STPA를 활용한 협업 가상물리시스템의 안전성 테스트 케이스 생성

허윤아○, 유준범
건국대학교 컴퓨터공학과
hyoona1202@naver.com, jbyoo@konkuk.ac.kr

Generation of Safety Test Cases for Cooperative Cyber-Physical Systems Using STPA

Yoona Heo○, Junbeom Yoo
Department of Computer Science and Engineering, Konkuk University

요 약

가상물리시스템 (Cyber-Physical System, CPS)은 연산 (computation)부, 제어 (control)부, 그리고 통신 (communication)부가 통합되어 동작하는 실시간 시스템으로, 일반적으로 안전성이 중요한 시스템이다. 많은 경우 서로 다른 CPS들이 공동의 목표를 달성하기 위하여 협업 (cooperate)하는데, 이러한 상황에서는 다양한 시스템이 협업할 때 나타날 수 있는 잠재적 위험 (hazard) 또한 고려되어야 한다. 따라서 개별적인 CPS 뿐 아니라 협업하는 CPS들 전체의 안전성을 확보하기 위해 노력할 필요가 있다. 본 연구에서는 안전성을 확보하기 위한 방안 중 하나인 안전성 테스팅을 위하여 위험 분석 기법인 Systems-Theoretic Process Analysis (STPA)를 활용하여 안전성을 검증하기 위한 테스트 케이스를 생성하는 프로세스를 제안한다. 또한, guideword를 제안하여 STPA 결과를 활용해 안전성 테스트 케이스를 작성함에 있어 도움을 줄 수 있도록 한다. 추가로, 기존에 개발한 테스트베드 시스템을 사례 연구 대상 시스템으로 활용하여 안전성 테스트 케이스를 생성하고, 생성한 테스트 케이스를 통해 안전성 테스팅까지 진행한 결과를 통해 제안하는 프로세스를 분석한다.

1. 서론

가상물리시스템 (Cyber-Physical System, CPS)은 연산 (computation)부, 제어 (control)부, 그리고 통신 (communication)부가 통합되어 동작하는 실시간 시스템 **[1, 2]**으로, 일반적으로 시스템에 문제가 발생할 경우 인명 피해, 재산 손실 등의 손실을 미칠 수 있는, 안전성 (safety)이 중요한 시스템이다. 많은 경우 서로 다른 CPS들이 공동의 목표를 달성하기 위하여 협업 (cooperate)한다. 시스템 이론에서는 시스템의 각 구성 요소 (component) 수준에서는 발견되지 않았던 특성들이 시스템 수준에서는 나타날 수 있다고 이야기하며, 이러한 특성들을 emergent property라고 부른다 **[3]**. 이와 마찬가지로, CPS가 공동의 목표를 가지고 서로 연계되어 동작, 즉 협업할 때, 개별 CPS에서는 확인할 수 없었던 emergent property가 나타날 수 있다. 이러한 Emergent property 중 하나인 safety는 개별 CPS 뿐 아니라 협업하는 여러 CPS에서도 중요하게 다뤄져야 한다. 다양한 CPS가 협업하면서, 개별 CPS에서는 나타나지 않았던 잠재적인 위험 (hazard)들이 등장할 수 있다. 따라서 개별적인 CPS 뿐 아니라 협업하는 CPS들 전체의 안전성을 확보하기 위해 노력할 필요가 있다.

일반적으로, 시스템 안전 표준인 미 국방부 (Department of Defense)의 MIL-STD-882E **[4]**, 기능 안전성 (functional safety) 표준인 IEC 61508 **[5]**, ISO 26262 **[6]**, 기능 안전성에 대한 직접적인 언급은 없으나 항공 분야에서의 기능 안전성 표준으로 간주되는 DO-178C **[7]** 등을 살펴보면, 안전 관련 (safety-related) 시스템 혹은 안전 중요 (safety-critical) 시스템의 안전성은 보장되어야 하며, 이를 위하여 위험 분석 (hazard analysis, HA), 리스크 평가 (risk assessment), 안전 요구사항 정의 등의 활동을 수행하여 안전성을 보장할 수 있어야 한다고 말한다. 이 외에도 시스템의 안전성을 확보하기 위한 활동에는 안전성에 대한 검증 (verification)도 포함될 수 있다 **[8]**.

본 연구에서는 안전성을 확보하기 위한 방안으로 safety verification 방법 중 하나인 안전성 테스팅 (safety testing)을 위하여 위험 분석 기법인 Systems-Theoretic Process Analysis (STPA) **[9]**의 결과를 활용하여 안전성 테스트 케이스 (test case, TC)를 수동으로 생성하는 프로세스를 제안한다. 이 때 품질 특성 (Quality Attribute, QA) 요구사항을 시나리오 형태로 나타내는 Quality Attribute Scenario (QAS) **[10]**의 형태를 빌려 안전성 테스트 케이스를 생성한다. 또한, TC 생성에 있어 도움을 줄 수 있는 guideword를 제안하여 STPA 결과를 활용해 safety TC를 생성함에 있어 도움을 줄 수 있도록 한다. 추가로, 사례 연구를 통해 사례 연구 대상 시스템의 safety TC를 생성

하고, 생성한 TC를 활용하여 safety testing을 진행하고 그 결과를 살펴본다.

본 논문의 구성은 다음과 같다. 2장에서는 앞서 언급된 STPA와 QAS에 대한 배경지식을 설명하고, 3장에서는 관련 연구에 대하여 설명한다. 다음으로 4장에서는 제안하는 안전성 테스트 케이스 생성 프로세스에 대한 내용을 설명하고, 5장에서는 사례 연구 대상 시스템과 프로세스 적용 결과에 대하여 설명하고 논의한다. 마지막으로 6장에서는 결론과 향후 연구에 대해 기술한다.

2. 배경지식

2.1 Systems-Theoretic Process Analysis (STPA)

Systems-Theoretic Process Analysis (STPA)는 HA 기법의 한 종류로, 시스템 이론에 기반한 causality model인 Systems-Theoretic Accident Model and Process (STAMP)에서 새롭게 식별된 원인 요소 (causal factor)를 포함하기 위해 제안된 기법이다 **[9]**. **[11]**에 따르면, STPA는 총 네 단계로 구성된다.

첫 번째 단계는 '분석 목적 정의' 단계이다. 이 단계에서는 시스템 수준에서 발생 가능한 loss (accident)와 각 loss를 발생시키는 원인이 되는 시스템 수준에서의 hazard를 식별한다. 이 과정에서 분석 대상 시스템과 그 경계 또한 식별한다. 또한 hazard를, 더 나아가서는 시스템의 loss를 막기 위한 시스템 수준의 제약 사항까지도 식별한다.

두 번째 단계에서는 control structure를 모델링한다. Control structure는 하나 이상의 control-feedback loop로 구성된 시스템의 계층 관계를 나타내는 추상적 모델이다. 각 control-feedback loop는 control의 주체가 되는 controller, control의 대상이 되는 controlled process, controller가 controlled process를 제어하기 위해 제공하는 control action (CA), 그리고 controlled process가 controller에게 제공하는 feedback으로 구성된다. 이 외에도 controller에서 controlled process를 제어하기 위해 필요한 actuator와 controller가 controlled process의 feedback을 입력 받기 위한 sensor가 있다. Controller는 의사 결정을 내리기 위해 내부적으로 control algorithm과 process model을 가지며, 이 중 process model은 controller가 가지는 내부적인 믿음으로, 시스템 혹은 환경에 대한 정보나 제어하는 controlled process의 상태 등을 포함할 수 있다.

세 번째 단계에서는 두 번째 단계에서 식별한 control structure에서의 CA가 특정 상황 또는 환경에서 hazard를 초래할 때, 즉 unsafe control action (UCA)이 될 때 이를 식별한다. 어떤 상황 또는 환경에서 CA가 제공되는지에 따라 안전한지 아닌지가 결정되므로, UCA를 식별할 때는 context를 명시해야 한다. UCA에는 네 가지 유형이 있고, 각각은 다음과 같다: (1) CA를 제공하지 않는 것이 hazard를 유발한다; (2) CA를 제공하는 것이 hazard를 유발한다; (3) 잠재적으로 안전한 CA를 제공하나, 너무 일찍, 너무 늦게, 혹은 잘못된 순서로 CA를 제공한다; (4) (지속적으로 제공되는 CA의 경우) 너무 오래 지속되거나 너무 빨리 중지된다.

마지막 단계에서는 UCA의 발생 원인을 나타내는 loss (혹은 causal) scenario를 식별한다. Loss scenario는 크게 두 유형으로 나뉜다. 각 유형에서는 'UCA가 발생하는' 원인과 'CA가 잘못 실행되거나 실행되지 않아서 hazard가 발생하는' 원인에 집중한다. UCA가 발생하도록 하는 원인에는 네 가지가 있다. (물리적 controller일 때) controller의 고장, 부적절한 control algorithm, 다른 controller에서 제공받은 UCA, 그리고 부적절한 process model이 이에 해당한다. 다음으로 CA가 잘못 실행되거나 실행되지 않아서 hazard가 발생하는 원인으로는 actuator를 포함하는 control path에서의 문제와 실제로 control에 따라 동작하는 controlled process의 문제가 있다. Loss scenario 식별 시에는 각 항목에 대한 구체적인 causal factor를 기술하고, 이것이 발생하지 않도록 하는 후속 조치에 활용할 수 있도록 한다. 이렇게 도출된 loss scenario를 통해 시스템의 안전성을 확보하기 위한 추가 요구사항이 도출될 수 있으며, 시스템의 아키텍처를 도출하거나 테스트 케이스를 정의하는 등의 활동이 후속될 수 있다.

2.2 Quality Attribute Scenario (QAS)

Quality Attribute (QA)는 시스템을 둘러싼 이해관계자들의 요구 (needs) 중 가치를 창출할 수 있으며 측정 가능한 (measurable) 비기능적 속성을 의미하며, 그 예시로는 안전성 외에도 신뢰성 (reliability), 확장성 (scalability), 사용성 (usability) 등이 있다 **[10, 12, 13]**. 이런 QA에 대한 요구사항을 시나리오의 형태로 구체화한 것을 품질 특성 시나리오 (Quality Attribute Scenario, QAS) **[10]**라고 부른다. QA는 일반적으로 SW의 설계 단계에서 architecture를 설정하는 데에 영향을 미치는 가장 큰 요인으로, QAS를 통해 SW가 가져야 할 QA 관점의 요구사항을 나타낸다. QAS는 총 여섯 가지의 요소로 구성되며, 각각의 이름과 정의는 다음과 같다:

- Source: 사람, 다른 시스템 등 stimulus를 제공하는 개체
- Stimulus: source에 의해서 artifact에게 주어지는 이벤트
- Artifact: stimulus가 도착하는 대상 (시스템의 전체 또는 일부, 혹은 시스템의 집단)

- Response: stimulus가 도착한 이후 artifact의 동작
- Response measure: 테스트를 위한, response에 대한 측정 가능한 값
- Environment: 시스템의 state 등 시나리오가 발생하는 환경의 집합

QAS는 크게 어느 한 시스템에 특정되지 않는 general scenario와 시스템에 특정되는 concrete scenario로 구분한다. 몇몇 QA의 general scenario는 **[10]**에서 확인할 수 있다. 특정 시스템에 대하여 제공된 general scenario를 유용하게 사용하기 위해서는 해당 시스템에 적용 가능하도록 general scenario에서 제공하는 값을 수정해야 한다. Concrete scenario는 이를 위하여 general scenario에서 제공하는 값들을 시스템에 적용 가능하도록 수정하여 작성한, 실제 시스템에 대한 QAS이다. 본 연구에서는 STPA 결과를 활용하여 safety에 대한 general scenario를 설정하고, 테스트베드에 적용하여 concrete scenario를 식별하는 예시를 보인다.

3. 관련 연구

3.1 협업/협력 가상물리시스템의 안전성

[14]에서는 협력하는 (collaborative) CPS의 가변성 (variability)을 고려하여 기존의 여러 HA 기법 중 세 가지 기법, 이벤트 트리 분석 (Event Tree Analysis, ETA) **[15]**, 결함 트리 분석 (Fault Tree Analysis, FTA) **[16]**, 고장 모드 및 효과 분석 (Failure Mode and Effect Analysis, FMEA) **[17]**을 확장하는 방법을 제안하였다. **[14]**에서는 또한 여러 HA 기법을 통해 도출된 결함을 추적하기 위하여 variability를 고려한 결함 추적 그래프 (fault traceability graph, FTG)를 개발하였다. 추가로, 확장된 각 HA의 모델링과 결함 추적 그래프의 생성을 지원하는 도구까지 개발하였다. 이를 통해, ISO 21448 **[18]**에서 언급되는 known-safe 영역을 최대화하는 동시에 unknown-unsafe 영역을 최소화하고자 하였다.

[19]에서는 무선 통신을 이용한 Safe Cooperating CPS (SafeCOP) 프로젝트 **[20]**의 일부로 진행된 군집 주행 시스템에 대하여 진행한 연구를 나타냈다. 해당 연구에서는 안전성을 보장하기 위하여 설계 단계에서는 Goal Structuring Notation (GSN) **[21]**을 활용한 safety case를 사용하고, 런타임에는 런타임 관리자를 통하여 지속적으로 시스템의 동작에 대한 명세에 해당하는 contract의 위반을 확인하고자 하였다. **[19]**에 직접적으로 언급되어 있지는 않으나, 같은 프로젝트를 다루고 있는 **[20]**을 통해 contract의 여러 subset은 STAMP **[9]**를 활용해 도출한 것으로 이해할 수 있다.

3.2 System of Systems의 안전성

System of Systems (SoS) **[22]**는 기존의 시스템들이 각 구성 요소가 합쳐져 하나의 시스템으로써 동작할 수 있었던 것처럼, 공동의 목표를 달성하기 위해 여러 시스템들이 합쳐져 하나의 시스템으로써 동작하는 것을 의미한다. 그런 의미에서 협업하는 여러 CPS를 하나의 더 큰 시스템으로 바라본다면, SoS와 유사한 형태를 가진다고 볼 수 있다.

[23]에서는, ISO 26262에서 요구되는 위험 분석 및 리스크 평가 (HARA) 방법을 사용하여 SoS의 hazard를 식별하는 프로세스를 제안하고 채석장 자동화 환경을 사례 연구로 활용하였다. 채석장에서 작동되는 자율 주행 기계의 impact matrix를 생성, 제안한 프로세스를 통해 분석하여 잠재적 사고와 hazard를 분석하였다.

[24]에서는 HA 기법인 Hazard and Operability Study (HAZOP) **[25]**와 FTA를 활용하여 전기 채석장 내에서의 서로 다른 기계 사이의 emergent behavior를 다루는 것에 초점을 맞췄다. 우선적으로 HAZOP을 적용하여 hazard와 잠재적 영향을 식별한 후, 결과물을 FTA에서의 top event로 설정한 후 FTA를 적용하여 원치 않는 이벤트의 발생을 예방할 수 있는 방법을 도출하였다.

본 연구와 기존 연구의 공통점은, 궁극적으로 더 안전한 협업하는 CPS를 만들고자 한다는 것이다. 단, 그것을 달성하기 위한 접근 방법에는 차이가 있다. 앞서 3.1절과 3.2절에서 언급한 관련 연구들의 경우, 대부분 HA를 적용하여 hazard의 발생을 예방하는 데에 초점을 맞추었고, HA 기법의 적용 이후 안전성을 검증하는 내용을 다루는 연구는 찾아보기 어려웠다. 단, 기존에 단일 CPS의 안전성 검증에 대한 연구 **[26, 27]**는 존재한다. **[26]**의 경우 인공 신경망과 정형 기법 (formal method)을 활용해 검증을 진행했다는 점에서 테스팅을 목표로 하는 본 연구와는 다소 접근이 다르다. **[27]**은 자율운항 선박 (autonomous ship) 시스템을 대상으로 하여 safety verification, 그 중에서도 safety testing을 위한 test scenario의 작성에 대한 내용을 포함하며, 본 연구에서처럼 STPA를 작성하는 과정에서 활용하였다.

Safety testing은 safety verification 중 동적 분석 (dynamic analysis)의 한 방법으로, safety verification에는 이외에도 정적 분석 (static analysis)인 정형 검증 (formal verification)과 SW 수준에서 수행하는 FTA (SFTA) 등이 있다 **[8]**. HA는 safety verification을 수행할 때 시스템의 hazard와 safety-critical한 구성요소 또는 변수 등에 대한 기본적인 내용들을 제공해줄 수 있기 때문에, 사전에 수행되는 것이 권장된다. 또한, safety verification은 HA의 후속 활동이므로, 테스팅의 형태가 아니더라도 safety-critical 시스

템의 안전성의 검증은 시스템 검증 단계에서 수행되어야 한다. **[11]**에서는 STPA 수행 이후에 결과물인 loss scenario를 활용해 TC를 생성하고 테스트 계획을 수립할 수 있다고 설명한다. 본 논문에서는 검증 단계에서 시스템의 안전성을 검증하기 위해 활용 가능한 safety TC를 생성하는 방법을 제안하고자 한다.

4. STPA를 활용한 안전성 테스트 케이스 생성 프로세스

본 연구에서 제안하고자 하는 주된 내용은 'STPA를 활용하여 safety TC를 생성하는 프로세스'이다. 이때 1장에서 언급한 것처럼 QAS의 형태로 safety TC를 생성하며, 이를 위하여 STPA 결과를 활용할 수 있는 safety general scenario를 제시한다. 협업 CPS의 분석에서 STPA를 사용하는 이유는, STPA에서는 기존의 다른 HA 기법과는 달리 컴포넌트 간의 상호작용으로 인해 발생 가능한 hazard를 포함하기 때문이다. 이것을 협업 시스템 수준에서 적용한다면, STPA를 통해 시스템 간의 협업 시에 발생하는 상호작용을 다룰 수 있다.

제안하고자 하는 프로세스에 대한 기본 아이디어, 즉 STPA의 결과를 활용하고 QAS를 작성하여 안전성을 확보하는 데에 활용하는 것은 저자의 학위논문 **[28]**에서 처음 제시하였다. 해당 논문에서는 STPA의 결과를 활용하여 QAS를 도출하면, 이를 협업 CPS의 요구사항이나 설계 명세 등을 안전성 측면에서 보완하기 위해 활용할 수 있다고 주장하였다. 이를 위해 먼저 협업하는 각각의 CPS를 전체 CPS의 구성 시스템으로 간주하여 각 구성 시스템들에 대해 별도로 STPA를 수행하고, 이후 협업으로 인해 추가되는 내용들을 고려한 전체 CPS에 대한 STPA를 추가로 진행하였다. QAS 식별을 위한 프로세스도 제안하였는데, 당시에는 단순히 STPA의 결과물인 UCA와 loss scenario를 어떻게 활용하여 QAS를 식별할 수 있는지에 대한 과정을 언급하였다. 또한, QAS의 각 항목으로 선택할 수 있는 STPA 결과 값들을 제안하고, 제안한 값들을 활용하여 QAS를 식별하는 예시를 보였다. 이때 QAS의 식별을 돕기 위한 guideword도 함께 제시하였는데, 당시에 제시한 guideword는 'without failure, 'until (operation), 'within (time)', 'after (operation)'의 네 가지로, QAS의 response measure로 활용할 수 있다고 설명하였다. 이후 추가 연구를 진행함으로써 기존의 아이디어를 발전시키고 제안하는 프로세스를 다듬어 본 논문을 통해 제안한다. 제안하는 프로세스를 통해 STPA의 결과물과 제공되는 guideword를 활용하여 QAS의 형태로 safety TC를 생성할 수 있으며, 생성한 TC를 통해 safety testing을 진행할 수 있다.

본 논문에서 제안하는 Safety TC를 생성하는 프로세스는 다음과 같다. 먼저, TC 생성에 앞서 일반적인 STPA 프로세스를 따라 대상 시스템을 분석한다. 이후, STPA 결과물을 활용하여 safety TC를 생성하는 프로세스를 따른다. Safety TC를 생성하는 프로세스는 2.1절에서 언급된 STPA의 프로세스와 함께 다음의 <그림 1>에서 확인할 수 있다.

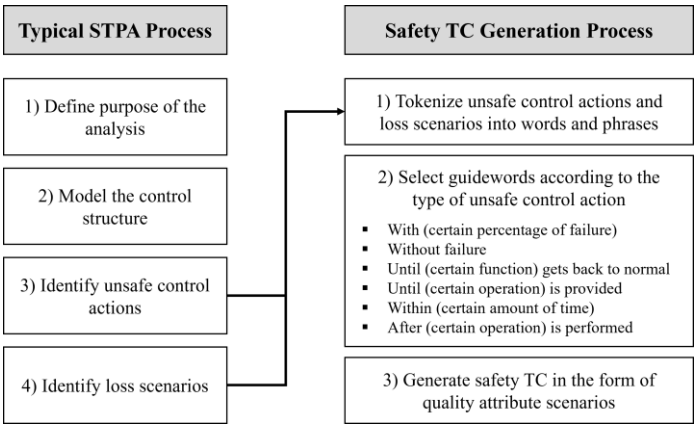

그림 1 STPA를 활용한 안전성 테스트 케이스 생성 프로세스

먼저, STPA의 결과물인 UCA와 loss scenario를 필요에 따라 단어 혹은 절의 형태로 토큰화 (tokenize)한다. 이후, <그림 1>에 나타낸 guideword 중 적절한 것을 선택한다. 선택한 guideword는 response measure로 활용된다. 마지막으로, 토큰화한 STPA 결과물과 선택한 guideword를 활용하여 safety TC를 생성한다. 제안하는 프로세스를 따라 생성한 safety TC는 STPA의 결과로써 도출된 각각의 loss scenario에 대응되어야 하고, 그 형태는 QAS의 형태를 따른다.

<그림 1> 우측의 Safety TC Generation Process의 두 번째 단계에서 제시된 guideword들은 앞서 설명한 **[28]**에서 제시하였던 guideword에서 추가 연구를 거쳐 작성되었으며, 추후 연구를 통해 추가될 수 있다. STPA 결과를 활용하여 QAS의 형태로 safety TC를 생성할 때 사용 가능한 값, 즉 safety general scenario는 <표 1>에서 확인할 수 있다.

5. 사례 연구

5.1 테스트베드 시스템

본 논문에서 제안한 approach를 적용하는 대상은 기존에 개발한 테스트베드 시스템으로, 간단한 지능형 도로교통시스템 (Intelligent Transportation System) **[29]**의 예시로써 활용된다. 해당 시스템은 내부에 직교하는 도로를 포함하여 총 다섯 개의 교차로를 가지는 루프 형태의

표 1 STPA 결과를 활용하여 QAS의 형태로 safety TC를 생성할 때 활용 가능한 safety general scenario

항목	사용 가능한 값
Source	▪ Operator ▪ Sensor ▪ I/O device ▪ Controller ▪ (internal/external) Data source
Stimulus	Source가 제공하는 데이터 혹은 제어명령
Environment	([10]에서 제공하는 safety general scenario) ▪ In normal operation ▪ In degraded operation ▪ In manual operation ▪ In recovery mode
Artifact	▪ Controller ▪ Controlled process ▪ I/O device
Response	토큰화한 STPA 결과 중 동작에 대한 부분의 내용에 반대되는 내용 또는 아래에 기술된 것처럼 [10]에서 제공하는 safety general scenario의 response 항목에 대한 값 중에서 선택한다. Unsafe state를 인식하고 다음 중 하나 이상을 따른다: ▪ Avoid the unsafe state ▪ Recover ▪ Continue in degraded or safe mode ▪ Shut down ▪ Switch to manual operation ▪ Switch to a backup system ▪ Notify appropriate entities (people or systems) ▪ Log the unsafe state (and the response to it)
Response measure	식별하는 사람의 판단에 따른 적절한 guideword 사용

가상의 도로 상에서 동작하며, 내부에는 통신 모듈을 통해 동작하는 두 종류의 간단한 CPS가 협업하여 '도로교통시스템을 외부 환경에 맞춰 적응형으로 제어한다'는 공동의 목표를 달성하기 위해 존재한다. 각 시스템의 이름은 '교통정보시스템'과 '신호체계관제시스템'으로 설정하였다. 테스트베드 시스템이 동작하는 가상의 도로 환경을 포함하는 전체 시스템의 모습은 <그림 2>를 통해 나타냈다.

교통정보시스템은 각 교차로에 배치되어 도로의 환경

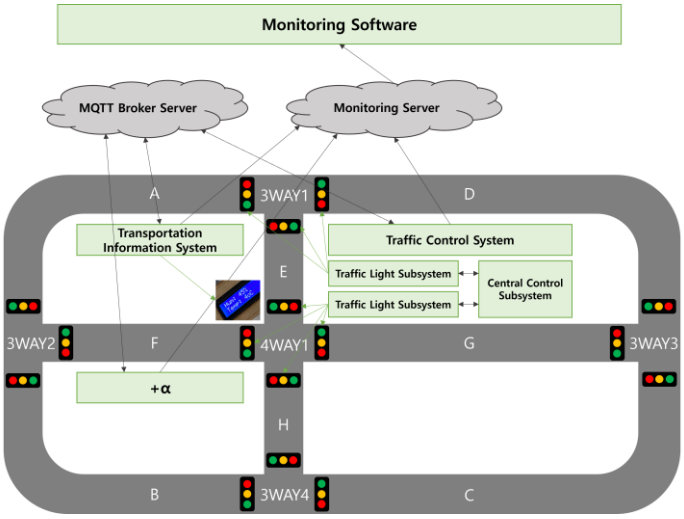

그림 2 테스트베드 시스템의 구조

정보를 수집하고, 도로 상에 화재가 발생하거나 도로가 침수되는 등의 긴급 상황이 발생하면 이를 알린다. 이외에도 우회 도로를 안내하거나, 정체 상황을 알릴 수 있고, 긴급 상황이 발생하지 않으면 날씨 정보를 디스플레이 모듈에 출력한다. 총 5개의 서로 다른 인스턴스가 각 교차로에 배치되며, 개별 시스템처럼 상호작용하며 동작한다.

신호체계관제시스템은 두 종류의 서브시스템으로 구성되어 있으며, 각각은 중앙통제 서브시스템과 신호등 서브시스템으로 설정하였다. 중앙통제 서브시스템은 신호등 서브시스템을 필요에 따라 제어하기 위한 시스템으로, 관리자의 입력 혹은 외부 시스템의 입력에 따라 신호등 서브시스템을 제어한다. 추후 다른 시스템이 테스트베드 시스템에 편입될 수 있으나, 여기서의 외부 시스템은 현재까지는 교통정보시스템만 해당된다. 신호등 서브시스템은 중앙통제 서브시스템으로부터 별도의 명령이 입력되지 않는 경우 내부 규칙을 따라 신호등의 점등 및 소등을 제어하여 교통 흐름을 통제한다. 신호등 서브시스템은 교통정보시스템과 마찬가지로 가상의 도로의 서로 다른 5개 교차로에 배치된다. 각 교차로마다 서로 다른 인스턴스가 배치되므로, 동일한 요구사항을 가지는 시스템이지만 서로 다른 시스템처럼 개별적으로 중앙통제 서브시스템과 상호작용하며 동작한다.

각 시스템은 아두이노 보드를 중심으로 통신 모듈, 센서 모듈, 디스플레이 모듈 등 동작에 필요한 모듈이 물리적으로 연결되어 있으며, 아두이노 보드에 탑재된 SW 컨트롤러를 통해 각 모듈을 제어한다. 이외에도 두 시스템의 정보 교환을 위한 MQTT (Message Queuing Telemetry Transport) 프로토콜 [30]을 따르는 브로커 서버를 포함하며, 관리자의 모니터링을 위한 모니터링 서버와 SW로 구성된 모니터링 시스템도 추가로 구현되어 있다.

해당 시스템은 [28]에서도 사례 연구 대상으로 활용하

였으며, 당시의 연구 결과를 활용하여 작성되었으므로 본 논문에서 사용하는 시스템의 명칭이나 내부 구성에 있어 다소 차이는 있으나, 본질적으로는 동일한 목적을 가진 시스템이다. 단, 다음 5.2절에서 후술되는 STPA의 적용이나 safety TC의 생성에 관한 내용에는 차이가 있다.

5.2 안전성 테스트 케이스의 생성

제안한 프로세스를 테스트베드 시스템에 적용하기에 앞서, STPA 프로세스를 따라 테스트베드 시스템을 분석하였다. STPA를 수행하여 도출된 결과인 UCA 중 일부는 다음과 같다. 예시로 나타낸 UCA는 관리자의 개입에 의한 시스템의 동작과 서로 다른 종류의 두 CPS의 협업을 위한 상호작용을 확인할 수 있는 내용을 포함한다. 즉, 서로 다른 CPS 간의 협업을 고려하지 않았다면 드러나지 않았을 UCA를 다루고 있다.

- UCA-13: 관리자가 교통정보시스템이 MQTT 서버를 통해 긴급 정보를 제공하지 않는 경우에 중앙통제 서브시스템에 수동 제어 명령을 제공하지 않는다.
- UCA-19: 중앙통제 서브시스템의 SW 컨트롤러가 교통정보시스템으로부터 MQTT 서버를 통해 긴급 정보가 제공된 경우에도 RF (Radio Frequency) 모듈에 제어 명령을 제공하지 않는다.
- UCA-30: 신호등 서브시스템의 SW 컨트롤러가 중앙통제 서브시스템으로부터 수동 신호 제어 명령을 제공받은 경우에도 수동 신호등 제어 명령을 제공하지 않는다.

위 예시 중 UCA-13은 관리자가 개입하여 시스템을 제어하는 상황을 다루고 있다. 관련된 hazard는 ‘WiFi 통신 오류’와 ‘제어 명령의 누락’이며, 이 hazard는 인명 피해, 도로 인프라 및 시스템 인프라의 손실 또는 손상이라는 loss를 초래할 수 있다. UCA-19는 교통정보시스템과 신호체계관제시스템의 중앙통제 서브시스템 사이의 상호작용을 다루고 있으며, 이와 관련된 hazard는 ‘시스템의 제어권 상실’이다. UCA-30은 신호체계관제시스템 내부의 두 서브시스템의 상호작용을 다루고 있으며, 이에 관련된 hazard는 ‘RF 통신 오류’와 ‘신호체계 오작동’이다. UCA-19와 UCA-30에 관련된 세 가지 hazard도 위와 동일한 loss를 초래할 수 있다. 각각에 관련된 loss scenario 중 일부는 다음과 같다.

- LS13-4: 교통정보시스템이 올바르게 동작하지 못하는 경우에도, 중앙통제 서브시스템의 SW 컨트롤러가 관리자로부터 올바른 수동 제어 명령을 입력 받았으나 이를 처리하여 전달하는 동작을 수행하지 못하였다.
- LS19-2: 도로 상에 긴급 상황이 발생했음에도 해당 상황에 대한 데이터를 제공받지 못했기 때문에 교통정보시스템이 긴급 정보를 제공하지 않아서 중앙통제 서브시스템이 잘못된 process model 값을 가지게 되었다. 긴급 상황에 대한 데이터를 교통정보시스템이 제공받지 못한 것은 센서 기능의 저하로 인해 정상적인 데이터를 수집하지 못했기 때문이다.
- LS30-1: 신호등 서브시스템의 SW 컨트롤러가 잘못된 control algorithm을 가지고 있어서 중앙통제 서브시스템으로부터 수동 신호 제어 명령을 제공받았으나 수동 신호등 제어 명령을 제공하는 데에 실패하였다.

위의 loss scenario는 STPA handbook **[11]**에 제시된 causal factor의 예시를 따라 분석되었다. 이 중 LS19-2의 경우 교통정보시스템으로부터 잘못된 정보를 제공받아 중앙통제 서브시스템이 잘못된 process model을 가지게 되었기 때문에 UCA-19가 발생했다고 분석하였다. 따라서 LS19-2는 교통정보시스템에 발생한 문제를 중점적으로 다룬다. 경우에 따라서는 동일한 loss scenario 유형에서도 여러 가지 causal factor가 발견될 수 있으므로, 위의 넘버링은 하나의 UCA에 대해 여러 가지 loss scenario가 도출될 수 있음을 보이는 수단 정도로 이해해야 한다. 제안한 프로세스를 따라 위의 loss scenario에 대한 각 예시에 대응하는 QAS 형태의 safety TC를 생성한 예시는 다음과 같다.

- TC13-4: 관리자는 교통정보시스템이 올바르게 동작하지 못하는 경우 중앙통제 서브시스템을 수동 제어한다. 그러면 중앙통제 서브시스템의 SW 컨트롤러는 교통정보시스템이 올바르게 동작할 수 있도록 복구될 때까지 (until system function gets back to normal) 관리자가 제공한 수동 제어 명령을 받아들이고 그에 맞게 동작해야 한다.
- TC19-2: 센서 기능의 저하로 인해 시스템 전체의 기능이 저하된 상황에서 센서는 긴급 정보와 일치하지 않는 잘못된 센서 데이터를 제공한다. 그러면 교통정보시스템의 SW 컨트롤러는 센서의 기능이 정상적으로 돌아올 수 있을 때까지 (until sensor function gets back to normal) 시스템의 모든 데이터에 대한 로그를 남기고 관리자가 이를 확인할 수 있도록 모니터링 서버로 전송해야 한다.
- TC30-1: 정상 동작 시에, 신호등 서브시스템의 RF 모듈은 수동 신호 제어 명령을 SW 컨트롤러에 전달한다. 그러면 신호등 서브시스템의 SW 컨트롤러는 실패하지 않고 (without failure) 수동 신호등 제어 명령을 제공해야 한다.

각 safety TC의 구성을 살펴보면 <표 1>에 나타낸 각 항목에 대해 '사용 가능한 값'이 활용되었음을 알 수 있다. 이를 직관적으로 확인하기 위해 QAS를 나타내기 위한 다이어그램을 활용하면 <그림 3>과 같이 나타낼 수 있다. TC13-4는 (a)로, TC19-2는 (b)로, TC30-1은 (c)로 나타냈다.

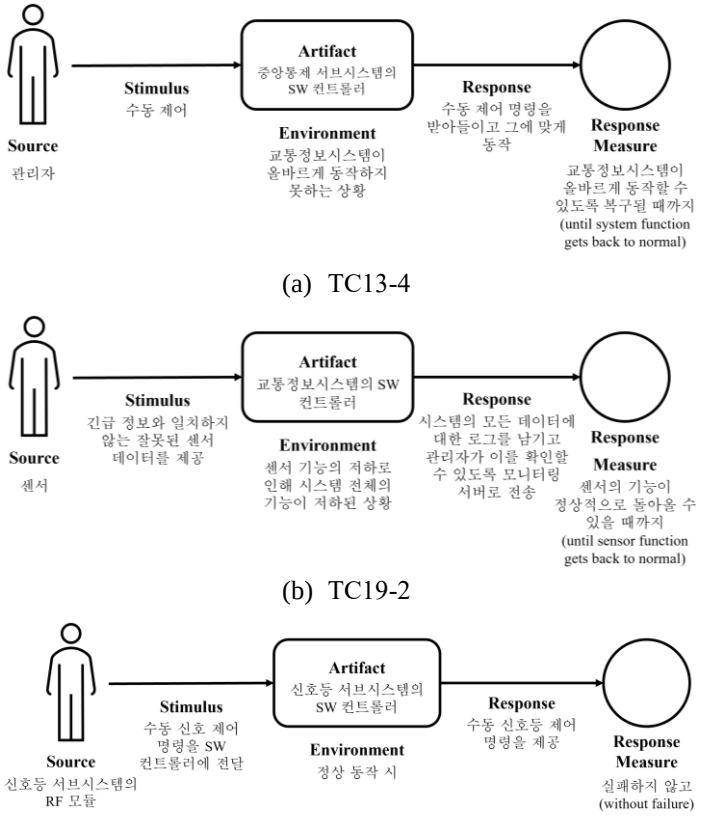

그림 3 QAS의 형태로 나타낸 safety TC의 예시

5.3 평가 및 논의

테스트베드 시스템에 제안한 프로세스를 적용하여 총 43 개의 UCA, 110 개의 loss scenario를 식별하고, 각 loss scenario에 상응하는 safety TC를 110 개 생성하였다. 생성한 safety TC를 활용해 테스팅을 진행한 결과, 테스트베드 시스템은 생성한 모든 safety TC를 통과하였다. 단, STPA 결과에 의존하여 safety TC를 생성하고 테스팅을 수행하였으므로, STPA의 수행 정도에 따라 식별되는 UCA, loss scenario, 그리고 그에 따른 safety TC의 개수와 통과율이 달라질 수 있음을 인지해야 한다. 그리고 만일 safety TC를 통과하지 못했다면, 어떤 원인으로 인해 통과하지 못했는지를 분석하고 해당 원인이 이후 시스템의 안전성을 위협할 수 있다고 판단될 경우 시스템의 요구사항을 수정하고 이에 맞게 시스템을 개선하는 과정을 거치는 것이 필요하다.

제안한 프로세스의 장점은 다음과 같다. 첫 번째는, 체계적인 HA 기법으로 알려진 STPA의 결과를 활용하고 안전성과 같은 시스템의 Quality Attribute에 대한 요구사항을 체계적으로 기술하기 위한 QAS의 형태를 빌려 safety TC를 생성하기 때문에 보다 체계적인 접근이 가능하다는 점이다. 두 번째는, STPA를 활용하기 때문에 loss scenario에 포함된 causal factor를 추후 시스템에 행해져야 할 점검에 대한 항목으로 고려할 수 있다는 점이다. 예를 들어, CPS를 제어하는 SW 컨트롤러가 올바르게 동작하는지에 대한 TC를 생성했을 때, 이에 관련된 loss scenario의 causal factor가 SW 컨트롤러가 잘못된 control algorithm을 가지고 있는 경우를 나타낼 수 있다. 만일 해당 TC를 통과하지 못했다면, control algorithm이 잘못되었는지 점검하는 등의 조치를 취할 수 있다. 마지막으로 여러 표준 **[5, 6]**에서 강조하는 기능 안전성을 포함하여 제어 관점에서의 안전성 (control safety), 물리적 동작의 안전성 (physical safety) 등 안전성의 다양한 측면 **[31]**을 고려할 수 있다는 장점이 있다.

단, 본 논문에서 제안한 프로세스의 가장 주요한 한계점은, 제안한 프로세스를 통해서 생성하는 safety TC가 다소 전형적일 수 있다는 것이다. 다르게 말하자면, 매우 드물게 발생하는 실패 (failure)의 경우에는 다루지 못할 수 있다는 것이다. 제안한 프로세스 외적으로는, 사례 연구를 진행하는 테스트베드 시스템의 상호작용의 형태와 그 동작이 단순하다는 한계점이 있다. 또한, 테스트베드 시스템이 지니는 특성 중 동일한 시스템의 여러 인스턴스가 존재한다는 특성에 대하여 다루지 못했다는 한계점이 있다.

단, TC가 전형적이라고 해서 그 TC가 의미 없는 TC가 되는 것은 아니다. 의미 없어 보이는 TC라고 해도, 해당 TC가 다루는 부분에 대한 테스팅을 진행하고 통과 여부를 확인하는 것은 필요한 과정이다. 또한, 테스팅을 통해서 모든 경우를 완벽하게 다루는 것은 거의 불가능에 가깝다. 이것은 테스팅의 근본적인 한계점이므로, 더 철저한 검증이 필요하다면 테스팅 이외의 검증의 수단을 추가로 마련하여 이를 보완할 수 있도록 하는 것이 필요하다.

6. 결론 및 향후 연구

본 논문에서는 STPA 수행 결과를 활용하여 QAS의 형태로 시스템의 안전성을 테스트하기 위한 safety TC를 식별하는 방법을 제안하고 사례 연구를 수행하였다. 사례 연구는 기존에 개발한 테스트베드 시스템을 대상으로 수행되었으며, 제안한 approach의 적용을 통하여 safety TC를 식별하고, safety testing까지 수행하여 결과를 살펴보았다. 또한, Safety testing의 수행 결과를 통해 제안한

approach의 적용 효과를 살펴보았다.

추후 사례 연구를 진행하는 테스트베드 시스템의 상호작용에 대한 복잡도를 증가시키고 기능적 단순함을 극복하기 위하여 기존의 테스트베드 시스템에 존재하는 CPS들과는 다른, 새로운 시스템을 추가할 계획이다. 또한, 테스트베드 시스템처럼 한 시스템의 여러 인스턴스가 존재하고 상호작용하는 경우에 대하여 다룰 수 있도록 제안한 프로세스를 발전시키고자 한다. 이를 통해 앞서 제시한 guideword 이외에 추가적인 guideword를 제시하는 데에도 도움이 될 것이라고 예상된다. 뿐만 아니라, STPA 이외에 다른 HA 기법의 추가적인 적용 또한 고려 중이다.

Acknowledgment

이 논문은 2023년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (NRF-2021R1F1A1047246). 또한 저자의 석사학위 논문 **[28]**에 기반하였음.

참고문헌

[1] R. Baheti, H. Gill, “Cyber-Physical Systems,” The Impact of Control Technology, vol. 12, no. 1, pp. 161-166, 2011.

[2] L. Zhang, Y. P. Fallah, R. Jihene, “Cyber-Physical Systems: Computation, Communication, and Control,” International Journal of Distributed Sensor Networks, vol. 9, no. 2, 2013.

[3] P. B. Checkland, “Systems Thinking, Systems Practice,” John Wiley, 1999.

[4] Department of Defense (DOD), MIL-STD-882E: System Safety, 2012.

[5] International Electrotechnical Commission (IEC), IEC 61508: Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems, 2010.

[6] International Organization of Standardization (ISO), ISO 26262: Road Vehicles – Functional Safety, 2011.

[7] Radio Technical Commission for Aeronautics (RTCA), DO-178C: Software Considerations in Airborne Systems and Equipment Certification, 1992.

[8] N. G. Leveson, “Safeware: System Safety and Computers,” Addison-Wesley, 1995.

[9] N. G. Leveson, “Engineering a Safer World: Systems Thinking Applied to Safety,” MIT Press, 2016.

[10] L. Bass, P. Clements, R. Kazman, “Software Architecture in Practice,” Fourth Edition, Addison-Wesley Professional, 2021.

[11] N. G. Leveson, J. P. Thomas, “STPA Handbook,” 2018, [Online]. Available: http://psas.scripts.mit.edu/home/materials/

[12] N. Rozanski, E. Woods, “Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives,” Second Edition, Addison-Wesley Professional, 2011.

[13] International Organization for Standardization (ISO), ISO/IEC 25010: Systems and Software Engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE) – System and Software Quality Models, 2011.

[14] N. Ali, M. Hussain, J-E. Hong, “Analyzing Safety of Collaborative Cyber-Physical Systems Considering Variability,” IEEE Access, vol. 8, 2020.

[15] M. Cepin, “Event Tree Analysis,” Assessment of Power System Reliability: Methods and Applications, pp. 89-99, 2011.

[16] C. A. Ericson II, “Fault Tree Analysis,” System Safety Conference, vol. 1, pp. 1-9, 1999.

[17] D. J. Reifer, “Software Failure Modes and Effects Analysis,” IEEE Transactions on reliability, vol. R-28, no. 3, pp. 247–249, 1979.

[18] International Organization of Standardization (ISO), ISO 21448: Road vehicles – Safety of the Intended Functionality, 2019.

[19] S. Medawar, D. Scholle, I. Sljivo, “Cooperative Safety Critical CPS Platooning in SafeCOP,” 2017 6th Mediterranean Conference on Embedded Computing (MECO), pp. 1-5, 2017.

[20] P. Pop, et al., “The SafeCOP ECSEL Project: Safe Cooperating Cyber-Physical Systems Using Wireless Communication,” 2016 Euromicro Conference on Digital System Design (DSD), pp. 532-538, 2016.

[21] T. Kelly, R. Weaver, “The Goal Structuring Notation – A Safety Argument Notation,” Proceedings of the Dependable Systems and Networks 2004 Workshop on Assurance Cases, vol. 6, 2004.

[22] R. L. Ackoff, “Towards a System of Systems Concepts,” Management Science, vol. 17, no. 11, pp. 661-671, 1971.

[23] S. Baumgart, J. Fröberg, S. Punnekkat, “Analyzing Hazards in System-of-Systems: Described in a Quarry Site Automation Context,” 2017 Annual IEEE International Systems Conference (SysCon), pp. 1-8, 2017.

[24] F. U. Muram, M. A. Javed, S. Punnekkat, “System of Systems Hazard Analysis Using HAZOP and FTA for Advanced Quarry Production,” 2019 4th International Conference on System Reliability and Safety (ICSRS), pp. 394-401, 2019.

[25] T. C. McKelvey, “How to Improve the Effectiveness of Hazard and Operability Analysis,” IEEE Transactions on Reliability, vol. 37, no. 2, pp. 167-170, 1988.

[26] H-D. Tran, et al., “Safety Verification of Cyber-Physical Systems with Reinforcement Learning Control,” ACM Transactions on Embedded Computing Systems (TECS), vol. 18, no. 5s, pp. 1-22, 2019.

[27] B. Rokseth, O. I. Haugen, I. B. Utne, “Safety Verification

for Autonomous Ships," MATEC Web of Conferences, vol. 273, pp. 02002, 2019.
[28] Yoona Heo, "A Study on Identification of Quality Attribute Scenario for Safety of Cooperating Cyber-Physical Systems Using Systems-Theoretic Process Analysis," Master's thesis, Konkuk University, 2023.
[29] G. Dimitrakopoulos, P. Demestichas, "Intelligent Transportation Systems," IEEE Vehicular Technology Magazine, vol. 5, no. 1, pp. 77-84, 2010.
[30] R. A. Light, "Mosquitto: Server and Client Implementation of the MQTT Protocol," Journal of Open Source Software, vol. 2, no. 13, pp. 265, 2017.
[31] X. Lyu, Y. Ding, S-H. Yang, "Safety and Security Risk Assessment in Cyber-Physical Systems," IET Cyber-Physical Systems: Theory & Applications, vol. 4, no. 3, pp. 221-232, 2019.