2020 졸업 프로젝트 2

KUtoKit

요구사항명세(SRS)

201510436 허윤아

201611261 민지호

201611293 전다윤

201614158 장다혜

201710515 최연지

목차

1. Introduction

1.2 Purpose

과거에는 어떤 시스템에 문제가 발생할 때, 그것을 각 component 의 문제로 간주하여 문제를 해결하고자 했었다. 하지만, 시스템이 점점 복잡해지고 규모가 커지면서, 더 이상 모든 문제의 원인이 component 에 있다고 단정짓기가 어려워졌고, 잠재적으로 어떤 문제가 발생 가능한지를 미리 예측해보는 것도 어려워졌다. 또한, 시스템의 각 component 그 자체에는 아무런 문제가 없음에도 이들이 상호작용을 하는 과정 속에서 문제가 발생하기도 했다. 즉, 시스템의 각 component 들 자체로써 문제의 원인이 되는 것이 아니라, 시스템이 외부의 다른 요소들 또는 시스템 내부의 component 들과 상호작용하면서 문제를 발생시킬 수 있다는 것을 알게 되었다.

위와 같이 생기는 문제들을 해결하기 위해서는 기존의 것과 다른 관점을 가진 새로운 위험 분석 기법이 필요했고, 2012 년 MIT 의 Nancy G. Leveson 교수가 STAMP(System Theoretic Accident Model and Process)라는 시스템 이론에 기반한 사고 분석 모델 및 프로세스를 발표했고, 이에 기반하여 STPA(System Theoretic Process Analysis)라는 위험 분석 기법을 제시했다. STAMP 와 STPA 에서는 시스템 내부의 component 들 간의 control problem 또는 시스템 외적인 다른 요소들과 시스템 사이의 상호작용에 문제의 원인이 있다고 보고, control 의 관점에서 분석을 수행한다.

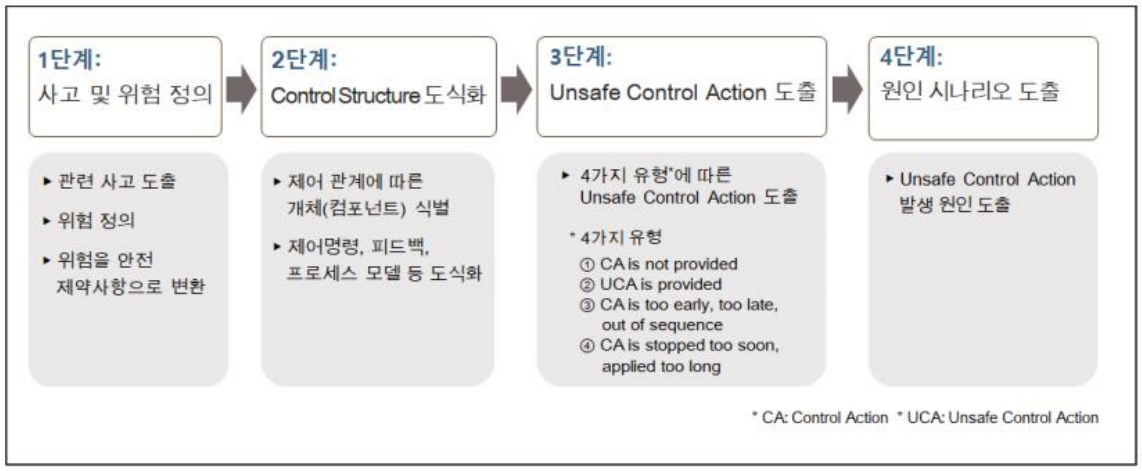
STPA 는 현재 다양한 분야에서 적용되고 있다. 특히 문제가 발생할 경우, 단순한 사고를 넘어서서 막대한 재산 피해나 인명 피해가 발생할 수 있는 critical system 을 사용하는 산업분야—자동차, 항공기, 원자력 발전소 등—에서 위험을 분석할 때 많이 사용되고 있다. 단, STPA 는 이제까지 전부 전문가에게만 의지해야 했기 때문에 STPA 를 수행할 수 있는 인원이 한정적이었고, 많은 시간과 노력을 필요로 했으며, 이를 지원해줄 만한 도구들—이 도구들은 MIT STAMP Workshop 홈페이지의 STAMP Tools 탭(https://psas.scripts.mit.edu/home/stamp-tools/)에서 확인할 수 있다—이 여럿 개발되었으나, 제한적이다. STPA 의 control structure 작성에서부터 UCA 및 cause 확인까지 분석에 많은 노력과 비용이 필요하다. 이를 좀 더

효율적으로 수행하고자 소프트웨어 컨트롤러 시스템의 formal SW specification 을 활용하여 STPA 의 수행을 지원하는 방법을 제안하고 이를 일부 자동화한 도구를 개발하고자 한다.

이 문서는 controller system 에 대한 STPA 의 적용을 돕기 위한 SW 를 구현하기 위한 요구사항을 명세한 문서이다. 프로그램을 개발하는 데에 있어, 좀 더 체계적으로 접근하여 waterfall model 에 가깝게 개발하기 위해 기본적으로 SASD 의 형식에 따라 본 문서를 작성한다. 단, 해당 SW 는 UI 를 위해 java 로 구현될 예정이므로 Use Case 등 일부는 OOAD 의 형식을 따른다. 팀 개발에서 다같이 협업하는 것에 도움을 주기 위해 IEEE standard 에 맞게 문서를 작성할 것이다.

1.3 Scope

이 문서에서 요구사항들을 서술하게 될 프로그램인 STPA Supporting Tool 은 Controller System 을 STAMP 기반 프로그램 분석 기법인 STPA 에 맞춰서 효율적으로 분석에 도움을 주는 프로그램이다.



|Fig. 1| STPA 절차

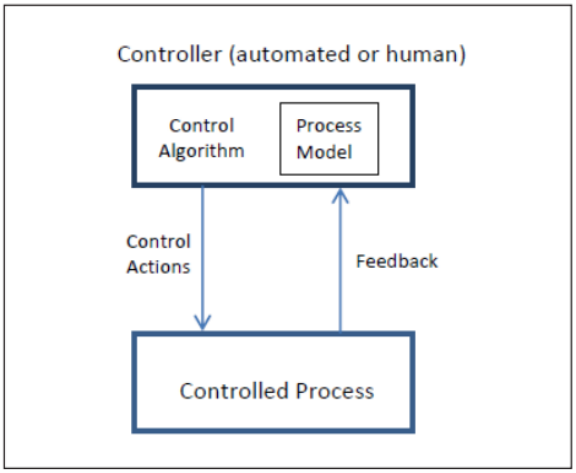
STPA 의 과정은 |Fig. 1|에서 보여지는 바와 같이 크게 4 단계로 나눌 수 있다.

1 단계: STPA 에서는 우선적으로 시스템에 발생할 수 있는 accident(loss)를 먼저 분석하고, 이를 발생시킬 가능성이 있는 hazard 와, 각 hazard 에 대한 System-level Safety Constraint 를 작성한다. 각각의 hazard 에는 해당 hazard 로 인해 발생 가능한 accident 가, System-level Safety

Constraint 에는 각각에 해당하는 hazard 가 마지막에 [index]로 포함되어 traceability 를 보장해줄 수 있어야 한다.

이 때 Loss 를 분석하는 이유는, STAMP 이론에서의 loss 가 negative environmental condition 과 hazard 의 combination 이기 때문이다. 단, negative environmental condition 이 발생하지 않도록 조절하는 것이 불가능하기 때문에, hazard 가 발생하지 않도록 하여 loss 를 줄이는 것이 가장 기본이 된다. 일반적으로 STPA 가 critical system 을 분석하는 데에 사용되는 것을 고려한다면, loss 를 줄이는 것이 가장 중요한 과제라는 점도 이해할 수 있을 것이다.

2 단계 : Control Structure 를 작성하는데, Control Structure 는 시스템을 추상화하여 모델링한 것으로 시스템의 Controller 와 Controlled Process, 그리고 Control Action 과 Feedback 으로 구성되어 있다. 먼저, Control 의 주체가 되는 Controller 에는 Control algorithm 과 Process Model 이 포함되어 있는데, Control algorithm 은 Controlled Process 를 Control 하기 위한 내부 알고리즘이고, Process Model 은 Controller 가 Controlled Process 들을 제어하기 위한 Control Action 을 제공하기 위해 필요한 정보들이다. 가장 일반적인 구조의 Control Loop 은 |Fig. 2|를 참고한다. Control Structure 는 가장 추상적이고 단순한 형태의 Control Loop 에서 시작하여 구체화 과정을 거치면서 복잡하게 나타낼 수 있다.



|Fig. 2| Control Loop

Control Structure 를 작성하면서 각 Controller 의 responsibility 를 정의해야 하는데, 이는 앞서 분석한 System-level Safety Constraint 를 지키기 위한 각 Controller 의 responsibility 와 역할을

의미한다. 전체적인 Control Structure 의 구조는 |Fig. 2| 에서와 같이 단순하지 않을 수 있고, 여러 개의 Controller 와 Controlled Process 가 다양하게 엮여 있을 수 있다.

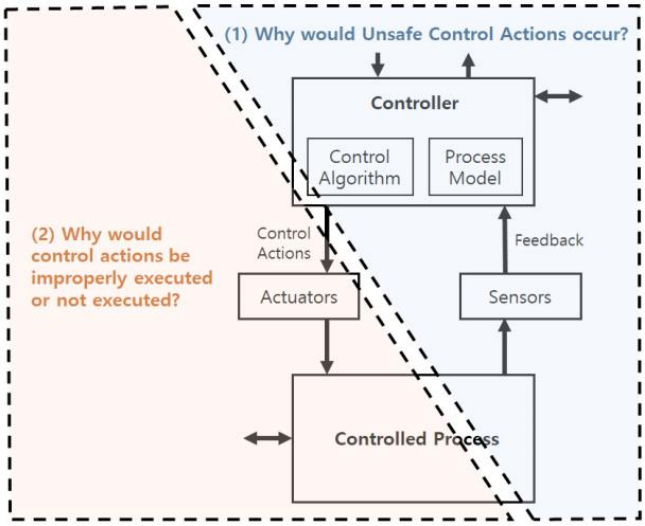
3 단계 : 앞서 작성된 Control Structure 에서 도출된 Control Action 들을 기반으로 UCA(Unsafe Control Action)을 작성한다. UCA 를 작성할 때에는 Control Action 을 제공할 때에 해당되는 UCA 의 유형과 Control Action 이 실행되는 조건인 Context 가 필요하다. UCA 의 유형은 크게 4 가지로 나눌 수 있다.

유형(Type)	설명
Not Providing Causes Hazard	(Controller가) Control Action을 제공하지 않아서 위험이 발생될 수 있음
Providing Causes Hazard	(Controller가) Control Action을 제공하여 위험이 발생될 수 있음
Too Late, Too Soon, Out of order	(Controller가) Control Action을 제공하였으나, 너무 늦게 또는 너무 빨리, 또는 잘못된 순서로 제공하여 위험이 발생될 수 있음
Stopped Too Soon / Applied Too Long	(Controller가) Control Action을 너무 이른 시점에 제공이 종료되거나 너무 오랫동안 제공하여 위험이 발생될 수 있음

|Fig. 3| UCA 의 4 가지 유형

Context 의 경우 2 단계에서의 Process Model 과 밀접한 연관이 있는데, Process Model 을 통해서 Context 를 작성할 수도 있고, 반대로 Context 로부터 Process Model 을 정의할 수도 있다. 각각의 UCA 는 1 단계에서 도출한 hazard 가 마찬가지로 [index]로서 포함되어 있어 traceability 를 보장할 수 있어야 한다.

4 단계 : 3 단계에서 도출해낸 UCA 를 기반으로 UCA 가 왜 발생하는지, 그 원인들을 분석하여 Causal Scenario 를 작성한다. Causal Scenario 는 도출 과정에 따라 크게 두 가지 유형으로 구분할 수 있는데, 아래의 |Fig. 4|에서와 같이 (1) 왜 UCA 가 발생하였는가 (2) 왜 Control Action 이 부적절하게 수행되거나 수행되지 않았는가 로 구분할 수 있다.



|Fig. 4| Causal Scenario 유형

(1) UCA 가 발생하는 이유는 Controller 에서 찾아볼 수 있다. Controller 가 UCA 를 제공하는 원인은 Controller 그 자체에 있을 수도 있고, Controller 가 받는 Feedback 에 있을 수도 있다. Control Algorithm 또는 Process Model 이 부적절하거나 Controller 자체의 고장 등의 경우는 전자로, Sensor 나 Controlled Process 로부터 받는 부적절한 정보, 또는 필요한 정보의 부재 등의 경우는 후자로 생각할 수 있다.

(2) Control Action 이 부적절하게 수행되거나 수행되지 않은 이유는 Control Path 와 Controlled Process 에서 찾아볼 수 있다. Control Path 는 대부분 간단한 Actuator 로 구성되어 있는데, 통신 지연 또는 단절, 전송 오류, Actuator 자체의 물리적 오류, 전원 문제, 동작 상의 부정확성, 성능 저하 등이 원인이 될 수 있고, Controlled Process 차원에서는 Process Input 의 손실 또는 잘못된 Input 값의 입력, environment 문제, 다른 component 의 오류, Process 그 자체의 오류, 여러 Controller 간의 명령 충돌 등이 원인이 될 수 있다.

이러한 원인들이 각각 Causal Factor 가 되어, 이를 통해 Causal Scenario 를 도출해낼 수 있다. 각 과정에서는 traceability 를 보장하기 위해 indexing 을 해 두는 데, 이를 통해 Causal Scenario → UCA → Hazard → Loss 순서로 연관된 항목들을 되짚어가며 최종적으로 Loss 를 줄이기 위해선 어떠한 Safety Constraint 가 필요한지를 분석하게 된다.

우리가 만들고자 하는 프로그램은, 사용자가 Loss, Hazard, Safety Constraint 를 분석한 후 작성할 수 있는 페이지를 제공하는 것에서 시작하여, Editor 화면을 제공해 Control Structure 를 그릴 수 있도록 하고, Process Model 을 해당 Control Structure 에 추가할 수 있도록 하며, Process Model 과 MCS 를 통해 Context Table 을 작성한 뒤 이 Context 들을 바탕으로 UCA Table 이 자동으로 작성되도록 하는 것에서 끝나기 때문에 STPA 의 4 단계 과정 중 1~3 단계를 조금 더 쉽게 수행할 수 있도록 도와주는 역할을 한다. 특히 1 단계와 3 단계에서는 indexing 을 할 수 있는 기능을 제공하여, loss 와 hazard, safety constraint, 그리고 UCA 사이에 traceability 가 보장되어야 한다.

기본적으로 프로그램은 크게 두 가지 파트로 나눌 수 있는데, 기본적인 STPA 의 과정을 수행하기 위한 파트와, NuSCR 이라는 Specification Language 를 이용하여 Supporting 하는 파트로 생각할 수 있다. 현재 우리 프로그램, STPA Supporting Tool 의 목표는 아래와 같다.

① Loss, Hazard, Safety Constraint 를 분석하여 정리할 수 있도록 하는 기능과, Process Model 을 포함한 Control Structure 를 그릴 수 있는 기능, 그리고 Context Table 과 UCA Table 을 작성할 수 있는 기능을 지원하여 반-자동화된 STPA 과정 전반을 지원할 수 있도록 하는 것.

② Formal Specification Language 인 NuSCR 을 사용해 작성된 requirements 로 Process Model 을 채우는 것과, NuFTA 를 사용해 해당 requirement 파일을 Backward Analysis 하여 MCS(Minimal Cut-Set) 값을 도출해내 Context Table 과 UCA Table 의 context 를 채울 수 있도록 하는 것.

1.4 Definitions, acronyms, and abbreviations

Name	Definition
SW	Software
STPA	System Theoretic Process Analysis, 위험 분석 기법
Loss	프로그램 상에서 발생할 수 있는 '사고'
Hazard	Loss 발생의 원인
Safety Constraint	Loss 발생을 막거나 줄이기 위한 제약 조건
Control Structure	SW 의 동작에 있어 문제가 발생하는 원인이 각 component 가 아닌 Control 에 있음에 착안하여 system 을 Control 과 Feedback 관점으로 추상화, 단순화하여 도식화하는 표현법
Process Model	Control Action 을 결정하는 데에 필요한 정보들을 포함
CA	Controller 가 제공하는 Control Action
Feedback	Controlled Process 에서 Control Action 을 이행한 결과, Controlled Process 의 상태 값 등의 정보
Context	CA 를 제공하기 위해 필요한 정보
MCS	Minimal Cut-Set
UCA	Unsafe Control Action
UCA Type	Providing causes Hazardous, Not Providing causes Hazardous, Incorrect Timing/Order, Stopped Too Soon/Applied Too Long
NuSCR	SCR(software cost reduction)에서 확장된 data flow 기반의 정형 명세 언어
NuSRS	정형 명세 언어인 NuSCR 을 사용하여 Nuclear Software Requirements Specification 을 작성할 수 있도록 돕는 프로그램. 계층 구조를 지원한다.

NuFTA	Fault Tree analysis for NuSCR
LHC	Loss, Hazard, (Safety)Constraint
CSE	Control Structure Editor
PMM	Process Model Maker
CTM	Context Table Maker
UTM	UCA Table Maker

1.5 References

IEEE Std. 830-1998

TTA : STPA 를 활용한 위험 분석 가이드

[1] Nancy G. Leveson, John P. Thomas, "STPA Handbook", MIT, 2018

[2] Nancy G. Leveson, "SafeWare : System Safety and Computers", Addison-Wesley, 1995

[3] Nancy G. Leveson, "Engineering a Safer World : Systems Thinking Applied to Safety", MIT Press, 2011

[4] Clifton A. Ericson, II, "Hazard Analysis Techniques for System Safety", Wiley, 2005

[5] Nancy G. Leveson, "An STPA Primer version1", MIT, 2015

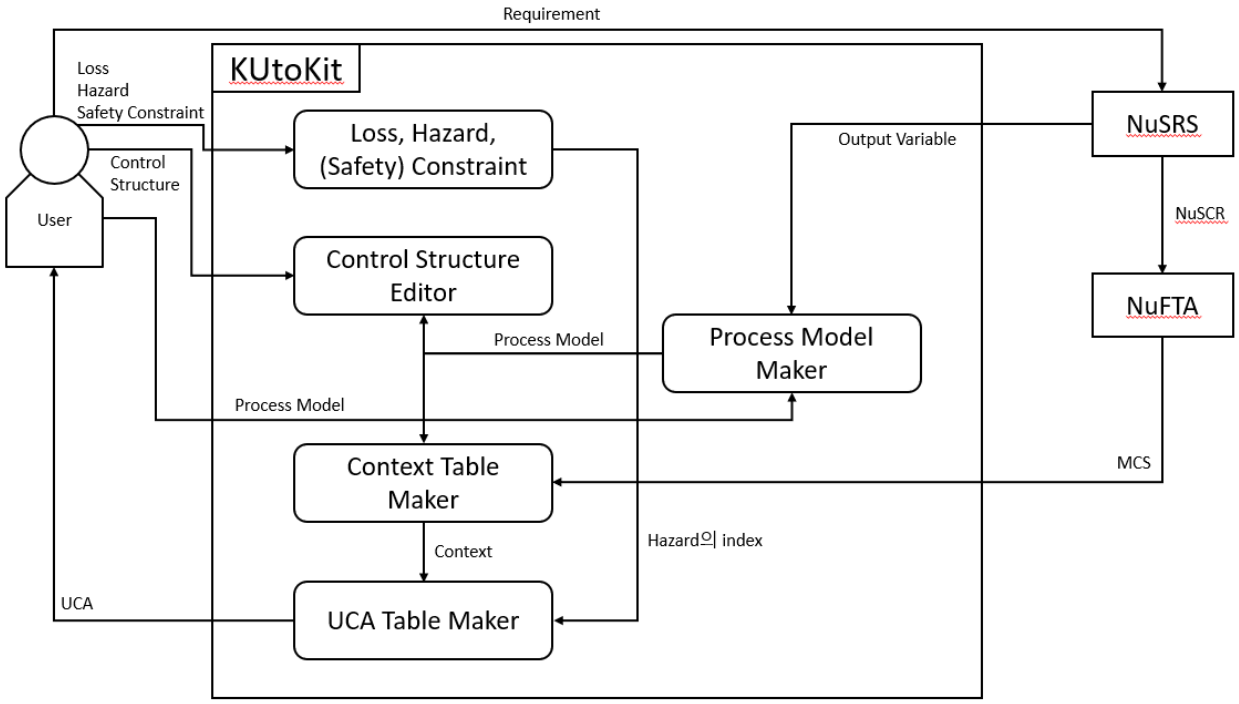
1.6 Overview

개발하고자 하는 프로그램의 purpose, interface, operation, 그리고 추후에 검증 가능한 구체적인 functional requirement 와 non-functional requirement 를 명시하고 프로그램의 제약조건(Constraint)과 시스템의 전체적인 구조와 interface, 함수에 대해 설명한다.

2. Overall Description

2.2 Product Perspective

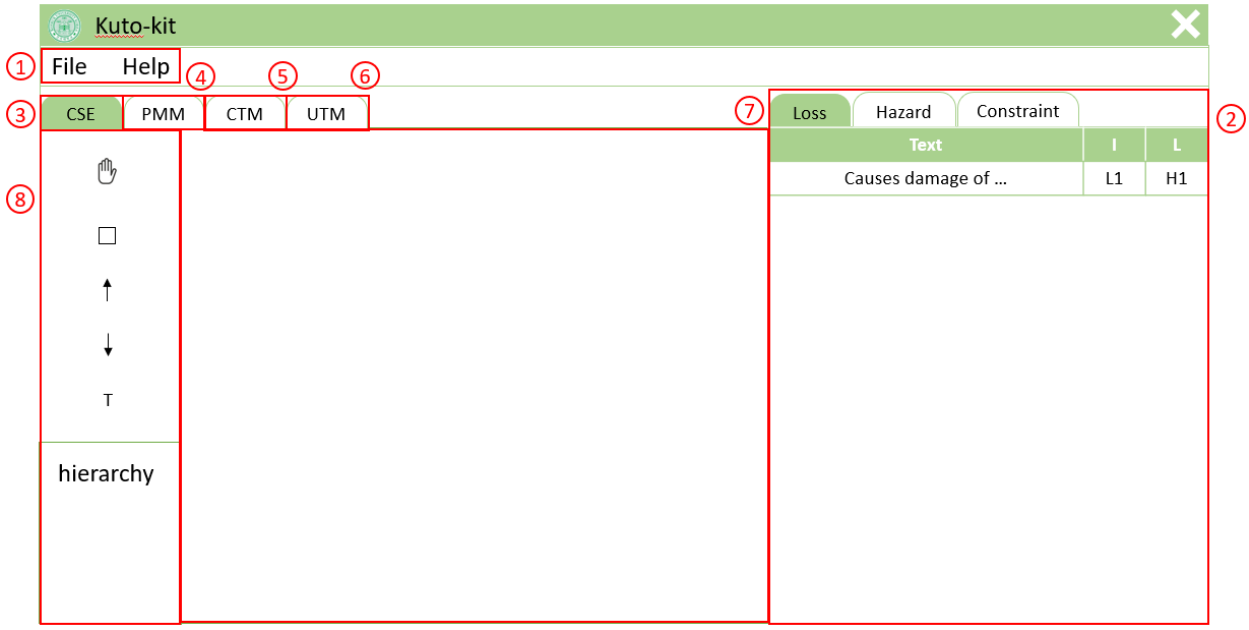
2.2.1 System Interfaces : System Architecture Diagram



- Loss, Hazard, Safety Constraint(LHC) : User 가 분석한 Loss 와 Hazard, Safety Constraint 를 입력하면 자동으로 indexing 하고 link 를 연결할 수 있도록 지원한다.
- Control Structure Editor(CSE) : User 가 Control Structure 를 작성할 수 있는 editor interface 를 지원한다. 이 때 User 는 Control Action 과 Feedback 에 해당하는 내용들 또한 각각 입력할 수 있다.
- Process Model Maker(PMM) : Control Structure 내부의 Controller 에 Process Model 를 작성한다. 단, 이는 NuSRS 로부터 가져온 Requirement 로부터 자동으로 작성되는 것이 기본이나, User 가 별도로 입력할 수도 있어야 한다. 또한, 필요하다면 Control Algorithm 도 작성할 수 있도록 한다.

- Context Table Maker(CTM) : Process Model 로부터 도출된 Context 들을 가지고 Context table 을 작성한다. 이 때 모든 정보들을 사용하는 것이 아니라 NuFTA 를 통해 얻은 MCS 를 기반으로 정보들을 간추려서 작성하도록 한다.
- UCA Table Maker(UTM) : Context Table 로부터 도출된 Context 를 기반으로 UCA table 을 작성한다. UCA table 은 Control Action 과 4 가지의 UCA type 으로 이루어져 있고 각 항목 안에 앞선 단계에서 도출해낸 Context 가 들어간다.

2.2.2 User Interfaces



① Menu Bar : 프로젝트 관련 설정 바

1. File
   - New: 새로운 프로젝트 생성
   - Open: 존재하던 프로젝트 불러옴
   - Save: 현재 프로젝트 저장
   - Save As: 현재 프로젝트를 다른 이름으로 저장하고 원하는 위치에 저장
2. Help: 프로그램 사용 매뉴얼

② LHC Side Bar : Loss, Hazard, Safety Constraint 를 입력하고 관리할 수 있는 사이드바

1. Loss : Loss 의 입력, 관리
2. Hazard : Hazard 의 입력, 관리, loss 로의 link to 입력
3. Safety Constraint : Safety Constraint 의 입력, 관리, hazard 로의 link to 입력
4. Mode Function
    - Input 값들의 자동 indexing
    - Hazard 와 Loss, Safety Constraint 와 Hazard 사이의 link 지정

③ CSE Mode Button : CSE Mode 진입 버튼

1. CSE Mode: LHC 모드에서 Loss, Hazard, Safety Constraint 를 분석하고 입력해 STPA 1 단계를 마친 이후에 실행되어야 하는 모드
2. Input
    - Side Bar 에서 필요한 컴포넌트 선택
    - 필요한 텍스트 입력
3. Mode Function
    - Input 을 바탕으로 Control Structure 생성

④ PMM Mode Button : PMM Mode 진입 버튼

1. PMM Mode: CSE Mode 를 수행하여 Control Structure 을 그린 이후에 원하는 Controller 에 Process Model 을 추가하는 모드
2. Input
    - Process Model 생성에 필요한 NuSCR 파일 (.xml)
3. Mode Function
    - Process Model 을 필요로 하는 Controller 를 선택한 뒤, 외부 파일을 읽어 들여 그에 맞는 Process Model 생성
    - 사용자가 직접 Controller 를 선택하여 텍스트를 입력하여 Process Model 생성

⑤ CTM Mode Button : CTM Mode 진입 버튼

1. CTM Mode: PMM Mode 까지 수행한 이후, context table 을 생성하는 모드
2. Input
   - Context Table 생성에 필요한 MCS 파일 (.txt)
3. Mode Function
   - 입력 받은 MCS 파일을 적용하여 Control Action 별로 context 를 채워 넣어 Context Table 작성

⑥ UTM Mode Button : UTM Mode 진입 버튼

1. UTM Mode: CTM Mode 까지 수행한 이후 UCA Table 을 자동으로 생성해주는 모드, hazard 로의 link to 입력
2. Input
   - CTM Mode 에서 생성된 Context Table 의 Context
3. Mode Function
   - 사용자가 입력하는 것이 아닌, 프로젝트 내에서 저장된 Context 를 기반으로 자동으로 UCA Table 작성

⑦ Board: 사용자 인터페이스 레이아웃

- 각 Mode 에 맞는 화면을 출력
- 유저의 입력을 받음

⑧ Side Bar : CSE 모드에서의 기능 수행에 필요한 버튼들이 포함되어 있는 사이드 바

- CSE Mode: Control Structure 생성 시 필요한 컴포넌트(Controller/Controlled Process, Control Action/Feedback)
- Control Structure 에서의 계층구조를 보여준다.

2.2.3 Hardware Interfaces

2.2.4 Software Interfaces

A. NuSRS

Name : Nuclear Software Requirement Specification

Mnemonic : NuSRS

Specification Number : 1.0

Version Number : 2.0

Source : http://dslab.konkuk.ac.kr/Nuclear-Requirement/Nuclear-Requirement.htm

B. NuFTA

Name : NuSRS Fault Tree Analysis

Mnemonic : NuFTA

Specification Number : 1.0

Version Number : 2.0

Source : http://dslab.konkuk.ac.kr/Nuclear-Requirement/NuFTA.htm

2.2.5 Communications Interfaces

2.2.6 Memory Constraints

2.2.7 Operations

- 사용자는 traceability 를 보장하기 위해 Loss 와 Hazard, Hazard 와 Safety Constraint 사이의 link 를 지정할 수 있다.
- 사용자는 CSE 에서 Control Structure 를 작성할 수 있다.
- Control Structure 에서 CA 와 Feedback 의 내용들은 사용자가 입력할 수 있다.
- NuSCR 로 작성된 Requirement 파일을 이용해 지정한 Controller 에 Process Model 을 입력할 수 있다.

- Process Model 은 사용자가 임의로 추가할 수도 있다.
- NuFTA 를 통해 도출된 MCS 파일을 불러올 수 있다.
- 불러온 MCS 파일에서 중요한 정보만을 추려내 Context Table 을 작성할 수 있다.
- Context table 에서 해당 Context 가 hazardous 한지 아닌지 여부를 선택할 수 있다.
- Hazardous 하다고 판단한 Context 를 기반으로 UCA table 을 자동으로 작성할 수 있다.
- UCA Table 에서 UCA 와 hazard 사이의 link 를 지정할 수 있다.

2.2.8 Site Adaptation Requirements

JDK : 8.0 or above

2.3 Product Functions

① Open : .xml 형식으로 저장된 파일을 불러온다. 전체 프로젝트를 불러올 수 있다.

② Save : .xml 형식으로 진행 중이던 전체 프로젝트를 저장할 수 있다.

③ Add Loss, Hazard, Constraint : LHC Side Bar 에서 Loss, Hazard, Constraint 에 대한 text 를 입력 받고, 이들 각각을 자동으로 indexing 하여 테이블을 생성한다.

④ Create Control Structure : Control Action 과 Feedback 을 입력 받고, 사용자의 입력을 통해 Control Structure 을 생성한다.

⑤ Get NuSCR File : NuSCR 로 작성된 requirement 파일을 가져온다.

⑥ Parse NuSCR File : NuSCR 로 작성된 requirement 파일을 분석하여, 분석 대상이 되는 Controller 의 CA 와 연관된 variable 을 추출한다.

⑦ Create Process Model : 추출된 variable 들을 이용하여 Controller 별 Process Model 을 생성한다.

⑧ Select MCS : NuFTA 를 통해 추출된 MCS 중 유의미한 것을 선별한다.

⑨ Create Context Table : 선별된 MCS 를 이용해 Context table 을 생성한다.

⑩ Create UCA : 생성된 Context table 에서 hazardous 하다고 선택된 context 들을 기반으로 Controller 별 CA 에 따라 UCA 후보군을 생성한다.

⑪ Create UCA Table : loss scenario 를 분석할 수 있도록 생성된 UCA 후보군을 바탕으로 자동으로 UCA table 을 생성한다.

2.4 User Characteristics

STPA 의 적용에 어느 정도 지식이 있는 user 를 타겟으로 한다. 사용자는 기본적으로 STPA 과정에 대한 전반적인 지식이 있고, 이 SW 를 사용하기 위해 STPA 의 4 단계 과정 중 4 단계를 제외한 나머지 단계 전반을 수행할 줄 알아야 한다. 1 단계의 경우 사용자가 직접 loss 와 hazard, hazard 와 safety constraint 를 link 시킬 수 있어야 하고, 3 단계에서는 UCA 와 hazard 를 link 시킬 수 있어야 하므로 이를 분석할 줄 알아야 한다.

2.5 Constraints

- NuSCR 로 작성된 SW 를 분석하는 것을 기본으로 한다.
- Controller 가 존재하는 System 에 관한 분석만 할 수 있다.
- 반드시 2.2 에서 기술된 function 의 순서를 지켜서 프로그램을 동작하도록 한다.
- 프로그램 내부에서 사용되는 MCS 는 NuFTA 를 통해 추출된 것이다.
- NuSRS 와 NuFTA 는 프로그램과 직접적으로 연동되지 않으므로, 사용자가 별도로 조작해야 한다.
- NuSRS 를 통해 Process Model 의 변수들을 추출하는 것을 기본으로 한다.

2.6 Assumptions and dependencies

- SW 는 최종적으로 jar 파일로 생성된다.
- SW 에서 파일 저장 시, 저장되는 형식은 xml 을 기본으로 한다.

- 처음 프로젝트를 생성하는 경우, 각 기능들은 LHC → CSE → PMM → CTM → UTM 순서대로 순차적으로 실행된다.

2.7 Apportioning of requirements

- NuFTA 를 통해 추출된 MCS 중 유의미한 값을 자동으로 선별해낼 수 있도록 기준을 제시하여 적용할 수 있도록 한다.
- NuSCR 이외의 formal specification language 로 requirement 가 작성된 SW 도 분석할 수 있도록 한다.

3. Specific Requirements

3.2 External Interfaces

3.2.1 User Interfaces

Input: 키보드/마우스 입력

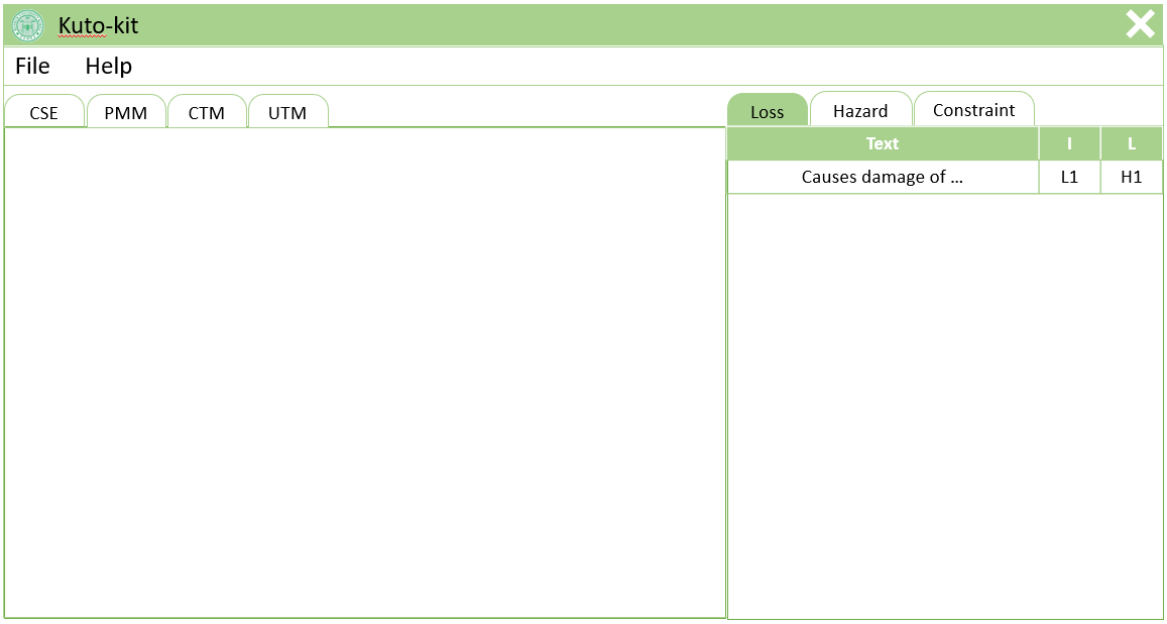
Output: 화면 출력

단, 표시되는 UI 의 기본 사이즈는 지정되어 있으며 변경할 수 없다.

3.2.2 HW Interfaces

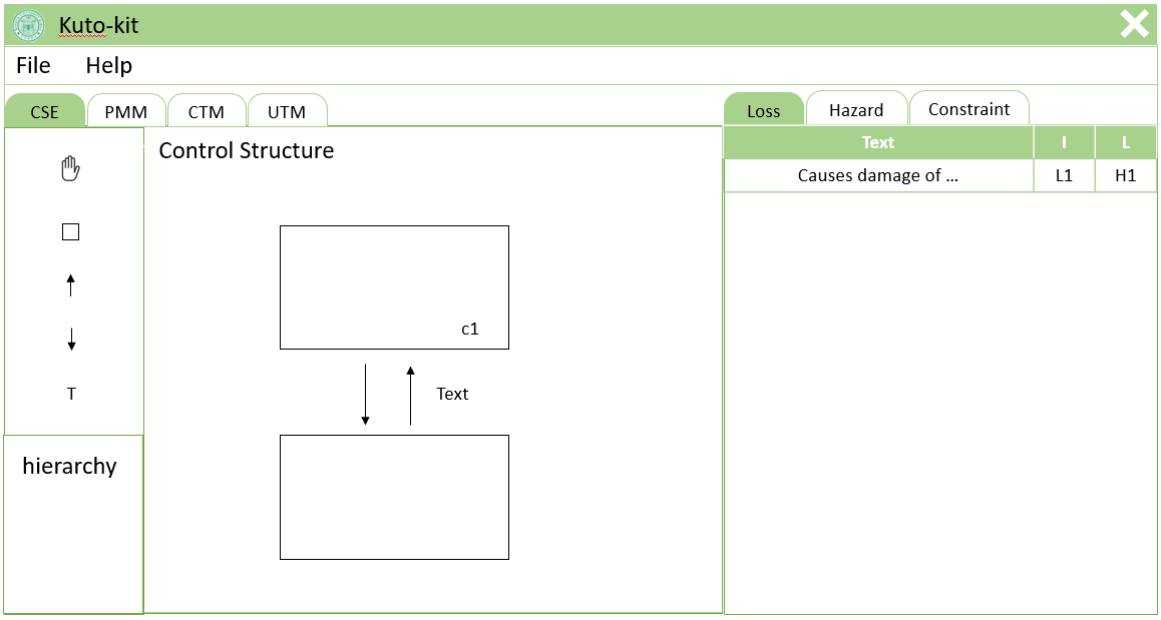
3.2.3 SW Interfaces

3.2.3.1 Loss, Hazard, Constraint mode (LHC Mode)



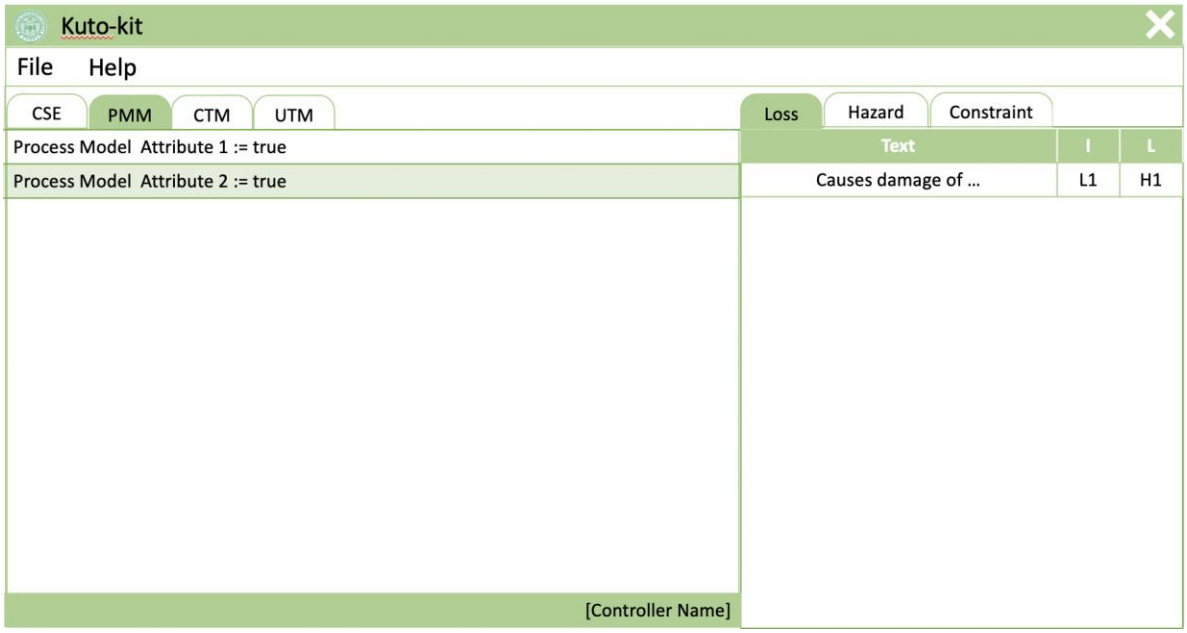
- Input : Loss/Hazard/Constraint 와 요소들간의 link 입력 (키보드/마우스 입력)
- Output : LHC Side Bar 에 input 에 따른 Table UI 출력

3.2.3.2 Control Structure Editor Mode (CSE Mode)



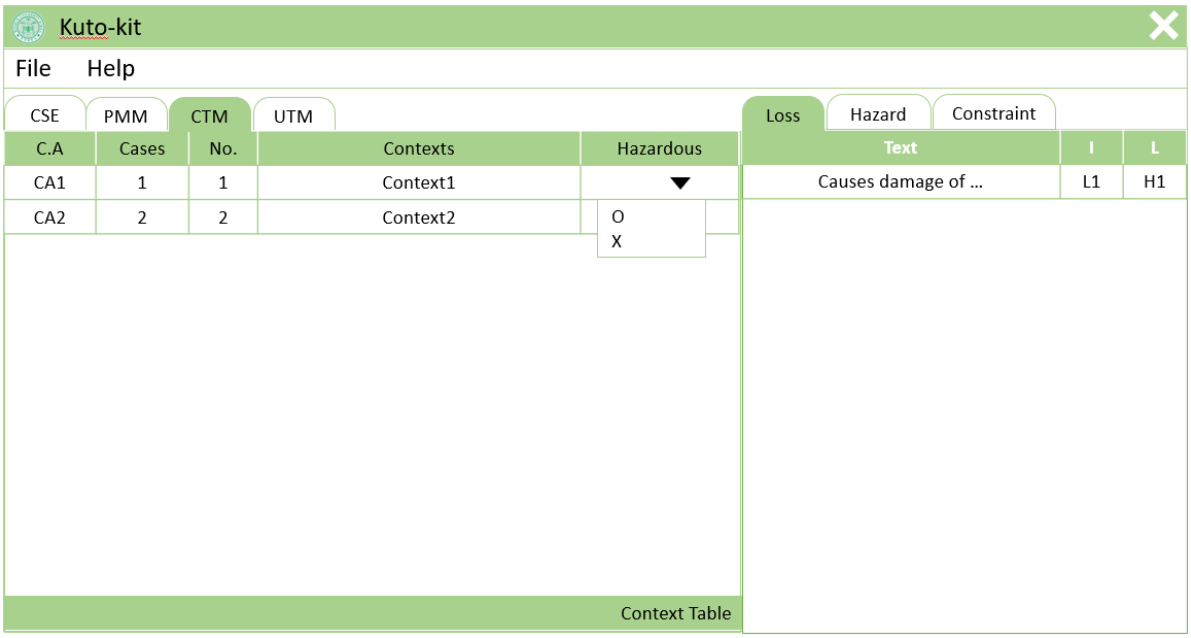
- Input : Controller/Controlled process/control action/feedback 선택 (키보드/마우스 입력)
- Output : board 에 Control Structure 생성

3.2.3.3 Process Model Maker mode



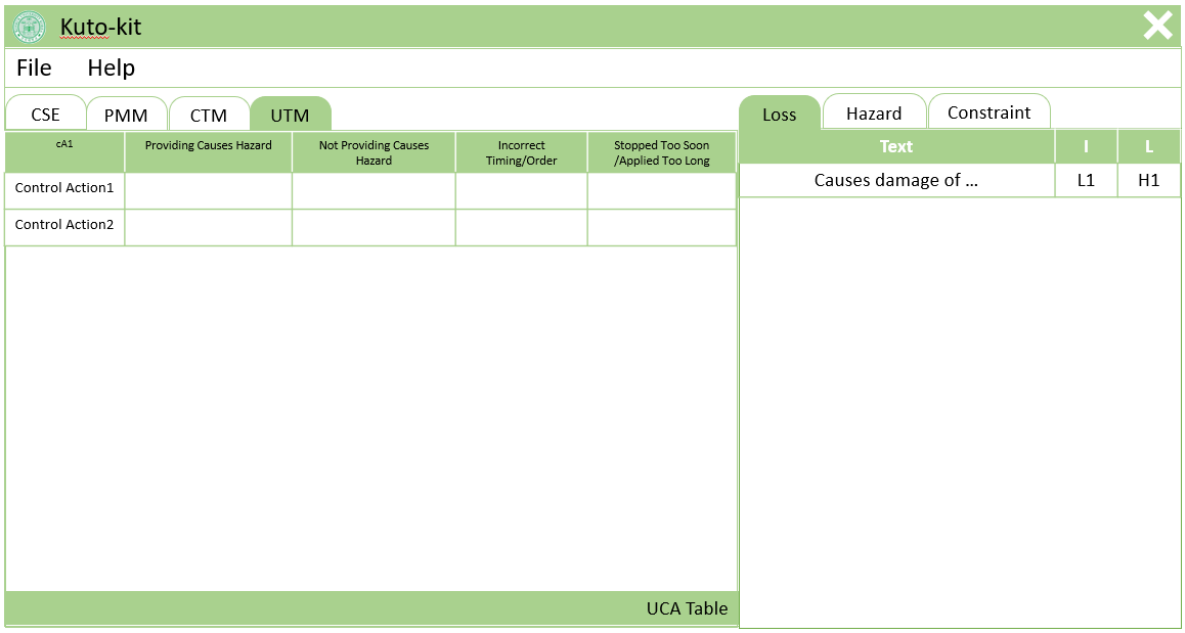
- Input : Process model 에 필요한 state, environment (NuSCR 파일 파싱 과정을 거쳐 얻어진 데이터 또는 키보드 입력)
- Output : board 에 input 에 따른 UI 출력

3.2.3.4 Context Table Maker mode



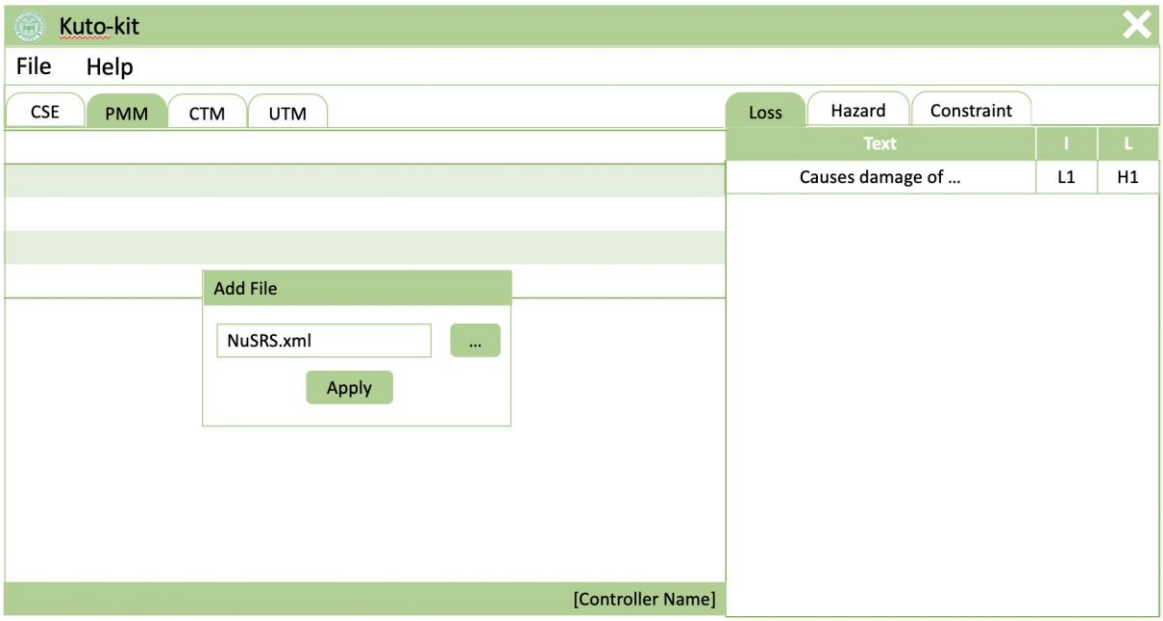
- Input : MCS 파일, hazardous 여부 (마우스 입력)
- Output : board 에 Context table 생성 및 출력

3.2.3.5 UCA Table Maker mode

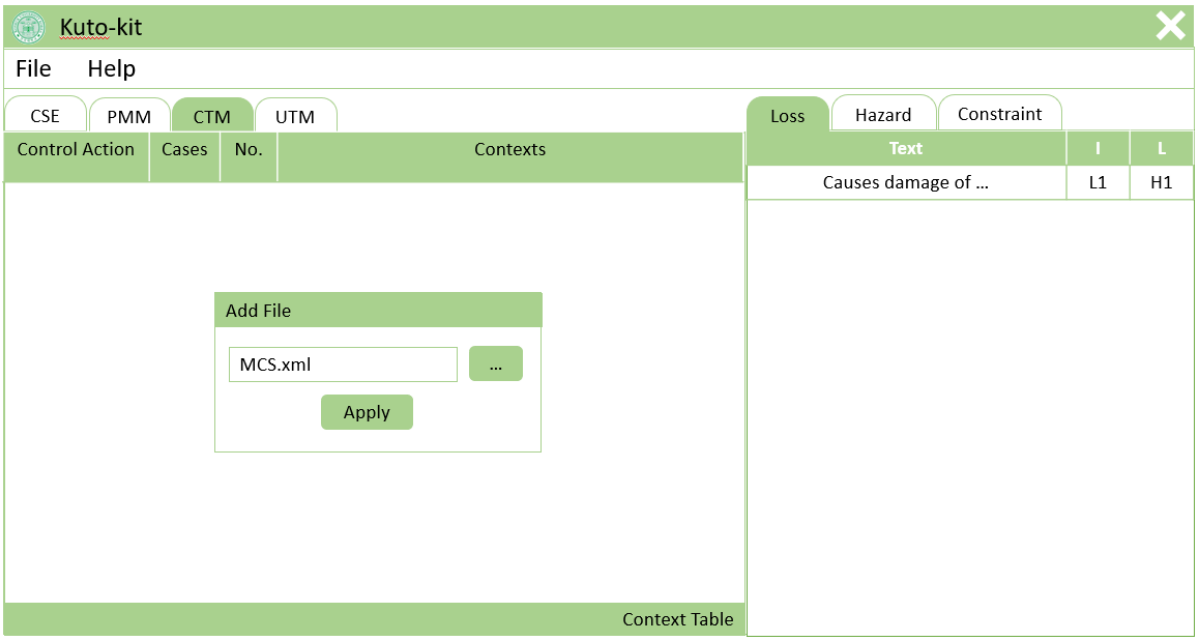


- Input : Context table 에 채워진 context 의 내용과 hazardous 여부(사용자 입력 없음), link to (키보드 입력)
- Output : board 에 UCA table 생성 및 출력

3.2.3.6 Add external file mode



- Input: NuSCR file (.xml)
- Output: Process Model



- Input: MCS file (.txt)
- Output: Context Table 의 context

3.2.4 Communications Interfaces

3.3 Classes

3.3.1 Main Screen

3.3.1.1 Attributes

- mode : 현재 User 가 사용하고 있는 Mode 를 의미한다. 이때의 Mode 는 3.1.3.1~3.1.3.5 에 서술된 5 가지 Mode 중 하나를 의미한다.

3.3.1.2 Functions

- Select mode : User 가 원하는 Mode 로 화면이 전환되도록 한다. 사용 가능한 Mode 에는 3.1.3 SW interface 에 서술된 모드 5 가지가 존재한다.

- Create Project : 하나의 프로젝트를 생성하는 파일이다. 프로젝트 내에서 생성될 Control Structure, Process Model, Context table, UCA table 이 들어갈 폴더 이름을 지정하고 생성한다.
- Open Project : 프로젝트를 불러오는 함수이다. 지금 사용중인 프로젝트가 있다면 프로젝트를 종료하고 지정한 폴더의 프로젝트를 불러온다.
- Save Project : 현재 사용하고 있는 프로젝트 폴더에 지금까지 만든 LHC, Control Structure, Process Model, Context table, UCA table 을 저장하는 함수이다. 만일 경로가 손상된 경우 error 메시지 출력.
- Save As Project : User 가 현재 사용중인 프로젝트를 다른 이름으로 원하는 경로에 저장할 수 있다.
- Print help : 이 시스템을 사용하는 방법을 알려주는 매뉴얼을 보여주는 함수이다. 원하는 모드를 선택하면 그에 맞는 사용법을 출력해준다.

3.3.2 LHC Controller

3.3.2.1 Attributes

- LHC : LHC

3.3.2.2 Functions

- Choose LHC type : LHC 의 세 가지 type(Loss, Hazard, Safety Constraint) 중에서 추가/수정/삭제하고자 하는 type 의 탭을 선택한다.
- Add LHC text : 선택한 Type(Loss, Hazard, Safety Constraint)에 해당하는 내용을 키보드로 입력 받고, 사용자가 Add 버튼을 누르면 table 에 추가한다. Text 입력 없이 add 하는 경우 error 메시지 출력.
- Add LHC index : 내용 입력 시 자동으로 type 별 index 를 추가한다.
- Select LHC link : Loss 와 Hazard, Hazard 와 Safety Constraint 를 어떻게 연결하여 indexing 할지 선택한다. 한 번에 여러 개의 link 를 선택할 수 있도록

한다. Loss 를 제외한 두 가지 항목에서 Link 를 설정하지 않는 경우 error 메시지 출력. (default: null)

- Delete LHC : 선택된 Loss/Hazard/Safety Constraint 를 삭제한다.
- Delete Index : Cell 삭제 시 해당 cell 의 index 삭제. Index 는 위로 하나씩 밀려서 재정렬되도록 한다.
- Delete Link : Cell 삭제 시 link 되어있는 cell 에서 해당 link 삭제.
- Modify LHC text : 선택된 블록의 Text 를 수정한다.
- Save LHC : 작업 중이던 LHC table 의 내용을 저장한다.

3.3.3 Control structure editor

3.3.3.1 Attributes

- Board: Pane
- Components: Array<Node>

3.3.3.2 Functions

- Select Component: User 가 추가할 컴포넌트를 선택한다.
- Select Text: User 가 텍스트를 추가한다.
- Add Component: User 가 선택한 컴포넌트를 board 에 추가한다.
- Add Text: 텍스트를 board 에 추가한다.
- Drag Component: User 가 해당 컴포넌트를 드래그 한다.
- Move Component: User 가 드래그한 컴포넌트를 이동한다.
- Modify Component : User 가 컴포넌트를 수정한다. controller 의 경우 name 을, control action 와 feedback 의 경우 text/source/destination 을 수정할 수 있다. 이 때, control action 의 이름이 하나라도 채워지지 않은 경우 경고창을 띄운다.
- Delete Component : User 가 선택한 컴포넌트를 삭제한다.

- Delete Text : User 가 선택한 텍스트를 삭제한다.
- Save CSE : 현재 control structure 을 저장한다. 이 때, control structure 에서 최종적으로 loop 구조(닫힌 구조)가 만들어지지 않는 경우 경고창을 띄운다.
- Parse XML: 현재 control structure 을 xml 파일로 파싱한다.

3.3.4 Process model maker

3.3.4.1 Attributes

- File : NuSRS File
- XML Reader : xml reader
- Observable List : value list

3.3.4.2 Functions

- Add File : 선택된 파일을 PMM 에 가져올 수 있게 한다.

  E1. 파일 형식이 맞지 않는 경우 '파싱 불가' 라는 경고 창을 띄우고, 작업을 중지한다.
- Apply File : 선택된 Controller 와 CA 에 기반하여 파싱 된 파일을 불러온다.

  E1. 선택된 Controller 의 이름 / Controller 나 CA 에 대한 정보가 존재하지 않아 모델 생성 불가한 경우 경고 창을 띄우고, 작업을 중지한다.

  E2. 읽어오려는 file 에 error 가 존재하는 경우, 경고 창을 띄우고 작업을 중지한다.
- Make Process Model : 선택된 Controller 와 CA 에 기반하여 PM 을 생성한다.
- Select Controller : 원하는 Controller 를 선택한다.
- Select Process Model : 원하는 PM 을 선택한다.
- Add Value:
  i. PM 이 존재할 때, 입력된 variable 를 추가한다.
  ii. PM 이 존재하지 않을 때, 입력된 variable 을 기반으로 PM 을 생성한다.

- Delete Value :
  i. PM 이 존재할 때, 선택된 variable 를 삭제한다.
  ii. PM 이 존재하지 않을 때, 작동하지 않는다.
- Save PMM : 현재 PMM 모드를 저장한다.

3.3.5 Context table maker

3.3.5.1 Attributes

- CTM : ct
- File : MCS File

3.3.5.2 Functions

- Add File : 선택된 파일을 CTM 에 가져올 수 있게 한다. 단, 포맷에 맞지 않는 파일을 파싱 하려고 할 경우 경고창을 띄운다.
- Parse file : 포맷에 맞는 파일을 선택하여 필요한 정보들을 context 로 파싱한다. 파싱 해오는 기준은 이미 정해져 있어야 한다. NuFTA 를 통해 도출된 MCS 파일이 존재하는 상태에서 진행할 수 있다. 단, 파싱해온 파일로 Context 에 해당하는 칸이 다 채워지지 않은 경우 경고창을 띄운다.
- Make Table : context table 에 파싱 해온 파일의 내용들을 채운다.
- Modify Table: context 의 내용을 수정한다.
- Select Hazardous : 각각의 context 에 대한 hazardous 여부를 선택한다 . 선택하지 않을 경우 다음 단계인 UCA table 을 생성하는 단계로 넘어갈 수 없다. 단, hazardous 여부를 선택하지 않은 경우 경고창을 띄운다.
- Save CTM : 현재 CTM 모드를 저장한다.

3.3.6 UCA table maker

3.3.6.1 Attributes

- Context : context
- UCA : uca

3.3.6.2 Functions

- Set UCA Table : Context table 에서 Hazardous 항목의 값이 'O' 인 Context 들로 UCA table 을 자동 생성한다.
- Add UCA : 추가적으로 필요하다고 생각되는 UCA 를 User 추가한다.
- Modify UCA : 적절하지 않은 UCA 를 User 가 수정한다.
- delete UCA : 필요없는 UCS 를 User 가 지운다.
- Save UCA : UCA Table 을 .xml 형태로 저장한다.

3.3.7 Xml Reader

3.3.7.1 Attributes

- File : NuSRS File
- XPath : xPath
- Document : document

3.3.7.2 Functions

- Parse File : 특정 name 을 가진 NuSRS File 을 파싱한다.
- Show Valid FODs : 현재 노드에서 유효한 모든 FOD 를 추출하여 출력한다..
- Get Node : 원하는 SDT/TTS/FSM 노드를 추출한다.
- Get Node List : 원하는 SDT/TTS/FSM 노드의 연관 Input variables 을 추출한다.
- Get Transition Nodes : 원하는 노드와 연관된 노드를 추출한다.

3.3.8 Control Action

3.3.8.1 Attributes

- text : String
- source : Controller
- destination : Controller

3.3.8.2 Functions

- Get CA : control action 의 정보(attributes)를 가져온다.
- Set CA : 입력 받은 정보를 control action 에 저장한다.

3.3.9 Controller

3.3.9.1 Attributes

- name : String
- Process Model : Array<String>
- CA : Array<Control Action>

3.3.9.2 Functions

- Get Controller : controller 의 정보(attributes)를 가져온다.
- Set Controller : 입력 받은 정보를 controller 에 저장한다.
- Add CA : controller 에 control action 을 추가한다.
- Delete CA : controller 에 있는 control action 을 삭제한다.

3.3.10 LHC

3.3.10.1 Attributes

- Type : Loss, Hazard, Safety Constraint
- Text : 각 type 에 해당하는 내용
- Index : Type[index]
- Link to : Loss 의 경우 어떤 Hazard 와, Hazard 의 경우 어떤 Safety Constraint 와 matching 될 것인지에 대한 link

3.3.10.2 Functions

- Get LHC : LHC 의 type 과 text, index 를 가져온다.
- Set LHC : 입력 받은 type 에서의 text 와 index 를 LHC 에 저장한다.

3.3.11 Context

3.3.11.1 Attributes

- Control Action : Control Action Name
- case : Case Number
- no : Context Number
- Hazardous : 각 Context 의 위험 여부
- Context : Control Acton 에 대한 설명

3.3.11.2 Functions

- Get Control Action : control Action 을 가져오는 함수
- Get Case : case 를 가져오는 함수
- Get No : no 를 가져오는 함수
- Get Hazardous : hazardous 를 가져오는 함수
- Get Context : context 를 가져오는 함수
- Set Hazardous : hazardous 를 설정하는 함수
- Set Context : context 를 설정하는 함수

3.4 Performance Requirements

- 사용자의 입력(버튼 클릭, 드래그, 텍스트 입력)에 대한 반응 속도는 1 초 이내로 한다.
- 전체 과정은 하나의 프로젝트 형태로 저장된다.
- 동시에 여러 프로젝트에 대한 작업을 수행할 수는 없다.

3.5 Logical Database Requirements

해당 툴은 STPA 의 과정을 좀 더 빠르고 간단하게 수행할 수 있도록 돕는 것이 목적이므로, 파싱해 올 파일에서 어떠한 정보를 취할 것인지 대한 정보를 이미 가지고 있어야 한다.

단, 어떤 정보가 유용할지는 추후 여러 차례의 시도를 통해 알아가야 할 것이다.

3.6 Design Constraints

3.7 Software System Attributes

- Maintainability : 프로그램 내부에서 충돌이 발생하거나 실행에 있어 너무 오랜 시간이 걸리지 않도록, 유지 보수성을 높일 수 있도록 함수의 계층 구조나 각 함수 간의 관계의 복잡도가 너무 높지 않도록 제한한다.

3.8 Additional Comments

4. Supporting Information

4.2 Table of contents and index

4.3 Appendixes