317	10,359ms	122,418ms	48	18s
Test 10	356	10,475ms	117,134ms	356	9,896ms	116,857ms	47	20s

(2) Different from overall task redundancy, the overall response time of safety-critical automotive functions generated by LLT and RE are not the same at all. RE always has shorter overall response time than LLT in 10 tests. When using LLT, the overall response time ranges from 10,475ms to 13,493ms; while using RT, it ranges from 9,718ms to 12,671ms. The maximum difference occurs in Test 4, where RE has shorter 1,706ms than LLT. These results indicate that such a simple strategy (i.e., LLT) of priority sequence of functions is not sufficient to find suitable priority sequence during the design phase, but a non-exhaustive exploration method (e.g., RE) can find the approximate optimal priority sequence. The RE method proposed in this article achieves the above purpose by refining the number of iterations.

(3) In overall, LLT has more available slack time than RE is most cases. For example, LLT has more slack time than RE in most tests (i.e., Tests 2, 3, 5, 6, 7, 8, 9, and 10), whereas RE has more slack time than LLT in two tests (i.e., Tests 1 and 4). The results of the above three experiments reflect that the optimization objective for RE is overall response time rather than the overall task redundancy and the slack time.

(4) The iteration count for RE is merely from 47 to 56 in total counts of 3,628,800. Therefore, about 99.9985%–99.9987% sequences are skipped (i.e., refinement ratio) using RE. The RE method proposed in this article reflects the powerful non-exhaustive exploration ability. In addition, the iteration time is only within 24s.

Table 11 shows the result of RE during the design phase and HR during the runtime phase when the total number sequences safety-critical functions is 10. The driver's controllability changes from C3 to C1 at 2,000ms.

(1) Similarly to the result of overall task redundancy in Table 9, HR also generates less overall task redundancy than RE in all the tests. Especially under some tests (i.e., Test 1, Test 4, Test 9, and Test 10), the overall task redundancy generated by HR is almost only half of that by RE.

(2) Similarly to the result of overall response time Table 9, HR also generates shorter overall response time than RE in all the tests. The maximum and minimum reduction in overall response time occur at Test 3 (reducing 4,397ms) and Test 4 (reducing 2,402ms),

Table 11. Result of RE during the Design Phase and HR During the Runtime Phase When the Total Number of Safety-critical Functions Is 10

Method	RE start at 0ms			HR controllability changes at 2,000ms		
Metric	$NR(S)$	$RT(S)$	$Slack(S)$	$NR(S)$	$RT(S)$	$Slack(S)$
Test 1	360	11,267ms	115,693ms	183	8,316ms	135,005ms
Test 2	432	12,549ms	103,950ms	224	8,066ms	127,387ms
Test 3	396	12,596ms	108,773ms	202	8,199ms	131,025ms
Test 4	340	9,718ms	119,723ms	171	7,353ms	136,428ms
Test 5	324	11,554ms	119,723ms	168	7,491ms	137,214ms
Test 6	362	12,671ms	114,283ms	192	8,720ms	134,187ms
Test 7	370	11,221ms	113,405ms	195	6,983ms	134,927ms
Test 8	420	12,475ms	106,658ms	227	8,315ms	128,633ms
Test 9	317	10,359ms	122,418ms	160	6,881ms	139,605ms
Test 10	356	9,896ms	116,857ms	179	7,494ms	135,794ms

respectively. As long as the overall response time is shortened, this reduction can provide better condition for the real-time assurance of ACPS.

(3) As expected, HR obtains more slack time than RE in all the tests. The average slack time for RE, is about 115,000ms, whereas that for HR is increased to 135,000ms. That is, about 20,000ms slack time are increased during the runtime phase.

In overall, this experiment confirms that the proposed human-interaction-aware adaptive functional safety processing is quite useful during the runtime phase. HR can reduce overall task redundancy of safety-critical automotive functions while meeting their functional safety requirements, shorten the overall response time of safety-critical automotive functions, and increase the slack time for non-safety-critical automotive functions.

6 CONCLUSION

As the functional safety assurance involves human-interaction between the driver and ACPS at runtime, an adaptive functional safety processing, including refined exploration during the design phase and human-interaction-aware task remapping during the runtime phase, was proposed in this article. The proposed refined exploration method shows good overall response time reduction of safety-critical functions. The human-interaction-aware task remapping shows significant overall response time reduction, overall task redundancy reduction, and slack time increase. By combing these two phases, the proposed adaptive functional safety processing can autonomously respond to the change of the driver's controllability in a fast manner at runtime. Our future work will study the human-interaction-aware driving automation for autonomous vehicles by considering the combination of functional safety and cyber security.

REFERENCES

[1] ISO 26262: Road vehicles-functional safety (Dec. 2018). Retrieved from https://www.iso.org/standard/68383.html.

[2] ISO 26262: Road vehicles-functional safety (Nov. 2011). Retrieved from https://www.iso.org/standard/43464.html.

[3] Theodore P. Baker. 2005. An analysis of EDF schedulability on a multiprocessor. *IEEE Trans. Parallel Distrib. Syst.* 16, 8 (2005), 760–768.

[4] Guillem Bernat, Antoine Colin, and Stefan M. Petters. 2002. WCET analysis of probabilistic hard real-time systems. In *Proceedings of the 23rd IEEE Real-Time Systems Symposium*. IEEE, 279–288.

[5] Martin Burns, Joe Manganelli, David Wollman, Ronald Laurids Boring, Stephen Gilbert, Edward Griffor, Yi-Ching Lee, Dan Nathan-Roberts, and Tonya Smith-Jackson. 2018. Elaborating the human aspect of the NIST framework for cyber-physical systems. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, Vol. 62. SAGE Publications, Los Angeles, CA, 450–454.

[6] Wanli Chang, Samarjit Chakraborty, et al. 2016. Resource-aware automotive control systems design: A cyber-physical systems approach. *Found. Trends Electr. Des. Automat.* 10, 4 (Dec. 2016), 249–369.

[7] Wanli Chang, Dip Goswami, Samarjit Chakraborty, and Arne Hamann. Jul. 2018. OS-aware automotive controller design using non-uniform sampling. *ACM Trans. Cyber-Phys. Syst.* 2, 4 (Jul. 2018), 26.

[8] Simon Fürst and AUTOSAR Spokesperson. 2015. Autosar the next generation–the adaptive platform. In *CARS@ EDCC2015* (2015).

[9] Simon Fürst and AUTOSAR Spokesperson. 2016. AUTOSAR adaptive platform for connected and autonomous vehicles. In *Proceedings of the 8th Vector Congress Conference.*

[10] ggtce. 2015. Task graph generator. Retrieved from https://sourceforge.net/projects/taskgraphgen/.

[11] Debkalpa Goswami, Reinhard Schneider, Alejandro Masrur, Martin Lukasiewycz, Shiladri Chakraborty, Harald Voit, and Anuradha Annaswamy. 2012. Challenges in automotive cyber-physical systems design. In *Proceedings of the 2012 International Conference on Embedded Computer Systems (SAMOS'12).* IEEE, 346–354.

[12] Chris Hobbs and Patrick Lee. Jul. 2013. Understanding ISO 26262 ASILs. Retrieved from https://www.electronicdesign.com/embedded/understanding-iso-26262-asils.

[13] Masao Ito. 2015. Controllability in ISO 26262 and driver model. In *Proceedings of the European Conference on Software Process Improvement.* Springer, 313–321.

[14] Maki Kawakoshi, Takashi Kobayashi, and Makoto Hasegawa. 2015. *ISO 26262: Controllability Evaluation Technique by Expert Riders.* Technical Report. SAE Technical Paper.

[15] Pratyush Kumar, Dip Goswami, Samarjit Chakraborty, Anuradha Annaswamy, Kai Lampka, and Lothar Thiele. 2012. A hybrid approach to cyber-physical systems verification. In *Proceedings of the 2012 49th ACM/EDAC/IEEE Design Automation Conference (DAC'12).* 688–696.

[16] Andrew L. Kun et al. 2018. Human-machine interaction for vehicles: Review and outlook. *Found. Trends Hum.–Comput. Interact.* 11, 4 (2018), 201–293.

[17] Yue Ma, Junlong Zhou, Thidapat Chantem, Robert P. Dick, Shige Wang, and X. Sharon Hu. 2018. On-line resource management for improving reliability of real-time systems on big–little type MPSoCs. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* (2018). DOI: https://doi.org/10.1109/TCAD.2018.2883990

[18] Arslan Munir. Apr. 2017. Safety assessment and design of dependable cybercars: For today and the future.*IEEE Consum. Electr. Mag.* 6, 2 (Apr. 2017), 69–77.

[19] M. Di Natale and A. L. Sangiovanni-Vincentelli. Mar. 2010. Moving from federated to integrated architectures in automotive: The role of standards, methods and tools. *Proc. IEEE* 98, 4 (Mar. 2010), 603–620.

[20] Jonas Nilsson, Anders C. E. Ödblom, and Jonas Fredriksson. 2016. Worst-case analysis of automotive collision avoidance systems. *IEEE Trans. Vehic. Technol.* 65, 4 (2016), 1899–1911.

[21] Roman Obermaisser, Christian El Salloum, Bernhard Huber, and Hermann Kopetz. Jul. 2009. From a federated to an integrated automotive architecture. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 28, 7 (Jul. 2009), 956–965.

[22] Ingrid Pettersson and Wendy Ju. 2017. Design techniques for exploring automotive interaction in the drive towards automation. In *Proceedings of the 2017 Conference on Designing Interactive Systems.* ACM, 147–160.

[23] Bobbie D. Seppelt and Trent W. Victor. 2016. Potential solutions to human factors challenges in road vehicle automation. In *Road Vehicle Automation 3.* Springer, 131–148.

[24] Georgios L. Stavrinides and Helen D. Karatza. 2012. Scheduling real-time DAGs in heterogeneous clusters by combining imprecise computations and bin packing techniques for the exploitation of schedule holes. *Fut. Gener. Comput. Syste.* 28, 7 (2012), 977–988.

[25] Haluk Topcuoglu, Salim Hariri, and Min-you Wu. Mar. 2002. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Trans. Parallel Distrib. Syst.* 13, 3 (Mar. 2002), 260–274.

[26] Guy H. Walker and Neville A. Stanton. 2017. *Human Factors in Automotive Engineering and Technology.* CRC Press, Boca Raton, FL.

[27] Tongquan Wei, Junlong Zhou, Kun Cao, Peijin Cong, Mingsong Chen, Gongxuan Zhang, Xiaobo Sharon Hu, and Jianming Yan. 2018. Cost-constrained QoS optimization for approximate computation real-time tasks in heterogeneous MPSoCs. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 37, 9 (2018), 1733–1746.

[28] Tingming Wu, Haifeng Gu, Junlong Zhou, Tongquan Wei, Xiao Liu, and Mingsong Chen. 2018. Soft error-aware energy-efficient task scheduling for workflow applications in DVFS-enabled cloud. *J. Syst. Arch.* 84 (2018), 12–27.

[29] Guoqi Xie, Yuekun Chen, Yan Liu, Yehua Wei, Renfa Li, and Keqin Li. 2017. Resource consumption cost minimization of reliable parallel applications on heterogeneous embedded systems. *IEEE Trans. Industr. Inf.* 13, 4 (Aug. 2017), 1629–1640.

[30] Guoqi Xie, Zhetao Li, Na Yuan, Renfa Li, and Keqin Li. 2018. Toward effective reliability requirement assurance for automotive functional safety. *ACM Trans. Des. Autom. Electr. Syst.* 23, 5 (Aug. 2018), 65. DOI:https://doi.org/10.1145/3230620

[31] Guoqi Xie, Hao Peng, Zhetao Li, Jinlin Song, Yong Xie, Renfa Li, and Keqin Li. 2018. Reliability enhancement towards functional safety goal assurance in energy-aware automotive cyber-physical systems. *IEEE Trans. Industr. Inf.* 14, 12 (Dec. 2018), 5447–5462. DOI:https://doi.org/10.1109/TII.2018.2854762

[32] Guoqi Xie, Gang Zeng, Jiyao An, Renfa Li, and Keqin Li. 2018. Resource cost-aware fault-tolerant design methodology for end-to-end functional safety computation on automotive cyber-physical systems. *ACM Trans. Cyber-Phys. Syst.* 3, 1 (Aug. 2018), 4. DOI:https://doi.org/10.1145/3162052

[33] Guoqi Xie, Gang Zeng, Junqiang Jiang, Chunnian Fan, Renfa Li, and Keqin Li. 2017. Energy management for multiple real-time workflows on cyber–physical cloud systems. *Fut. Gener. Comput. Syst.* (May 2017). DOI:https://doi.org/10.1016/j.future.2017.05.033

[34] Guoqi Xie, Gang Zeng, Zhetao Li, Renfa Li, and Keqin Li. 2017. Adaptive dynamic scheduling on multi-functional mixed-criticality automotive cyber-physical systems. *IEEE Trans. Vehic. Technol.* 66, 8 (Aug. 2017), 6676–6692.

[35] Guoqi Xie, Gang Zeng, Yan Liu, Jia Zhou, Renfa Li, and Keqin Li. 2018. Fast functional safety verification for distributed automotive applications during early design phase. *IEEE Trans. Industr. Electron.* 65, 5 (May 2018), 4378–4391.

[36] Junlong Zhou, Jin Sun, Xiumin Zhou, Tongquan Wei, Mingsong Chen, Shiyan Hu, and Xiaobo Sharon Hu. 2018. Resource management for improving soft-error and lifetime reliability of real-time MPSoCs. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* (2018). DOI:https://doi.org/10.1109/TCAD.2018.2883993

[37] Junlong Zhou and Tongquan Wei. 2015. Stochastic thermal-aware real-time task scheduling with considerations of soft errors. *J. Syst. Softw.* 102 (2015), 123–133.

[38] Junlong Zhou, Jianming Yan, Tongquan Wei, Mingsong Chen, and Xiaobo Sharon Hu. 2017. Energy-adaptive scheduling of imprecise computation tasks for QoS optimization in real-time MPSoC systems. In *Proceedings of the Conference on Design, Automation and Test in Europe.* European Design and Automation Association, 1406–1411.

[39] Xiumin Zhou, Gongxuan Zhang, Jin Sun, Junlong Zhou, Tongquan Wei, and Shiyan Hu. 2019. Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based HEFT. *Fut. Gener. Comput. Syst.* 93 (2019), 278–289.

Received December 2018; revised May 2019; accepted May 2019