IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 2 December 2023, accepted 5 January 2024, date of publication 30 January 2024, date of current version 7 February 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3360275*

RESEARCH ARTICLE

# Assessing Critical Adaptations in Automated Adaptive Software Systems by Stage Decomposition

**SHUJI MORISAKI[1], (Member, IEEE), MICHIYO WAKIMOTO[1], AND NORIMITSU KASAI[2]**

[1]Graduate School of Informatics, Nagoya University, Nagoya, Aichi 464-8601, Japan
[2]Communication Systems Center, Information Security Management and Investigation Department, Security Section, Mitsubishi Electric Corporation, Hyogo 661-0981, Japan

Corresponding author: Shuji Morisaki (s.morisaki.jp@ieee.org)

**ABSTRACT** In environments involving a variety of connected devices and systems, there is an ever-increasing demand for automated adaptation. To ensure that all threats are identified and manageable in such environments, quality assurance activities including testing and inspections in design-time should focus on assessing the reliability of critical adaptations, which may threaten life, economic property, or important information. This work proposes an approach for identifying and evaluating critical adaptations on the basis of their automation level, reliability, detectability, and recoverability by decomposing adaptations into four stages: monitor, analyze, plan, and execute. This work also empirically evaluates the effectiveness of the proposed approach by assessing a real safety-critical telecommunication system with critical adaptation features and comparing the results with the STAMP (System Theoretic Accident Model and Processes)/STPA (System-Theoretic Process Analysis) approach. The results of the evaluation indicated that the proposed approach could assess critical adaptation features provided by the system with reasonable effort. Additionally, structured views provided by the proposed approach enable efficient quality assurance activities. In the evaluation, the proposed approach achieves similar results to the STAMP/STPA approach but requires 33% less effort.

**INDEX TERMS** Automated adaptation, inspection, verification and validation, safety.

## I. INTRODUCTION

The demand for automated, self-adaptive systems has increased in heterogeneous environments, where a variety of devices and systems are connected to the same network. A lot of research contributes to self-adaptive systems that have capabilities of automated adaptation at runtime, as defined at design-time [1], [2], [3], [4], [5], [6]. Many of such research evaluated their frameworks, approaches, and techniques with architecture-based adaptations of distribution servers such as web servers and content delivery servers. The demand for adaptation in critical systems also increases. Critical adaptations are adaptations that have high impacts in areas that are safety- or mission-critical. A dynamic configuration in an automotive control software system is an example of critical adaptations because it has a high impact on human safety in the vehicle. Pairing between a Bluetooth compliant portable music player and an onboard speaker on the vehicle is an example of non-critical adaptations.

Automated adaptations in critical systems require more discussions and empirical evaluations [7]. Since uncertainty in critical adaptations may threaten assets (e.g., human life, economic property, or important information), adaptations in critical systems require more empirical evaluations. While some previous research pointed out that the human-assisted adaptations and human in the loop may be required or more appropriate in critical systems [8], [9], other research referred to the risk of the uncertainties of human-assisted adaptation and human in the loop [10], [11], [12]. Some research demonstrated hybrid approaches for automated and human-assisted adaptations [8], [13], [14]; however, they do not explicitly assess the impact of adaptations on the assets at

The associate editor coordinating the review of this manuscript and approving it for publication was Fabrizio Messina.

design-time to reduce the uncertainty at runtime. Without a proper assessment at design-time, critical adaptations with unacceptable uncertainty may occur at runtime.

Common safety assessment approaches at design-time for critical systems such as STAMP (System Theoretic Accident Model and Processes)/STPA (System-Theoretic Process Analysis) [15] and FTA (Fault Tree Analysis) do not provide specific procedures to evaluate the impact of critical adaptations on the assets. Specific procedures such as decomposing the adaptation phases may verify the acceptability and reliability of critical adaptations.

Similarly, common automation evaluation approaches such as MAPE-K framework [16], Parasuraman's model [17], and adaptation design considering human in the loop [9] do not explicitly specify a procedure to assess the impact of critical adaptations on the assets. Although some studies have proposed assessment methods for risks of critical adaptations [18], [19], the risks do not include threats to the assets. Specifically, Mostafa et al. defined risk as mission failure that does not include the threats to the external assets [18]. Roehr and Shi defined the highest risk in a reconfigurable integrated multi-robot exploration adaptive system as the loss of the robot [19]. The loss of the robot does not include the threats to the external assets. Neither of these studies refers to the impact of the adaptation on the assets as risk.

This work proposes a threat assessment approach for critical adaptations of the system at design-time. The approach assesses the impact of critical adaptations on the assets to validate whether they negatively impact assets such as injury or damage to human life, damage to economic property, or important information breaches. This work empirically evaluated using a real safety-critical telecommunication system developed and operated in industry. The proposed approach is refined from our paper [20] after applying the previously proposed approach to several adaptive software systems. The contributions of this work are the following.

- Refinement of the procedure of the previously proposed approach
  We applied the previously proposed critical adaptation detection approach [20] to several adaptive software systems and discussed the results with practitioners. The results indicated that moving the last step of the procedure for the approach defined in [20] to the first step enables to eliminate of non-critical adaptations from target adaptations. This leads to effort reduction for assessment.
- An empirical evaluation of the refined approach in a real telecommunication system in industry
  Moving the step of identification of assets to the beginning of the procedure can reduce adaptations to be assessed and lead to effort reduction because selecting assets to be protected can reduce adaptations to be assessed.
- Comparison of the assessment coverage with a general hazard analysis method STAMP/SPTA (Systems-Theoretic Accident Model and Processes/System-Theoretic Process Analysis)

The rest of this paper is organized as follows. Section II introduces related research. In Section III, we propose an approach to assess critical adaptations by decomposing adaptations into four stages. Section IV describes the case study. Section V discusses the results, and Section VI concludes the study.

## II. RELATED RESEARCH

### A. ADAPTATION AND AUTOMATION

Many studies have investigated adjustable autonomy [21], which allows an adaptive system to change its automation level at runtime. This feature enhances the flexibility because human decision-making may be more appropriate than automated decision-making in some situations [21], [22]. Dorais et al. noted that adaptive systems should have specific criteria and circumstances to override manual control (decision-making transfer) [21], but they did not identify the procedure, criteria, or circumstances. If the decision-making transfer results in a safety-critical impact on the assets such as injury to human life or damage to economic property, the feasibility of such a transfer must be assessed at design-time and limited at runtime, if necessary.

Although previous studies referred to decision-making transfers in adjustable autonomy [23], [24], [25], [26], [27], they do not explicitly assess the impact of safety-critical adaptations on the assets. Some studies considered the risk of adjustable autonomy [18], [19], [26], [28], [29], [30], [31]. Bush et al. defined risk as the degree that the mission goal cannot be achieved [28]. Durand et al. proposed a formal modeling method to detect inconsistency of control architecture, however, the method does not consider the critical impact of the inconsistency on the assets [29]. Roehr and Shi defined the highest risk in a reconfigurable integrated multi-robot exploration adaptive system as the loss of the robot. They also pointed out that an efficient execution of the overall mission was essential [19]. Johnson et al. compared the autonomy level and efficiency of robots and human teams by measuring the number of hours and work failures (errors) [30]. None of these studies explicitly considered the critical impact on the assets.

Mostafa et al. defined risk as mission failure and not as the critical impact on the assets [18]. Furthermore, they defined eleven criteria for viability assessments, but did not explicitly include detectability and recoverability. Although Zieba et al. proposed the concept of barriers to prevent the unsafe behavior of humans, they did not evaluate barriers to prevent the unsafe behavior of automated decision-making [31]. Although another study [32] found that unreliability reduces human trust, the unreliability did not explicitly consider the critical impact of the assets.

Several approaches and frameworks that evaluate trade-offs between architecture-based adaptations and their qualities are proposed [14], [33], [34]. In the article [33],

Perez-Palacin et al. proposed a self-adaptation approach for analyzing tradeoffs between system adaptability and its quality of service. Their approach defines adaptation, quality attributes, and tradeoff points between them at design-time and monitors the conditions of the tradeoff points at runtime. When the conditions are satisfied, the adaptations are executed. Also, they evaluated that the scalability of the proposed approach and the performance required by the monitoring is feasible. In the article [34], Zoghi et al. proposed a search-based adaptive approach that defined control points, adaptations, and their priorities at design-time and activated the adaptations using a search-based algorithm at runtime. Designers define the priorities at design-time according to the results of the Analytic Hierarchy Process. In the article [14], Yang et al. proposed an approach for identifying quality attributes and analyzing their tradeoffs for the target adaptive system at design-time. Designers analyze the tradeoffs by using usage scenarios. These approaches and our approach analyze tradeoffs between adaptation (automation level) and quality attributes. While their approach assumes that adaptations are architectural adaptations for web systems such as the number of servers and does not explicitly refer to assessments of critical adaptations, our approach assesses critical adaptations not limited to such architectural adaptations.

Rainbow is a framework for architecture-based self-adaptation [1], [2]. In the Rainbow framework, designers define invariant rules and adaptations strategy rules at design-time, and the adaptation programs change architecture along with the defined rules. In the article [1], a web system implemented by the framework was evaluated. The article [13] evaluated the Rainbow framework in software evolutions with a case study that developed an adaptive system that evolved from a non-adaptive system. In the case study, the effort for non-adaptive system analysis and implementation for adaptive features by reusing components of the Rainbow framework is feasible. The framework does not explicitly refer to critical adaptations and their assessments.

Some research referred to human-assisted adaptations and human in the loop. Salehie and Tahvildari introduced the positive and negative sides of human-assisted adaptation research [35]. Mirial et al. pointed out that experience in autonomous systems shows that people cannot be excluded entirely from the adaptation loop, such as autonomous cars [9]. Weyns et al. referred to human in the loop and uncertainty from human-assisted adaptations [36]. Camara et al. proposed an approach for managing the uncertainty of human-assisted adaptation by applying the OWC (Opportunity willingness capability) human model in the security domain. These studies did not explicitly refer to critical adaptations nor refer to a detailed procedure for tradeoff analysis on human-assisted adaptations.

Many approaches and frameworks enable various adaptation strategies. Elkhodary et al. proposed a framework that enabled self-adaptive systems to tune their parameters by machine learning [37]. Nakagawa et al. proposed a configuration compiler that generated architecture design by specified goal-oriented requirements and design constraints [38]. Dorn et al. proposed an approach that defined not only adaptations but also user models for more accurate adaptation. Designers define user architecture, system architecture, and their mapping using mapping templates [39]. Luckey and Engels proposed an approach for separating adaptation and business logic. The adaptation logic, including quality concerns, is defined at design-time and written in ACML (Adapt Case Modeling Language) [40]. These research does not distinguish critical adaptations.

### B. SAFETY ANALYSIS

Leveson proposes the STAMP/STPA approach [15], [41]. The approach is a safety analysis technique that identifies hazard causality factors and hazardous scenarios by extracting accidents, hazards, safety constraints, and control structures of a system. Defining hazard causality factors as the execution results of critical adaptation features enables to assess potential hazards caused by the critical adaptations. The results of the assessment can be used for ensuring that criticalities of critical adaptation features are acceptable. However, the approach does not provide specific guides to identify hazardous scenarios such as stage decomposition by the proposed approach.

Failure mode and effect analysis (FMEA) [42] identifies possible failure modes of components in the system and analyzes the resulting effects of the failure modes to the rest of the system. Our approach first identifies assets to be protected and adaptations that may affect the assets, while FMEA identifies possible failure modes of all components. Furthermore, our approach does not require identifying failure modes of components.

Fault tree analysis (FTA) [43] identifies undesirable events for the system and users and decomposes the undesirable events into sub-events by using Boolean logic. The decomposition generates a fault tree with an undesirable event as the root node and decomposed basic events as leaf nodes. Reducing the risks of the basic events leads to a reduction in the risks of the undesirable event. Our approach first identifies assets to be protected to focus on hazards of the assets, while FTA does not explicitly specify the step. FTA requires decomposing the undesirable events into sub-events and basic events, while our approach does not require decomposition.

Functional hazard analysis (FHA) [44] enumerates components of the system, identifies hazards for each of the components, and refines the components and the architecture to reduce the risks of the hazards. Although FHA and our approach are common in identifying hazards for components, our approach first enumerates assets to be protected and extracts adaptations that affect the assets. Our approach analyzes tradeoffs between adaptations and automation levels while FHA does not focus on automation level.

Hazard and operability study (HAZOP) [45] identifies potential deviations that threaten assets to be protected by guide words such as more, less, and none. HAZOP reviews

```
main(){
  automation_level = determining_automation_level(critical_adaption)
  quality_attributes = determining_quality_attributes(critical_adaption)
  criticality = determining_acceptable_adaptation(automation_level, quality_attributes)
}
#--------------------------------
function determining_automation_level(critical_adaption){
  if (critical_adaptation.is_fully_implemented_by_external_systems_or_humans()){
    return “level 1”
  }
  predefined_processess = critical_adaptation.get_all_predefined_processes()
  for each predefined_process in predefined_processess{
    if (predefined_process.is_executed_by_external_systems_or_humans()){
        return “level 2”
    }
  }
  if not (critical_adaptation.is_fully_automated()){
    return “level 3”
  }
  if (critical_adaptation.is_intervened_by_ external_systems_or_humans()){
    return “level 4”
  }
  return “level 5”
}
#--------------------------------
function determining_quality_attributes(critical_adaption){
  if (critical_adaption.always_possible_under_anycondition()){
     return  “level 5”
  }
  if (critical_adaption.always_possible_excluding_rare_cases()){
       return  “level 4”
     }
  }
  if (critical_adaptation.always_possible_excluding_exceptional_or_error_conditions()){
      return  “level 3”
  }
  if not (critical_adaptation.ready_possible_in_normal_conditions()){
      return  “level 2”
  }
  return  “level 1”
}
#--------------------------------
function determining_acceptable_adaptation(automation_level, quality_attributes){
  while (criticality == “acceptable”) {
  if (quality_attributes.satisfy_reliability()){
     criticality = “acceptable”
  } else if (quality_attributes.satisfy_recoverability()){
    if (quality_attributes.satisfy_detectability()){
      criticality = “acceptable”
    }
    re_evaluate(automation_level, quality_attributes)
  }
  return criticality
}
```

**FIGURE 1.** Pseudocode of the proposed approach.

the design and engineering issues by decomposing the entire system or process into components or nodes. Our approach also analyzes critical adaptations by decomposing them into phases and identifying potential deviations. HAZOP does not limit the target to an automated (adaptation) process nor specify criteria to determine whether the identified hazards are acceptable.

### C. SOFTWARE REVIEWS

Architecture evaluation methods including architecture tradeoff analysis method (ATAM) [46], [47] identify important quality attributes including performance, availability, and security for system architecture. Then, it identifies architecture tradeoffs that may affect the identified quality attributes. Finally, it verifies that the architectural decisions affect the achievement of quality attributes. The quality attributes can include the reliability of critical adaptations, safety, and security to ensure that the criticalities of critical adaptations are acceptable. However, ATAM does not provide the decomposition and assessment procedures that the proposed approach provides.

Value-based review [48] provides guides to prioritize defect types in software reviews. In a VBR, inspectors assign a higher detection priority to defects categorized as defect types that have the potential to spoil the higher value capabilities of the target software. The stakeholders of the software

determine the order of priority using a checklist to identify the higher value capabilities. VBR can assess the criticalities of critical adaptations by prioritizing defects that may affect the reliabilities of critical adaptations, safety, and security. However, VBR does not provide the decomposition and assessment procedures that the proposed approach provides.

Common reading techniques for software inspections can assess critical adaptations. In defect-based reading [49], [50], first, the inspection organizer defines defect detection scenarios to detect critical adaptations and assigns the scenarios to inspectors. Next, the inspectors try to detect defects along with the assigned scenarios. Finally, the inspection organizer assesses that the criticalities of critical adaptations are acceptable. In perspective-based reading [51] [52], the inspection organizer defines perspectives to detect critical adaptations and assigns the perspectives to inspectors. Next, the inspectors try to detect defects. Finally, the inspection organizer assesses the criticalities by the result of detected defects. In usage-based reading [53], [54], by defining use cases as critical adaptations of the target adaptive software system, assessment can be performed with the inspection results. These reading techniques do not refer the stage decomposition and assessment procedures that the proposed approach provides.

## III. PROPOSED APPROACH

### A. OVERVIEW

The proposed approach identifies critical adaptations in system $X$ and assesses whether the criticalities of the adaptations are acceptable. Adaptation addresses software systems that must change their behavior during execution in response to environmental changes [55]. Critical adaptations are adaptations that have high impacts in areas that are safety- or mission-critical. A dynamic configuration in an automotive control software system is an example of critical adaptations because it has a high impact on human safety in the vehicle. Pairing between a Bluetooth compliant portable music player and an onboard speaker on the vehicle is an example of non-critical adaptations. Facial recognition in an image is an example of automatic adaptations. When facial recognition is used in a smartphone image synthesis app to add animal ears to the recognized face is non-critical adaptation because the recognition does not have to be of high fidelity. However, if facial recognition is used for authorization to enter the server room for a banking system, it will be a critical adaptation because the adaptation may threaten the property.

First, an assessor identifies hazards $H$ that the critical adaptations may cause along with the system lifecycle $L$ and then assesses whether the probabilities of exposure of the hazards are acceptable. Then, the proposed approach decomposes critical adaptations into monitor, analyze, plan, and execute stages defined by MAPE-K framework [16]. Also, the definitions of automation levels are based on the Parasuraman's model [17]. Table 9 (in Appendix A) shows a detailed description of the decomposition of critical adaptations into four stages by the automation level in the proposed approach: monitor, analyze, plan, and execute. Fig. 1 shows the pseudocode of the proposed approach. The assessor evaluates the levels of automation, reliability, detectability, and recoverability for each decomposed stage and ensures that the criticalities of the adaptation are acceptable.

### B. ASSETS AND LIFECYCLE

Assets $S$, including human life, economic property, and important information, are supposed to be protected by system $X$. Lifecycle $L$ is the lifecycle process of system $X$. As defined by ISO [56], [57], the lifecycle includes the system launch, the start and end of the service, system maintenance, system updates and evolutions, and system disposal. Critical adaptations (critical adaptation executions) $A_e$ are identified along with lifecycle $L$. Although the proposed approach does not assume a specific lifecycle, this work considers the general lifecycle, which includes initialization, startup, monitoring, shutdown, reconfiguration, update/evolution, termination, and disposal. This lifecycle should be replaced or adjusted to suit the target system. It can also be nested, as defined by the guide in [58]. For example, a UNIX background process (daemon) is initialized and started during the UNIX system initialization (boot) process.

### C. HAZARDS

The proposed approach assumes that sufficient completeness of the list of hazards or hazard identification procedures is provided. If standards or laws provide shared hazards or hazards identification procedures in the domain, the assessor selects applicable hazards in the critical adaptations. In case of absence of standards or laws, the assessor selects applicable hazards from in-house hazard repository, experts' knowledge, organization's standard quality assurance procedure, or results of root cause analyses in the past development. The proposed approach also assumes that the assessor has expertise on hazards and their causality in critical adaptations as well as other criticality assessment approach requires assessors with such expertise [59], [60].

### D. ADAPTATION STAGES

A set of critical adaptations $A$ consists of adaptations that may threaten assets $S$. Each critical adaptation $A_i$ ($\in A$) is decomposed into four stages as defined by MAPE-K framework:

(a) Monitor stage $A_{mi}$

System $X$ monitors and acquires information for critical adaptation $A_i$.

(b) Analyze stage $A_{ai}$

System $X$ analyzes the monitored information in the monitor stage $A_{mi}$ for critical adaptation $A_i$.

(c) Plan stage $A_{pi}$

System $X$ selects a plan and action by the analysis result in the analyze stage $A_{ai}$ for critical adaptation $A_i$.

(d) Execute stage $A_{ei}$

System $X$ carries out a plan that is selected in the plan stage $A_{pi}$ for critical adaptation $A_i$.

Each critical adaptation is denoted as a four-tuple $A_i = (A_{mi}, A_{ai}, A_{pi}, A_{ei})$. The selected plan in plan stage $A_{pi}$ is executed in execute stage $A_{ei}$. When plan stage $A_{pi}$ uses two or more results of analyze stage $A_{ai}$, each analyze stage is denoted as an $n$-tuple $A_{ai} = (A_{ai.1}, A_{ai.2}, A_{ai.3}, \dots)$. Similarly, two or more monitor stages are denoted as $A_{mi} = (A_{mi.1}, A_{mi.2}, A_{mi.3}, \dots)$. In cases where plan stage $A_{pi}$ corresponds to two analyze stages $A_{ai.1}$ and $A_{ai.2}$ and each analyze stage $A_{ai}$ corresponds to two monitor stage $A_{mi}$, the critical adaptation is denoted as $A_i = (((A_{mi.1.1}, A_{mi.1.2}), (A_{mi.2.1}, A_{mi.2.2})), (A_{ai.1}, A_{ai.2}), A_{pi}, A_{ei})$.

First, the proposed approach identifies critical adaptation execution $A_{ei}$. Then corresponding stages $A_{pi}$, $A_{ai}$, and $A_{mi}$ are extracted in this order due to the dependencies among $A_{ei}$, $A_{pi}$, $A_{ai}$, and $A_{mi}$.

Parasuraman proposed the Parasuraman's model [17], which consists of acquisition, analysis, decision, and action stages. Stages in the proposed approach are similar to their framework. Phases defined in the MAPE-K framework, which the proposed approach uses, and phases defined in the Parasuraman's model are mutually exchangeable. Boyd proposed the OODA loop as a military strategy [61], [62]. Then, widely used in various areas, including business strategies. OODA consists of observation, orientation, decision, and action processes [62]. Phases in the proposed approach are similar to the OODA loop. However, in the OODA loop, the observed process is not required to be a predefined way.

### E. AUTOMATION LEVEL AND ITS QUALITY ATTRIBUTES

The criticalities can be assessed by determining automation level $e_l$ and estimating its quality attributes consisting of reliability $e_r$, detectability $e_d$, and recoverability $e_c$ for each stage of critical adaptation $A_i$. Table 1 shows five possible automation levels: 1 (fully manual), 2 (partially automated), 3 (semi-automated), 4 (automated with human override), and 5 (fully automated). Fig. 2 shows a flowchart for determining the automation level, which corresponds to the 'determining_automation_level()' function in Fig. 1. Note that the original Parasuraman's model categorized 10-level model [17]. For the proposed approach, we reduced 5-level model because 10-level is redundant. As shown in Table 1, the proposed approach categorizes the ten levels in Parasuraman's model into five levels. Levels 1, 2, 3, 4, and 5 in the proposed approach correspond to levels 1, 2–4, 5, 6–9, and 10, respectively.

Table 2 summarizes the quality attributes. Reliability $e_r$ is estimated from the accuracy of the results, using a five-point scale from 1 (lowest) to 5 (highest). Detectability $e_d$ is estimated from the probability of finding an incorrect result using a five-point scale from 1 (lowest) to 5 (highest). Recoverability $e_c$ is estimated from the probability of correcting an incorrect result on a five-point scale from 1 (lowest) to 5 (highest). Levels of three quality attributes $e_r$, $e_d$, and $e_c$ can be categorized as coverage of nominal (goal fulfillment) behavior group (levels 1, 2, and 3) and coverage of exceptional (exception and error handling) behavior group (levels 4 and 5). The nominal behavior group has no exception and error handling. Level 1 indicates that correct results for nominal behavior are rarely obtained. Level 2 indicates that correct results for nominal behavior are obtained excluding some cases. Level 3 indicates that correct results for nominal behavior are always obtained. Level 4 indicates that correct results for nominal behavior are always obtained and that correct results for exceptional behavior are obtained excluding rare cases. Level 5 indicates that correct results for nominal and exceptional behavior are always obtained. Fig. 3 shows a flowchart for determining the quality attributes, which corresponds to the 'determining_quality_attributes()' function in Fig. 1.

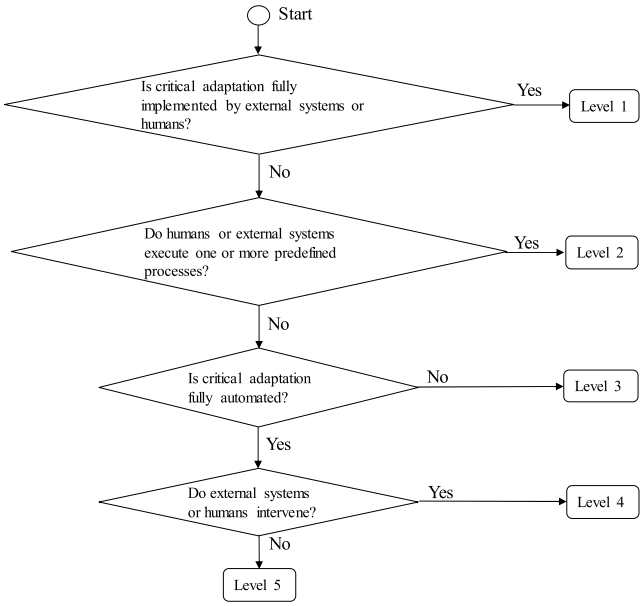


**FIGURE 2.** Flowchart of determining the automation level.

**TABLE 1.** Automation level $e_l$.

| Level | Description |
|---|---|
| 5 | Fully automated. No intervention by humans or external systems allowed. |
| 4 | Automated with human override. System does not require assistance, but humans or external systems may intervene. |
| 3 | Semi-automated. System may request assistance, if required. |
| 2 | Partially automated. Humans or external systems are required to conduct one or more processes in a predefined way. |
| 1 | Fully manual. |

The proposed approach determines the automation level and estimates reliability, detectability, and recoverability for each stage in critical adaptation $A_i$ ($\in A$) as follows:

#### 1) MONITOR $A_m$

The input data are monitored. At automation level 1, the data is provided from (outside of system $X$) such as a manual

**TABLE 2.** Reliability $e_r$, detectability $e_d$, and recoverability $e_c$.

| Level | Reliability $e_r$ | Detectability $e_d$ | Recoverability $e_c$ |
|---|---|---|---|
| 5 | Correct result always obtained including rare cases in both nominal and exceptional behaviors | Always detectable including rare cases in both nominal and exceptional behaviors | Always recoverable including rare cases in both nominal and exceptional behaviors |
| 4 | Correct result always obtained in nominal behavior and in exceptional behavior excluding rare cases | Always detectable in nominal behavior and in exceptional behavior excluding rare cases | Always recoverable in nominal behavior and in exceptional behavior excluding rare cases |
| 3 | Correct result always obtained in nominal behavior, but no correct result obtained in exceptional behavior | Always detectable in nominal behavior but not detectable in exceptional behavior | Always recoverable in nominal behavior but not recoverable in exceptional behavior |
| 2 | Correct result occasionally obtained in nominal behavior and no correct result obtained in exceptional behavior | Occasionally detectable in nominal behavior and not detectable in exceptional behavior | Occasionally recoverable in nominal behavior and not recoverable in exceptional behavior |
| 1 | Correct result rarely obtained in nominal behavior and no correct result obtained in exceptional behavior | Rarely detectable in nominal behavior and not detectable in exceptional behavior | Rarely recoverable in nominal behavior and not recoverable in exceptional behavior |

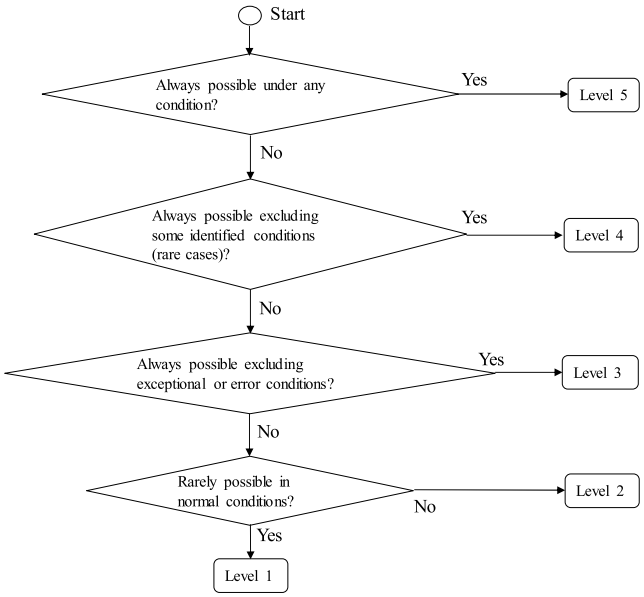


**FIGURE 3.** Flowchart of determining the quality attributes.

input or input from an external system. At automation level 5, data is obtained fully automatically from a subsystem or device consisting of system $X$ such as images captured by a camera device. Reliability $e_r$ is estimated from the accuracy of the input data. Detectability $e_d$ and recoverability $e_c$ are estimated from the probabilities of detecting and correcting incorrect input data.

#### 2) ANALYZE $A_a$

The monitored data is interpreted. At automation level 1, analysis is performed outside of system $X$ such as a user or an external system. At automation level 5, it is fully automated such as human facial recognition in the captured image. Reliability $e_r$ is estimated from the interpretation accuracy. Detectability $e_d$ and recoverability $e_c$ are estimated from the probabilities of detecting and correcting incorrect interpretation.

#### 3) PLAN $A_p$

A plan is selected depending on the interpretation. At automation level 1, the plan is made outside of system $X$ such as a user or external system. At automation level 5, it is fully automated such as an internal authentication for the recognized face. Reliability $e_r$ is estimated from the plan selection accuracy. Detectability $e_d$ and recoverability $e_c$ are estimated from the probabilities of detecting and correcting incorrect plan selection.

#### 4) EXECUTE $A_e$

A plan is carried out to execute the selected plan. At automation level 1, the execution is performed by a user or external system. At automation level 5, it is fully automated such as opening an automatic door for entrance control of a room via facial authentication. Reliability $e_r$ is estimated from the plan execution accuracy. Detectability $e_d$ and recoverability $e_c$ are estimated from the probabilities of detecting and correcting incorrect plan execution.

### F. CRITICALITY EVALUATION

Fig. 4 shows a flowchart of the criticality evaluation procedure, which corresponds to the 'determining_acceptable_adaptation()' function in Fig. 1. First, an assessor determines acceptable level $AL_i$ from 1 to 5 according to the required quality of the adaptation to protect asset $S_i$. The assessor can use several safety criteria as acceptable levels defined by safety standards, including IEC 61508 and the US government's Mil-Std882D [63], [64], [65]. Second, the assessor assesses that the reliability level $e_{ri}$ satisfies acceptable level $AL_i$. If the reliability level $e_{ri}$ satisfies the acceptable level $AL_i$, the criticality is acceptable. If the reliability level $e_{ri}$ does not satisfy the acceptable level $AL_i$, recoverability $e_{ci}$ and detectability $e_{di}$ levels are assessed. Recoverability and detectability levels are assessed because canceling, correcting, or redoing can cover insufficient reliability under some conditions such as loose constraints on realtimeness and resources. If the reliability $e_{ri}$, recoverability

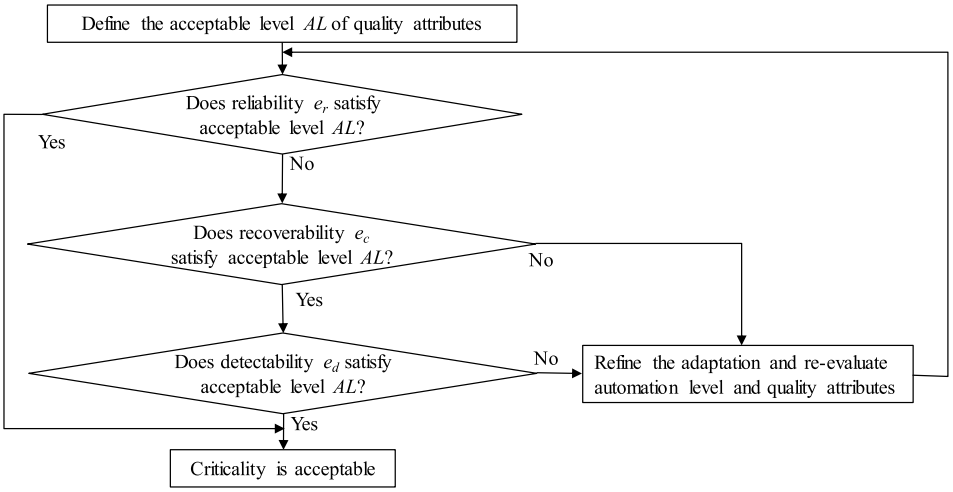


**FIGURE 4.** Flowchart for determining whether criticality is acceptable.

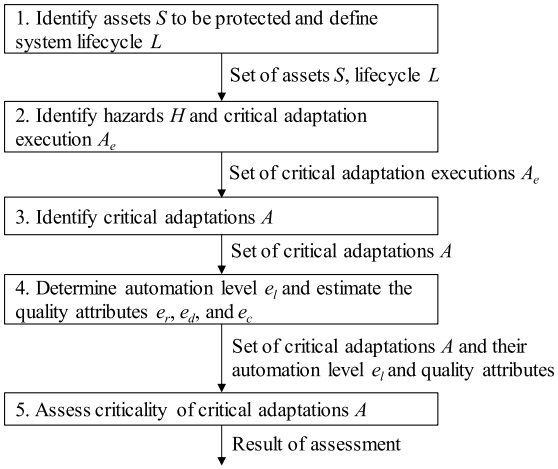


**FIGURE 5.** Procedure of the proposed approach.

$e_{ci}$, and detectability $e_{di}$ levels do not satisfy acceptable level $AL_i$, refine the adaptation and start re-assessment.

### G. PROCEDURE

Assessor evaluates critical adaptations with the procedure at design-time. The input is artifacts for system $X$ and hazard information. The artifacts for system $X$ include requirements and design documents. From the artifacts, the assessor extracts a set of assets to be protected $S$, system lifecycle $L$, and a set of critical adaptations $A$. The hazard information includes a hazards list shared in the domain of system $X$, in-house hazard repositories, the organization's standard procedure for quality assurance, root cause analysis results in past developments, and experts' knowledge. From the hazard information, the assessor extracts a set of hazards $H$. The guidelines of hazard lists are available in standards and laws (e.g., IEC 61508 [64]). In the case of experts' knowledge, the experts and assessors discuss potential hazards.

Fig. 5 overviews the proposed procedure. Below it is described in detail.

Step 1. Identify the assets to be protected and the system lifecycle

The assessor determines the set of assets $S = \{S_1, \ldots, S_m\}$ such as human life, economic property, or important information that system $X$ should protect and lifecycle $L$ of system $X$.

Step 2. Identify hazards and critical adaptation executions

The assessor determines the set of hazards $H = \{H_1, \ldots, H_n\}$ that have the potential to threaten assets $S$ obtained in step 1. III-H over system lifecycle $L$. If shared hazards in the domain of system $X$ or results of root cause analyses in the past system development in the domain are available, the assessor uses it. Additionally, the assessor identifies a set of critical adaptation executions $A_e = \{A_{e1}, \ldots, A_{en}\}$ that have the potential to cause hazards $H$. If traceability among artifacts of system $X$ is available, by using the traceability tool, the assessor can semi-automatically trace the corresponding specification, subsystem, algorithm, or program module that implements adaptation executions $A_e$ from requirements including assets $S_i$. If diagrams that provide the entire images of system $X$, such as an architectural diagram, entire timing chart, and sequence diagram are available, the assessor can trace the asset, hazard, and adaptation executions easily.

Step 3. Identify critical adaptations

The assessor determines a set of critical adaptations $A = \{A_1, \ldots, A_n\}$ and corresponding monitor $A_{mi}$, analysis $A_{ai}$, and plan $A_{pi}$ to critical adaptation execution $A_{ei}$ for each critical adaptation from set $A_e$ obtained in step 2 by tracing specification, sequence chart, timing chart, algorithms, or program modules. As mentioned in step 2 in the procedure, if artifact traceability is available, this procedure can be performed semi-automatically. Also, if the diagrams in step 2 are available, the assessor can trace them easily.

Step 4. Determine automation level $e_l$ and estimate the quality attributes

The assessor determines $e_l$ and estimates $e_r$, $e_d$, and $e_c$ for each stage in critical adaptation $A_i = (A_{mi}, A_{ai}, A_{pi}, A_{ei})$ obtained in step 3. Fig. 2 shows the procedure for determining

the automation level $e_l$. Fig. 3 shows the procedure for determining the quality attributes $e_r$, $e_d$, and $e_c$.

Step 5. Assess the criticality of critical adaptations $A$

The assessor evaluates the probability of cause hazard $H_i$ by critical adaptation $A_i$ using $e_l$, $e_r$, $e_d$, and $e_c$ for each stage ($e_l$, $e_r$, $e_d$, and $e_c$ are obtained in step 4 by the critical evaluation described in subsection III.III-F. If the evaluated probability is unacceptable, the corresponding critical adaptation is replaced with an alternative solution. Then, the alternative solution is evaluated and assessed.

### H. EXAMPLE ASSESSMENT

As an example, we assess an imaginary train speed limit enforcement system $T$ using the proposed approach. This system is assumed to be implemented as part of the train's onboard equipment and to receive radio signals from base stations on the train station platforms.

The requirements for system $T$, which are taken from the requirements document of the Institution of Railway Signal Engineers [66], are:

*Controls should be in place to prevent and/or mitigate the consequences of:*

*Trains exceeding the maximum permitted speed.*

In this example, each train has a device to measure its speed and secondary braking equipment. If its speed exceeds the predefined maximum speed permitted, the secondary braking system automatically decreases its speed. Trains may move between jurisdictions of different transport operators (e.g., international trains or through-services between a subway and a local railway). The maximum speeds permitted depend on the country and transport operator, but they are fixed for a given operator.

We apply the five steps of the proposed approach to this scenario, which are outlined below. Table 3 summarizes the results.

Step 1. The set of assets $S$ that system $T$ should protect are {$S_1$: Safety of passengers, crew, and others including nearby residents, $S_2$: Equipment (i.e., trains, rails, and stations)}. Lifecycle $L$ of system $T$ is {$L_1$: Initialization, $L_2$: System startup, $L_3$: Service startup, $L_4$: Service (speed monitoring), $L_5$: Operators change, $L_6$: Service termination, $L_7$: System shutdown, $L_8$: Reconfiguration, $L_9$: System update/evolution, $L_{10}$: System disposal}.

Step 2. Since we could not find publicly shared hazard information on this domain, we extracted hazards from the requirements document of the Institution of Railway Signal Engineers [66] mentioned above. The set of hazards $H$ are {$H_1$: Malfunction of train speed limit enforcement system $T$ causing threats to the safety of passengers, crew, and others and equipment health}. A set of critical adaptation execution $A_e$ are {$A_{e1}$: System $T$ uses the secondary braking system to reduce the train's speed when the maximum permitted speed is exceeded.}

Step 3. Corresponding plan stage $A_{p1}$ to execute stage $A_{e1}$ judges whether the current speed exceeds the maximum permitted speed by comparing the current train speed and the maximum permitted speed. As shown in Table 3, the corresponding analyze and monitor stages to plan stage $A_{p1}$ obtain the current speed and maximum permitted speed. The current train speed is measured by a speed measurement device without analysis. Thus, the stages to obtaining the current train speed are ($A_{m1.1}$: System $T$ obtains the current train speed from speed measurement device, $A_{a1.1}$: No interpretation or analysis required). The maximum permitted speed is obtained and set at service startup. The train driver inputs the service ID to system $T$. A train operator, the corresponding maximum permitted speed, and departure and arrival stations are obtained by searching a table with the service ID. Thus, the stages to obtain the maximum permitted speed at service startup are ($A_{m1.2}$: Driver inputs service ID to system $T$ at service startup, $A_{a1.2}$: System $T$ obtains the transport operator at departure from the service ID and searches for the maximum permitted speed in a table). When the train operators change, system $T$ receives a radio signal that represents the station ID from the train station platform. The stages to obtain maximum permitted speed at reconfiguration are ($A_{m1.3}$: System $T$ receives train station ID by radio signal from the train station platforms during the service, $A_{a1.3}$: System $T$ obtains transport operator at a station from the station ID and searches for the maximum permitted speed in a table).

Step 4. Table 3 summarizes the automation levels and the quality attributes. Automation level $e_l$ for $A_{e1}$ and $A_{p1}$ is 5 because the plan and execute stages are fully automatic. Detailed descriptions for the estimated quality attributes are shown in Table 10 (in Appendix B). Automation level $e_l$ for $A_{a1.2}$ and $A_{a1.3}$ are 5. Automation level $e_l$ for $A_{m1.1}$ is 5 because the measurement is fully automated. Automation level $e_l$ for $A_{m1.2}$ is 2 because the driver inputs the service ID at the service startup. Automation level $e_l$ for $A_{m1.3}$ is 5 because the radio signal receiver is always monitoring radio signals that indicate station ID.

Step 5. The acceptable critical level is determined as level 4 for all of the adaptations. The results show that the estimated reliability $e_r$ for $A_{m1.3}$ does not satisfy the acceptable critical level. There are two possible solutions to avoid this. One is to increase reliability $e_r$ for $A_{m1.3}$. Improving noise proofing ability or using other communication media will increase reliability. The other is to increase detectability $e_d$ and recoverability $e_c$ for $A_{m1.3}$ and $A_{a1.3}$. Increasing awareness by driver or external system leads to increasing the detectability. For example, adding a maximum permitted speed display or a maximum permitted speed change alert for the driver will increase the detectability.

## IV. CASE STUDY

### A. OVERVIEW

We conducted a case study to evaluate whether the proposed approach can assess a system with critical adaptation features and to compare the results of the proposed approach and

**TABLE 3.** Results of the example.

| Stage | $H_1$ | $e_l$ | $e_r$ | $e_d$ | $e_c$ |
|---|---|---|---|---|---|
| $A_e$ | $A_{e1}$: System $T$ uses the secondary braking system to reduce the train's speed when the maximum permitted speed is exceeded. | 5 | 5 | 5 | 1 |
| $A_p$ | $A_{p1}$: System $T$ judges whether the current speed exceeds the maximum permitted speed by comparing the current train speed and the maximum permitted speed. | 5 | 5 | 1 | 1 |
| $A_a$ | $A_{a1.1}$: No interpretation or analysis required. | - | - | - | - |
| | $A_{a1.2}$: System $T$ obtains the transport operator at departure from the service ID and searches for the maximum permitted speed in a table. | 5 | 5 | 4 | 5 |
| | $A_{a1.3}$: System $T$ obtains the transport operator at a station from the station ID and searches for the maximum permitted speed in a table. | 5 | 5 | 1 | 1 |
| $A_m$ | $A_{m1.1}$: System $T$ obtains the current train speed from speed measurement device. | 5 | 5 | 4 | 1 |
| | $A_{m1.2}$: Driver inputs the service ID to system $T$ at service startup. | 2 | 4 | 4 | 5 |
| | $A_{m1.3}$: System $T$ receives the train station ID by a radio signal from the train station platforms during the service. | 5 | 3 | 1 | 1 |

the results of STAMP/STPA approach [15]. STAMP/STPA is used as an approach of conventional hazard analysis. One of the authors conducted the assessment with the proposed approach followed by analysis with STAMP/STPA along with the procedures defined in the document [67].

First, STAMP/STPA identifies accidents, hazards, safety constraints, and control structures of the system. Second, the approach extracts unsafe control actions (UCA) using a table, where the columns correspond to the four-hazard guide words and rows correspond to the controls identified in the control structure. Third, the approach selects unsafe control actions from the table and creates the corresponding safety constraints. Finally, the approach identifies causal factors. In this case study, we added an assessment process with identified causal factors because STAMP/STPA does not define an assessment process.

### B. MATERIAL

Our case study involved a safety-critical telecommunication monitoring and switching system for Japanese public organizations (including local government) in actual use. The system checks the connection availability and switches to available connections. Fig. 6 shows the telecommunication monitoring and switching system in the case study. The telecommunication system transports safety-critical emergency information, including voice, video, and data communications. The communication network is a star network consisting of one control node and two or more terminal nodes. Each terminal node is connected to the control node by long-distance wired and long-distance wireless connections. The telecommunication monitoring and switching system monitors both the wired and wireless connections, including the communication terminal devices on both sides. Fig. 7 shows the control structure of the system. The control structure consists of the monitoring and switching system, the communication terminal device, the wired/wireless network, and the communication device.

In the operation, the telecommunication monitoring and switching system selects an available connection and switches the connection, if needed. The switching feature is a critical adaptation feature because communication is unavailable when switching to an unavailable connection. Moreover, environmental noise affects long-distance wireless network and the redundant architecture of communication terminal devices must be considered. The long-distance wireless connection is easy to be affected by environmental noise. Communication terminal devices in terminal nodes have different redundant architectures for each terminal node such as fully redundant hot standby for both wired and wireless connections (terminal node 1 in Fig. 6) and a shared communication terminal device for wired and wireless connections.

**TABLE 4.** Effort required for the proposed approach.

| Step | Effort (person-hour) |
|---|---|
| 1 | 0.5 |
| 2 | 0.5 |
| 3 | 0.5 |
| 4 | 1.0 |
| 5 | 0.5 |

### C. RESULTS BY THE PROPOSED APPROACH

Table 4 shows the required effort for the case study. The required effort for the proposed approach was three person-hours. Step 4 required 1.0 person-hour, while the effort for the other steps was 0.5 person-hours. Tables 5 and 6 summarize the results. The detailed descriptions are as follows.

Step 1. The set of assets $S$ that the system should protect is {$S_1$: Safety-critical emergency communication}. Lifecycle $L$ is {$L_1$: Setup for the control node and the terminal nodes, $L_2$: Startup of control node, $L_3$: Startup of terminal nodes, $L_4$: Connection setup, $L_5$: Monitoring system startup, $L_6$: Monitoring the control node and the terminal nodes, $L_7$: Maintenance and reconfiguration of terminal nodes, $L_8$: Nodes and system shutdown, $L_9$: System disposal}.

Step 2. The assessor used architectural design documents as diagrams providing entire images of the system and the results of root cause analysis in past development as the hazard information. The set of hazards $H$ is {$H_1$: Communication is unavailable due to failure to select an available connection, $H_2$: Communication is unavailable due to failure to detect

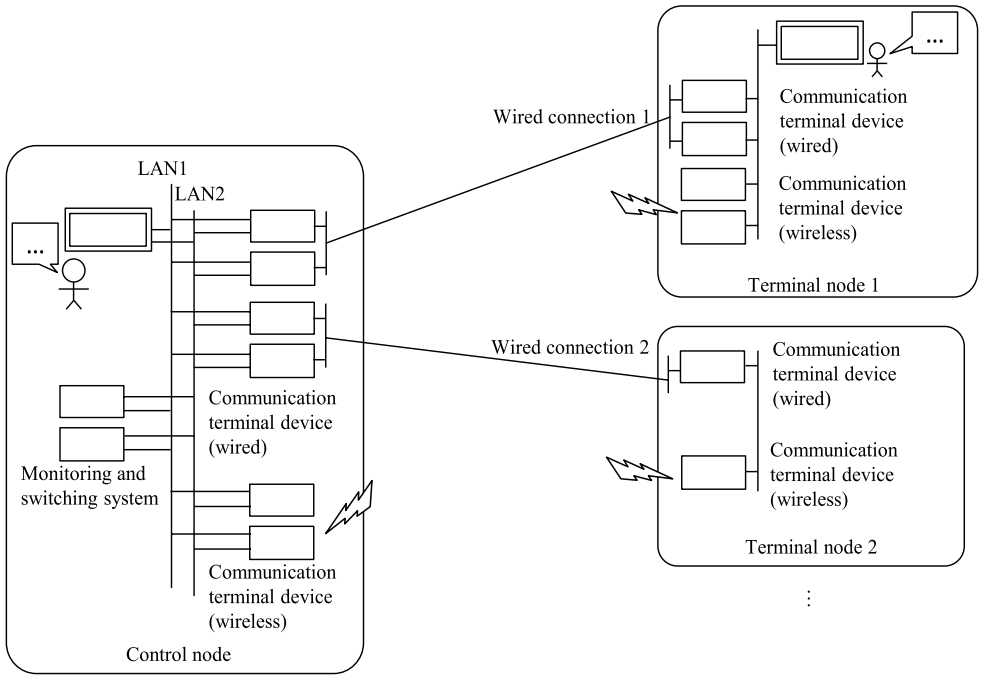


**FIGURE 6.** Telecommunication monitoring and switching system in the case study.

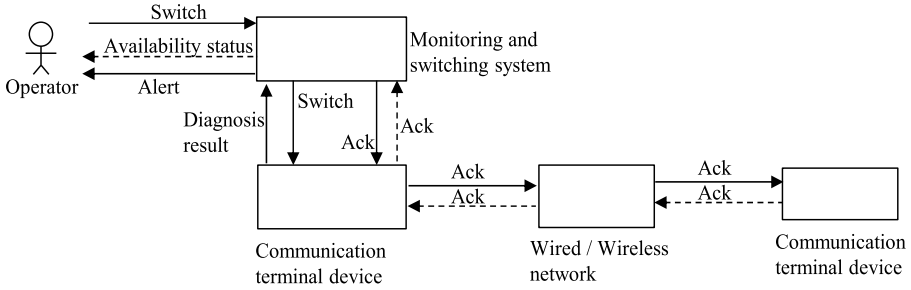


**FIGURE 7.** Control structure of the system.

control node failure}. A set of critical adaptation execution $A_e$ is {$A_{e1}$: System switches the connection upon detecting a failure of the connection in use, $A_{e2}$: System raises an alert when a control node failure is detected}.

Step 3. Corresponding plan stage $A_{p1}$ to execution stage $A_{e1}$ is used to judge whether the current connection is available. As shown in Table 4, the corresponding analyze stages to plan stage $A_{p1}$ are to analyze whether the wired and wireless connections are available. Connection availability is analyzed by receiving acknowledgment packets that are commonly used as a simple connectivity check and by ensuring that redundant or spare communication terminal devices are available. Corresponding plan stage $A_{p2}$ to execution stage $A_{e2}$ is to judge the health status of the control node. The health status is analyzed from the self-diagnostic results in communication terminal devices. The availability of the wired and wireless connection is checked by sending and receiving acknowledgment packets from the control node to the terminal nodes. Monitor stages $A_{m1.1}$ and $A_{m1.2}$ send and receive acknowledgment packets for the wired and wireless connections. Analyze stages $A_{a1.1}$ and $A_{a1.2}$ analyze the connectivity for the wired and wireless connections, the health status of the communication terminal devices from the acknowledgment packets, and the redundancy architecture of the devices.

Step 4. Tables 5 and 6 summarize the results. Automation level $e_l$ for $A_{e1}$ and $A_{p1}$ in the system is 2 because the switching plan requires not only the connection availability but also a forecast for noise affecting the wireless connection. Tables 11 and 12 (in Appendix C) show detailed descriptions for the estimated quality attributes for $A_{e1}$ and $A_{e2}$, respectively.

Automation levels $e_l$ for $A_{m1.1}$ and $A_{m1.2}$ are 5 because the sending and receiving of acknowledgment packets are fully automated. Automation levels $e_l$ for $A_{a1.1}$ and $A_{a1.2}$ are 3 because the analysis includes redundancy of

**TABLE 5.** Result for $H_1$.

| Stage | Description | $e_l$ | $e_r$ | $e_d$ | $e_c$ |
|---|---|---|---|---|---|
| $A_e$ | $A_{e1}$: System switches the connection upon detecting a failure of the connection in use. | 2 | 5 | 5 | 5 |
| $A_p$ | $A_{p1}$: System checks the availability of the connection in use. | 2 | 4 | 3 | 5 |
| $A_a$ | $A_{a1.1}$: System checks the wired connection availability and terminal device redundancy. | 3 | 4 | 1 | 5 |
| | $A_{a1.2}$: System checks the wireless connection availability and terminal device redundancy. | 3 | 4 | 1 | 5 |
| $A_m$ | $A_{m1.1}$: System sends and receives acknowledgment packets for the wired connection. | 5 | 5 | 1 | 1 |
| | $A_{m1.2}$: System sends and receives acknowledgment packets for the wireless connection. | 5 | 4 | 1 | 1 |

**TABLE 6.** Result for $H_2$.

| Stage | Description | $e_l$ | $e_r$ | $e_d$ | $e_c$ |
|---|---|---|---|---|---|
| $A_e$ | $A_{e2}$: System raises an alert when a control node failure is detected. | 4 | 5 | 5 | 5 |
| $A_p$ | $A_{p2}$: System checks the control node availability. | 4 | 4 | 1 | 5 |
| $A_a$ | $A_{a2}$: System analyzes the health status of the communication terminal devices in the control node. | 5 | 5 | 1 | 1 |
| $A_m$ | $A_{m2}$: System receives self-diagnostic results from the communication terminal devices in the control node. | 5 | 5 | 1 | 1 |

communication terminal devices. The redundant architectures among the terminal nodes differ. In case of communication terminal device failures, the operator must consider the redundant architectures.

Automation level $e_l$ for $A_{e2}$ is 4 because the operator can stop an alert if needed. Automation level $e_l$ for $A_{p2}$ is 4 because the operator must intervene in rare cases. In rare cases, the communication terminal device redundancy is not sufficient. Automation levels $e_l$ for $A_{a2}$ and $A_{m2}$ are 5 because the health check analysis from the self-diagnostic is fully automated.

Step 5. The acceptable critical level is determined as level 4 for all of the adaptations. The results show that estimated reliabilities $e_l$ for $A_1((A_{m1.1}, A_{m1.2}), (A_{a1.1}, A_{a1.2}), A_{p1}, A_{e1})$, and $A_2(A_{m2}, A_{a2}, A_{p2}, A_{e2})$ satisfy the critical level. Thus, the criticality of hazards $H_1$ and $H_2$ are acceptable, except in rare cases.

### D. RESULTS BY STAMP/STPA

Table 7 shows the required effort for the case study. The required effort was 4.5 person-hours. The proposed approach required 33% less effort than that of STAMP/STPA. An accident of the system is "unable to communicate with a terminal node." The hazards are the same as the results by the proposed approach, $H_1$: Communication is unavailable due to failure to select an available connection and $H_2$: Communication is unavailable due to failure to detect control node failure. Safety constraints are $C_1$: System selects available connection when the wired or wireless connection is available and $C_2$: Control node is available.

**TABLE 7.** Effort required for STAMP/STPA.

| Step | Effort (person-hour) |
|---|---|
| Preparation | 1.5 |
| 1: Extracting unsafe control action | 1.5 |
| 2: Identifying hazard causal factor | 1.5 |

Table 8 shows the identified unsafe control actions, where brackets denote unsafe controls that may cause hazards $H_1$ or $H_2$. The causal factors for the critical adaptation features are equivalent as described in plan stages $A_{p1}$ and $A_{p2}$ in Tables 5 and 6. The assessment results indicate that criticalities are acceptable.

## V. DISCUSSION

### A. ASSESSMENT AS A QUALITY ASSURANCE ACTIVITY

The case study shows that the proposed approach can assess the criticalities of a safety-critical telecommunication system. Even if a self-assessment is implemented (the assessor of the proposed approach is one of the developers of the adaptive software system), the assessor should be able to effectively assess from different viewpoints at development time. In ad hoc reviews, the effectiveness of the review decreases when the developer reviews an artifact that he or she developed.

The case study shows empirical evidence that the proposed approach could assess criticality by focusing on safety-critical adaptations and obtain equivalent results of STAMP/STPA. The proposed approach can analyze tradeoffs between human-assisted and automated critical adaptations, which other assessment techniques did not explicitly specify any procedure. Furthermore, if hazard information and diagrams providing entire images of the system are available, the required effort for step 3 in the proposed approach is expected to be less because mapping hazards to critical adaptations requires less effort. Also, in the case that traceability management is required by safety regulations, a part of steps 2 and 3 in the proposed approach can be performed semi-automatically.

The assessment by the proposed approach clarified the criteria to increase the automation level while discussing the reason for the low automation level of critical adaptations $A_{e1}$ and $A_{p1}$, which were attributed to redundant architectures among the terminal nodes and difficulties predicting trends of the noise affecting wireless connection quality. The discussed criteria were deploying the same redundant architectures in all terminal nodes and developing technology to predict wireless noise trends.

**TABLE 8.** Unsafe control actions.

| Control Action | Action Required but not Provided | Unsafe Action Provided | Incorrect Timing/Order | Stopped too Soon/Applied too Long |
|---|---|---|---|---|
| CA1: System switches to another connection. | CA1-N1: System does not switch to another connection when the connection in use or the communication terminal device is unavailable. [$H_1$] | CA1-P1: System switches to another connection when another connection or communication terminal device for the other connection is unavailable. [$H_1$] | CA1-T1: System switches to another connection before the connection in use or communication terminal device is unavailable.<br>CA1-T2: System switches to another connection after the connection in use or communication terminal device is unavailable. [$H_1$] | N/A |
| CA2: System raises an alert and stops the control node. | CA2-N1: System raises no alert and does not stop the control node in case of device communication failure for both connections. [$H_2$] | CA2-P1: System raises an alert and stops the control node when either communication terminal device is available. | CA2-T1: System raises an alert and stops the control node after the communication devices are unavailable.<br>CA2-T2: System raises an alert and stops the control node before the communication devices are unavailable. | N/A |

**TABLE 9.** Automation level in each of the phases.

| Level | Summary | Stage | | | |
|---|---|---|---|---|---|
| | | Monitor | Analyze | Plan | Execute |
| 5 | Fully automated. No intervention by humans or external systems allowed. | System always monitors information unaided. | System always analyzes the monitored information unaided. | System always makes plans unaided. | System always implements the necessary executions unaided. |
| 4 | System does not require assistance, but humans or external systems may intervene. | System monitors information unaided, but humans or external systems may check and change the monitored information if necessary. | System analyzes the monitored information unaided, but humans or external systems may check and change the results if necessary. | System makes plans unaided, but humans or external systems may check and change the plan selection if necessary. | System implements the necessary executions unaided, but humans or external systems substitute these executions if necessary. |
| 3 | System may request assistance, if required. | Humans or external systems may be asked for assistance if the information monitor criteria are not satisfied. | Humans or external systems may be asked for assistance if the analysis criteria are not satisfied. | Humans or external systems may be asked for assistance if the plan selection criteria are not satisfied. | Humans or external systems may be asked for assistance if the plan execution criteria are not satisfied. |
| 2 | Humans or external systems are required to conduct one or more processes in a predefined way. | Humans or external systems are required for part of the information monitor process. | Humans or external systems are required for part of the information analysis process. | Humans or external systems are required for part of the plan selection process. | Humans or external systems implement part of the necessary executions. |
| 1 | Fully manual. | Humans or external systems monitor all information. | Humans or external systems analyze all information. | Humans or external systems select plans. | Humans or external systems implement all executions. |

### B. THREATS TO VALIDITY

Although the results of the proposed approach and the STAMP/STPA approach indicate that the criticalities of the adaptations are acceptable, the assessment results in the case study should be verified. The safety-critical telecommunication system in the case study has been in operation for more than 10 years. During the operation, critical adaptations have worked as expected, which may be considered as a verification of the assessment results.

The evaluation was conducted by one of the authors because the assessors in the case study must fully understand the target system and no other qualified assessor was available. Although the assessor carefully tried to use both approaches without bias and the STAMP/STPA approach was conducted after the proposed approach, conducting another case study with assessors that excluded the authors is an important future work.

The adaptive software system in the case study had a few critical adaptations. Although the critical adaptations in the system are not specific to the target system, investigations with a larger number of critical adaptations enable the effectiveness and efficiency of the proposed approach to be generalized.

Investigating the required skill and experience for the assessor is also an important future work. Identifying critical adaptations and decomposing critical adaptations into

**TABLE 10. Detailed description of the quality attributes in the example.**

| Adaptation | Quality Attribute | Level | Description |
|---|---|---|---|
| $A_{e1}$ | $e_r$ | 5 | Component has been proven by its operational performance in many train operators and train cars. |
| | $e_d$ | 5 | Driver can immediately recognize that the second braking system is activated. |
| | $e_c$ | 1 | Intervention by the driver is impossible. |
| $A_{p1}$ | $e_r$ | 5 | Component has been proven by its operational performance in many train operators and cars. |
| | $e_d$ | 1 | Driver cannot monitor the plan. |
| | $e_c$ | 1 | Intervention by the driver is impossible. |
| $A_{a1.2}$ | $e_r$ | 5 | Implemented as a table search with a given service ID. |
| | $e_d$ | 4 | Driver can verify the departure and destination stations determined by input service ID. |
| | $e_c$ | 5 | Driver can re-enter the correct service ID. |
| $A_{a1.3}$ | $e_r$ | 5 | Implemented as a table search with a given station ID. |
| | $e_d$ | 1 | Driver cannot verify the received station ID from a radio signal. |
| | $e_c$ | 1 | Driver cannot change the station ID from a radio signal. |
| $A_{m1.1}$ | $e_r$ | 5 | Speed measurement device and its self-diagnostic feature have been proven by its operational performance. |
| | $e_d$ | 4 | |
| | $e_c$ | 1 | An accurate train speed cannot be obtained if the speed measurement device fails. |
| $A_{m1.2}$ | $e_r$ | 4 | Driver inputs service ID at the service startup. |
| | $e_d$ | 4 | Driver can verify the service ID by ensuring the departure and destination stations determined by input service ID. |
| | $e_c$ | 5 | Driver can re-enter the correct service ID. |
| $A_{m1.3}$ | $e_r$ | 3 | Some station platforms are noisy for receiving radio signal. |
| | $e_d$ | 1 | Received station ID is not displayed to the driver. |
| | $e_c$ | 1 | Monitoring is fully automated. |

**TABLE 11. Detailed descriptions of the quality attributes for $A_{e1}$ in the case study.**

| Adaptation | Quality Attribute | Level | Description |
|---|---|---|---|
| $A_{e1}$ | $e_r$ | 5 | Switching operation and switching equipment are simple and reliable. |
| | $e_d$ | 5 | Operator always checks the status shown by a display. |
| | $e_c$ | 5 | Operator can switch the connection at any time. |
| $A_{p1}$ | $e_r$ | 4 | System cannot predict trends of wireless noise. The operator might misunderstand the status. |
| | $e_d$ | 3 | No other operator can verify the plan. |
| | $e_c$ | 5 | Operator can switch the connection at any time. |
| $A_{a1.1}$ | $e_r$ | 4 | Operator might misjudge the connectivity if the redundant architecture is not taken into consideration. |
| | $e_d$ | 1 | No other operators can verify, and no feature or device detects incorrect analysis. |
| | $e_c$ | 5 | The operator can correct the analysis result. |
| $A_{a1.2}$ | $e_r$ | 4 | Operator might misjudge the connectivity if the redundant architecture is not taken into consideration. |
| | $e_d$ | 1 | No other operators can verify, and no feature or device detects incorrect analysis. |
| | $e_c$ | 5 | Operator can correct the analysis result. |
| $A_{m1.1}$ | $e_r$ | 5 | Sending and receiving acknowledgement packets are reliable for the wired connection. |
| | $e_d$ | 1 | Incorrect acknowledgement packets are hard to detect and correct. |
| | $e_c$ | 1 | |
| $A_{m1.2}$ | $e_r$ | 4 | Sending and receiving acknowledgement packets are occasionally affected by noise. |
| | $e_d$ | 1 | Incorrect acknowledgement packets are hard to detect and correct. |
| | $e_c$ | 1 | |

the stages require that the assessor understands the critical adaptations and target system well.

The proposed approach can assess adaptations, including dynamic configurations and autonomy adjustment mechanisms. Adaptations with dynamic configurations and autonomy adjustments are added to the critical adaptation executions $A_e$ (Step 2 in Subsection III-G) and assessed as critical adaptations $A$ (Step 3 in Subsection III-G). The subsequent analysis (Steps 4 and 5 in Subsection III-G) assesses whether the critical adaptations are acceptable. Here, we use the six levels of driving automation (Levels 0 to 5) defined by SAE J3016 [68], [69] as an example. SAE J3016 defines "Level 2: Partial Driving Automation" as the driver must constantly supervise driver support features and "Level 3: Conditional Driving Automation" as the system can drive the vehicle under limited conditions but when the system requests, the driver must take over. If the limited condition of Level 3 is on the highway (the vehicle is on the highway), and Level 2 is used on local roads. When a vehicle takes local roads from the highway, an adjustable autonomy is required to transition from Level 3 to Level 2. In this case, the assessments should include not only adaptations in Levels 2 ($A_1$) and 3 ($A_2$)

**TABLE 12. Detailed descriptions of quality attributes for $A_{e2}$ in the case study.**

| Adaptation | Quality Attribute | Level | Description |
|---|---|---|---|
| $A_{e2}$ | $e_r$ | 5 | Alerting failure and stopping the control node are systematic. |
| | $e_d$ | 5 | Operator receives an alert. |
| | $e_c$ | 5 | Operator can stop an alert. |
| $A_{p2}$ | $e_r$ | 4 | Communication terminal device redundancy is insufficient. In rare cases, the operator can intervene. |
| | $e_d$ | 1 | No opportunity for other operators' verification or diagnostic features. |
| | $e_c$ | 5 | Operator can intervene. |
| $A_{a2}$ | $e_r$ | 5 | Analysis automatically checks the result of self-diagnostics of the communication terminal devices, local area network (LAN) availability, and redundancy of the devices and LAN. |
| | $e_d$ | 1 | No verification means are provided. |
| | $e_c$ | 1 | Operator cannot intervene. |
| $A_{m2}$ | $e_r$ | 5 | Self-diagnostic results of the communication terminal devices and LAN availability in the control node are reliable. |
| | $e_d$ | 1 | No verification means are provided. |
| | $e_c$ | 1 | Operator cannot intervene. |

but also the adjustable autonomy from Level 3 to Level 2 ($A_3$). The autonomy adjustment must consider two factors: (1) the driver is ready to take over and (2) the automation system of Level 3 does not control the vehicle for emergency avoidance of obstacles. In this case, the proposed approach can assess the adjustable autonomy by adding $A_1$, $A_2$, and $A_3$ to adaptations $A$.

## VI. CONCLUSION AND FUTURE WORK

In this work, we propose an approach for identifying and assessing critical adaptations by decomposing adaptations in adaptive software systems into stages. First, the approach identifies assets that should be protected by the system. Next, it identifies hazards and critical adaptations, which are decomposed into monitor, analyze, plan, and execute stages. Then, the proposed approach determines the automation level for each stage and estimates its quality attributes, which consist of reliability, detectability, and recoverability. Finally, the proposed approach assesses whether the possibilities of threats exposed by critical adaptations are acceptable from the decomposed stages and their quality attributes.

We conducted a case study to evaluate the effectiveness and efficiency of the proposed approach in a safety-critical telecommunication system developed in industry. The proposed approach identified two critical adaptations, decomposed them into four stages, and assessed whether their criticalities were acceptable. In the evaluation, the proposed approach with structured views, including hazards and critical adaptations, achieves similar results to those of the STAMP/STPA approach but requires 33% less effort.

The case study shows that the proposed approach and the STAMP/STPA approach yield equivalent results. Hence, the proposed approach enables fine-grained analysis to clarify criteria to increase the automation level by considering the quality attributes. The proposed approach can be used in safety assessment approaches such as STAMP/STPA and FTA. It is also applicable as a safety analysis technique required by safety standards, including IEC 61508 [64] and ISO 26262 [70]. In addition, the structured views provided by the proposed approach enable efficient quality assurance activities.

Future works include refining the procedures in the proposed approach so that they are automated, which may realize runtime assessments such as analyzing and assessing dynamic mechanisms to enable adaptive software systems to change their automation levels at runtime such as dynamic configurations and adjustable autonomies.

## APPENDIX A

Detailed descriptions of automation levels in Section III

## APPENDIX B

Detailed descriptions of quality attributes in the example in Section III

## APPENDIX C

Detailed description of the quality attributes in Section IV

## ACKNOWLEDGMENT

The authors would like to thank the members of the working group on IoT High Reliability, Software Reliability Enhancement Center, technology headquarters, Information-Technology Promotion Agency, Japan.

## REFERENCES

[1] D. Garlan, S.-W. Cheng, A.-C. Huang, B. Schmerl, and P. Steenkiste, "Rainbow: Architecture-based self-adaptation with reusable infrastructure," *Computer*, vol. 37, no. 10, pp. 46–54, Oct. 2004.

[2] B. Schmerl, G. A. Moreno, A. Mellinger, J. Camara, and D. Garlan, *Architecture-based Self-adaptation for Moving Target Defense*. Pittsburgh, PA, USA: Carnegie Mellon University, 2014.

[3] G. A. Moreno, J. Cámara, D. Garlan, and M. Klein, "Uncertainty reduction in self-adaptive systems," in *Proc. IEEE/ACM 13th Int. Symp. Softw. Eng. Adapt. Self-Managing Syst. (SEAMS)*, Gothenburg, Sweden, May 2018, pp. 51–57.

[4] C. J. Foster, K. L. Plant, and N. A. Stanton, "A delphi study of human factors methods for the evaluation of adaptation in safety-related organisations," *Saf. Sci.*, vol. 131, Nov. 2020, Art. no. 104933.

[5] M. Chaal, O. A. V. Banda, J. A. Glomsrud, S. Basnet, S. Hirdaris, and P. Kujala, "A framework to model the STPA hierarchical control structure of an autonomous ship," *Saf. Sci.*, vol. 132, Dec. 2020, Art. no. 104939.

[6] C. Krupitzer, T. Temizer, T. Prantl, and C. Raibulet, "An overview of design patterns for self-adaptive systems in the context of the Internet of Things," *IEEE Access*, vol. 8, pp. 187384–187399, 2020.

[7] R. Calinescu, R. Mirandola, D. Perez-Palacin, and D. Weyns, "Understanding uncertainty in self-adaptive systems," in *Proc. IEEE Int. Conf. Autonomic Comput. Self-Organizing Syst. (ACSOS)*, Washington, DC, USA, Aug. 2020, pp. 242–251.

[8] J. Cámara, G. Moreno, and D. Garlan, "Reasoning about human participation in self-adaptive systems," in *Proc. IEEE/ACM 10th Int. Symp. Softw. Eng. Adapt. Self-Managing Syst.*, Florence, Italy, May 2015, pp. 146–156.

[9] M. Gil, V. Pelechano, J. Fons, and M. Albert, "Designing the human in the loop of self-adaptive systems," in *Proc. UCAmI, Int. Conf. Ubiquitous Comput. Ambient Intell.*, Gran Canaria, Spain, 2016, pp. 437–449.

[10] J. Andersson, R. De Lemos, S. Malek, and D. Weyns, "Modeling dimensions of self-adaptive software systems," in *Software Engineering for Self-Adaptive Systems*. New York, NY, USA: Springer, 2009, pp. 27–47.

[11] J. O. Kephart, "Research challenges of autonomic computing," in *Proc. ICSE 27th Int. Conf. Softw. Eng.*, New York, NY, USA, 2005, pp. 15–22.

[12] P. Oreizy, M. M. Gorlick, R. N. Taylor, D. Heimhigner, and G. Johnson, "An architecture-based approach to self-adaptive software," *IEEE Intell. Syst. Appl.*, vol. 14, no. 3, pp. 54–62, Jun. 1999.

[13] J. Cámara, P. Correia, R. de Lemos, D. Garlan, P. Gomes, B. Schmerl, and R. Ventura, "Evolving an adaptive industrial software system to use architecture-based self-adaptation," in *Proc. 8th Int. Symp. Softw. Eng. Adapt. Self-Managing Syst. (SEAMS)*, San Francisco, CA, USA, May 2013, pp. 13–22.

[14] J. Yang, G. Huang, W. Zhu, X. Cui, and H. Mei, "Quality attribute tradeoff through adaptive architectures at runtime," *J. Syst. Softw.*, vol. 82, no. 2, pp. 319–332, Feb. 2009.

[15] N. G. Leveson, *Engineering a Safer World: Systems Thinking Applied to Safety*. Cambridge, MA, USA: MIT Press, 2016.

[16] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, Jan. 2003.

[17] R. Parasuraman, T. B. Sheridan, and C. D. Wickens, "A model for types and levels of human interaction with automation," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 30, no. 3, pp. 286–297, May 2000.

[18] S. A. Mostafa, M. S. Ahmad, A. Ahmad, M. Annamalai, and S. S. Gunasekaran, "An autonomy viability assessment matrix for agent-based autonomous systems," in *Proc. Int. Symp. Agents, Multi-Agent Syst. Robot. (ISAMSR)*, Putrajaya, Malaysia, Aug. 2015, pp. 53–58.

[19] T. M. Roehr and Y. Shi, "Using a self-confidence measure for a system-initiated switch between autonomy modes," in *Proc. Int. Symp. Artif. Intell., Robot. Automat. Space*, Sapporo, Japan, 2010, pp. 507–514.

[20] S. Morisaki and N. Kasai, "An approach for detecting critical adaptations in automated adaptive software systems," in *Proc. IEEE Int. Conf. Softw. Qual., Rel. Secur. Companion (QRS-C)*, Lisbon, Portugal, Jul. 2018, pp. 526–530.

[21] G. A. Dorais, R. P. Bonasso, D. Kortenkamp, B. Pell, and D. Schreckenghost, "Adjustable autonomy for human-centered autonomous systems," in *Proc. IJCAI'99 Workshop Adjustable Autonomy Syst.*, Stockholm, Sweden, 1999, pp. 16–35.

[22] D. Kortenkamp, R. P. Bonasso, D. Ryan, and D. Schreckenghost, "Traded control with autonomous robots as mixed initiative interaction," in *Proc. AAAI Spring Symp.*, Palo Alto, CA, USA, 1997, pp. 89–94.

[23] M. Y. K. Cheng and R. Cohen, "A hybrid transfer of control model for adjustable autonomy multiagent systems," in *Proc. 4th Int. Joint Conf. Auto. Agents Multiagent Syst.*, The Netherlands, Jul. 2005, pp. 1149–1150.

[24] B. Sellner, F. W. Heger, L. M. Hiatt, R. Simmons, and S. Singh, "Coordinated multiagent teams and sliding autonomy for large-scale assembly," *Proc. IEEE*, vol. 94, no. 7, pp. 1425–1444, Jul. 2006.

[25] P. Scerri, D. V. Pynadath, and M. Tambe, "Why the elf acted autonomously: Towards a theory of adjustable autonomy," in *Proc. 1st Int. Joint Conf. Auto. Agents Multiagent Syst. (AAMAS)*, Bologna, Italy, 2002, pp. 857–864.

[26] S. Mercier, F. Dehais, C. Lesire, and C. Tessier, "Resources as basic concepts for authority sharing," in *Proc. Humans Operating Unmanned Syst. (HUMOUS)*, Brest, France, 2008, pp. 31–42.

[27] C. A. Miller and R. Parasuraman, "Designing for flexible interaction between humans and automation: Delegation interfaces for supervisory control," *Hum. Factors, J. Hum. Factors Ergonom. Soc.*, vol. 49, no. 1, pp. 57–75, Feb. 2007.

[28] L. A. M. Bush, A. J. Wang, and B. C. Williams, "Risk-based sensing in support of adjustable autonomy," in *Proc. IEEE Aerosp. Conf.*, Big Sky, MT, USA, Mar. 2012, pp. 1–18.

[29] B. Durand, K. Godary-Dejean, L. Lapierre, and D. Crestani, "Inconsistencies evaluation mechanisms for an hybrid control architecture with adaptive autonomy," in *Proc. 4th Nat. Conf. Control Archit. Robots*, Toulouse, France, 2009.

[30] M. Johnson, J. M. Bradshaw, P. Feltovich, C. Jonker, B. van Riemsdijk, and M. Sierhuis, "Autonomy and interdependence in human-agent-robot teams," *IEEE Intell. Syst.*, vol. 27, no. 2, pp. 43–51, Mar. 2012.

[31] S. Zieba, P. Polet, F. Vanderhaegen, and S. Debernard, "Principles of adjustable autonomy: A framework for resilient human–machine cooperation," *Cognition, Technol. Work*, vol. 12, no. 3, pp. 193–203, Sep. 2010.

[32] M. Ball and V. Callaghan, "Explorations of autonomy: An investigation of adjustable autonomy in intelligent environments," in *Proc. 8th Int. Conf. Intell. Environments*, Guanajuato, Mexico, Jun. 2012, pp. 114–121.

[33] D. Perez-Palacin, R. Mirandola, and J. Merseguer, "On the relationships between QoS and software adaptability at the architectural level," *J. Syst. Softw.*, vol. 87, pp. 1–17, Jan. 2014.

[34] P. Zoghi, M. Shtern, and M. Litoiu, "Designing search based adaptive systems: A quantitative approach," in *Proc. 9th Int. Symp. Softw. Eng. Adapt. Self-Managing Syst.*, Hyderabad, India, Jun. 2014, pp. 7–16.

[35] M. Salehie and L. Tahvildari, "Self-adaptive software: Landscape and research challenges," *ACM Trans. Auto. Adapt. Syst.*, vol. 4, no. 2, pp. 1–42, May 2009.

[36] D. Weyns, "Perpetual assurances for self-adaptive systems," in *Software Engineering for Self-Adaptive Systems III. Assurances*. New York, NY, USA: Springer, 2017, pp. 31–63.

[37] A. Elkhodary, N. Esfahani, and S. Malek, "FUSION: A framework for engineering self-tuning self-adaptive software systems," in *Proc. 18th ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, Santa Fe, NJ, USA, Nov. 2010, pp. 7–16.

[38] H. Nakagawa, A. Ohsuga, and S. Honiden, "Gocc: A configuration compiler for self-adaptive systems using goal-oriented requirements description," in *Proc. 6th Int. Symp. Softw. Eng. Adapt. Self-Managing Syst.*, Honolulu, HI, USA, May 2011, pp. 40–49.

[39] C. Dorn and R. N. Taylor, "Coupling software architecture and human architecture for collaboration-aware system adaptation," in *Proc. 35th Int. Conf. Softw. Eng. (ICSE)*, San Francisco, CA, USA, May 2013, pp. 53–62.

[40] M. Luckey and G. Engels, "High-quality specification of self-adaptive software systems," in *Proc. 8th Int. Symp. Softw. Eng. for Adapt. Self-Managing Syst. (SEAMS)*, San Francisco, CA, USA, May 2013, pp. 143–152.

[41] N. G. Leveson and J. P. Thomas, *STPA Handbook*. Cambridge, MA, USA: MIT Press, 2018.

[42] *Failure Modes and Effects Analysis (FMEA and FMECA)*, document IEC 60812:2018, 2018.

[43] *Fault Tree Analysis (FTA)*, document IEC 61025:2006, 2006.

[44] C. A. Ericson, *Hazard Analysis Techniques for System Safety*. New York, NY, USA: Wiley, 2015.

[45] *Hazard and Operability Studies (HAZOP Studies)-Application Guide*, document IEC 61882:2016, 2016.

[46] R. Kazman, G. Abowd, L. Bass, and P. Clements, "Scenario-based analysis of software architecture," *IEEE Softw.*, vol. 13, no. 6, pp. 47–55, Nov. 1996.

[47] R. Kazman, M. Barbacci, M. Klein, S. J. Carrière, and S. G. Woods, "Experience with performing architecture tradeoff analysis," in *Proc. 21st Int. Conf. Softw. Eng.*, Los Angeles, CA, USA, May 1999, pp. 54–63.

[48] K. Lee and B. Boehm, "Empirical results from an experiment on value-based review (VBR) processes," in *Proc. Int. Symp. Empirical Softw. Eng.*, Noosa Heads, QLD, Australia, 2005, p. 10.

[49] A. Porter and L. Votta, "Comparing detection methods for software requirements inspections: A replication using professional subjects," *Empirical Softw. Eng.*, vol. 3, no. 4, pp. 355–379, 1998.

[50] A. A. Porter, L. G. Votta, and V. R. Basili, "Comparing detection methods for software requirements inspections: A replicated experiment," *IEEE Trans. Softw. Eng.*, vol. 21, no. 6, pp. 563–575, Jun. 1995.

[51] V. R. Basili, S. Green, O. Laitenberger, F. Lanubile, F. Shull, and M. V. Zelkowitz, "The empirical investigation of perspective-based reading," *Empirical Softw. Eng.*, vol. 1, no. 2, pp. 133–164, 1996.
[52] O. Laitenberger, "Cost-effective detection of software defects through perspective-based inspections," *Empirical Softw. Eng.*, vol. 6, no. 1, pp. 81–84, 2001.
[53] T. Thelin, P. Runeson, and B. Regnell, "Usage-based reading—An experiment to guide reviewers with use cases," *Inf. Softw. Technol.*, vol. 43, no. 15, pp. 925–938, Dec. 2001.
[54] T. Thelin, P. Runeson, and C. Wohlin, "An experimental comparison of usage-based and checklist-based reading," *IEEE Trans. Softw. Eng.*, vol. 29, no. 8, pp. 687–704, Aug. 2003.
[55] H. Gomaa, K. Hashimoto, M. Kim, S. Malek, and D. A. Menascé, "Software adaptation patterns for service-oriented architectures," in *Proc. ACM Symp. Appl. Comput.*, Sierre, Switzerland, Mar. 2010, pp. 462–469.
[56] *Systems and Software Engineering—System Life Cycle Processes*, document ISO/IEC/IEEE 15288:201, 2015.
[57] *Systems and Software Engineering—Life Cycle Management—Part 1: Guidelines for Life Cycle Management*, document ISO/IEC/IEEE 24748-1:2018, 2018.
[58] S. R. E. Center, *Guidance for Practice Regarding IoT Safety/Security Development Guidelines*. Tokyo, Japan: Information-technology Promotion Agency, 2017.
[59] M. Yazdi, "Improving failure mode and effect analysis (FMEA) with consideration of uncertainty handling as an interactive approach," *Int. J. Interact. Design Manuf. (IJIDeM)*, vol. 13, no. 2, pp. 441–458, Jun. 2019.
[60] T. Myklebust, J. Eriksen, A. Hellandsvik, and G. Hanssen, "The agile FMEA approach," in *Proc. SCSC 26th Safety-Critical Syst. Symp.*, New York, NY, USA, 2018, pp. 1–17.
[61] J. R. Boyd, "The essence of winning and losing," 1996. [Online]. Available: https://fasttransients.files.wordpress.com/2010/03/essence_of_winning_losing.pdf
[62] J. R. Boyd, *A Discourse on Winning and Losing*. Maxwell AFB, AL, USA: Air Univ. Press, 2018.
[63] W. R. Dunn, "Designing safety-critical computer systems," *Computer*, vol. 36, no. 11, pp. 40–46, 2003.
[64] *Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems*, document IEC 61508:2016, 2016.
[65] *Standard Practice for System Safety*, Standard MIL-STD-882D:2000, 2000.
[66] F. How. (2014). *IRSE Fundamental Requirements for Train Control Systems*. [Online]. Available: https://www.rissb.com.au/wp-content/uploads/2019/03/AS-7711-2018_Signalling-Principles_Preview.pdf
[67] AS Primer. (2015). *An STPA Primer*. [Online]. Available: http://psas.scripts.mit.edu/home/wp-content/uploads/2015/06/STPA-Primer-v1.pdf
[68] *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*, Standard SAE J3016_202104, 2021.
[69] SAE International. (2021). *SAE J3016 Levels of Driving Automation*. [Online]. Available: https://www.sae.org/binaries/content/assets/cm/content/blog/sae-j3016-visual-chart_5.3.21.pdf
[70] *Road Vehicles - Functional Safety—Part 6: Product Development At the Software Level*, Standard ISO 26262-6:2018, 2018.

**SHUJI MORISAKI** (Member, IEEE) received the D.E. degree in information science from the Nara Institute of Science and Technology, Japan, in 2001. He is currently an Associate Professor with the Graduate School of Informatics, Nagoya University, Japan. Previously, he has been a Software Engineer in the Japanese software industry. His research interests include empirical software engineering and software quality.

**MICHIYO WAKIMOTO** received the Master of Management of Technology degree from Ritsumeikan University, Japan. She is currently pursuing the Ph.D. degree with the Graduate School of Informatics, Nagoya University, Japan. Her research interests include empirical software engineering and software quality.

**NORIMITSU KASAI** received the Doctor of Engineering degree from the Nara Institute of Science and Technology, Japan, in 2014. He is currently a System Engineer with Mitsubishi Electric Corporation. His research interests include software/system engineering and source code analysis.