TOOL-ASSISTED HAZARD ANALYSIS AND REQUIREMENT GENERATION BASED ON STPA

by

Dajiang Suo

B.S. Mechatronics Engineering
Beijing Institute of Technology, 2007

M.S. Computer Science
Tsinghua University, 2012

SUBMITTED TO THE ENGINEERING SYSTEMS DIVISION IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN ENGINEERING SYSTEMS
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
February 2016

Signature of Author: Signature redacted
Engineering Systems Division
December 7, 2015

Certified by: Signature redacted
Nancy G. Leveson
Professor of Aeronautics and Astronautics and Engineering Systems
Thesis Co-Supervisor

Certified by: Signature redacted
John Thomas
Research Engineer, Department of Aeronautics and Astronautics
Thesis Co-Supervisor

Accepted by: Signature redacted
John Tsitsiklis
Clarence J Lebel Professor of Electrical Engineering
Graduate Officer, IDSS

Tool-Assisted Hazard Analysis and Requirement Generation based on STPA

by

Dajiang Suo

SUBMITTED TO THE ENGINEERING SYSTEMS DIVISION IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN ENGINEERING SYSTEMS

ABSTRACT

The automotive industry has been observing a trend of integrating new features into old vehicle designs to provide more convenience and flexibility to customers. However, it can be challenging to ensure safety without the support of appropriate techniques and tools for hazard analysis and requirement engineering.

Systems Theoretic Process Analysis (STPA) is a hazard analysis technique that has been developed at MIT. It is based on systems and control theory and aims at capturing more causal factors leading to accidents, including component interactions. So far, STPA has been successfully applied to various industries. While there are tools that allow engineers to document the results of hazard analysis based on STPA, there are no tools that provide guidance during the analysis. Also, although a method is proposed to generate requirements from an STPA analysis, no tools have been developed to support that process.

This thesis illustrates how tools can provide support for hazard analysis and requirement generation based on STPA, based on the proof of concept of a software tool that was developed at MIT. This STPA tool assists STPA Step 1 analysis by applying logical simplification to the original Step 1 results and automatically generating the simplified requirement in formal and executable forms. The simplified requirements are easily understandable and address all of the unsafe control actions identified in the original STPA analysis. The use of the STPA tool is illustrated through a case study of automotive systems that include multiple features. The STPA tool generates simplified and formal requirements for each individual feature based on STPA Step 1 results. In addition, it is also used to check whether conflicts between features have been resolved and to identify exactly what decisions should be made jointly between multiple design teams.

Thesis Co-Supervisor: Nancy G. Leveson
Title: Professor of Aeronautics and Astronautics and Engineering Systems

Thesis Co-Supervisor: John Thomas
Title: Research Engineer, Department of Aeronautics and Astronautics

[Page intentionally left blank]

Acknowledgements

First, I would like to thank my supervisor, Professor Nancy Leveson, for believing in me and teaching me System thinking. Also, I would like to thank her for giving me the opportunity of joining the program in ESD and working on the research project. I am deeply grateful for her help and encouragement through my entire program of study.

I would like to thank Dr. John Thomas for his guidance throughout the project and my thesis writing. Thanks for his patience and always being willing to help during the research project.

To other members in our research group, I would like to thank you all for your help and discussion that makes 33-407 an awesome place to work. Also, I want to give my special thanks to those when I arrived at Boston for the first time and learned how to live here. I would like to thank Cody, the first person I met when I first came to MIT, for your kindness and encouragement to a "stranger", and to Qi, my instructor in the Systems Engineering course, for teaching me how to survive at MIT. Seth and Soshi, I should thank you for being so nice and taking me to lots of interesting places in Boston.

I would also like to thank the experts who were willing to discuss with me and teach me domain knowledge: Tony, Matt, Kyle and Sarra. I would like to acknowledge Professor Flach for his discussions and suggestions.

Finally, I must thank my mom, for giving me life, loving me and raising me up. I love you.

[Page intentionally left blank]

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Limitations in Old Safety Engineering Techniques

Although old safety engineering methods and techniques have been used to prevent accidents led by component failures, they become ineffective in dealing with hazardous interactions in feature-intensive systems (e.g., avionics, automotive and medical devices, etc.). Specifically, they have limitations when it comes to hazard analysis and early development of safety requirements for these systems.

First, design assumptions for old system components are often violated when adding new components into the existing system [1]. Often, old design decision rationales are forgotten during the feature integration. As Leveson [2] suggests, four aspects deserve attention when integrating new components into existing systems:

- Documentation of design rationale;
- Documentation of the assumptions about the operational environment that are "implicit" in the design;
- Traceability from high-level system requirements (e.g., system-level goals or hazards) to system design;
- Documentation of hazard analysis and safety information.

Secondly, most techniques for hazard analysis are effective in dealing with component failures, but are incapable of capturing hazardous scenarios arising from feature iterations at early stages of system development. In particular, little guidance is provided to engineers trying to evaluate the safety impact of design decisions.

Thirdly, the complexity in feature-intensive systems has increased to the extent that old techniques for hazard analyses fall short. For example, automotive systems that are rich in software features can interact with other vehicle systems in complex ways. Placke, Thomas and Suo [18] demonstrate that in a case study of a multiple control systems. As Leveson [4] suggests, the increased use of digital components makes the systems we build software-intensive and much more complex than those based on analog equipment. In addition, human operators (e.g., drivers) often respond to new features in unpredictable ways, adding more complexities to automotive systems that have been built.

For automotive manufacturers, new techniques for hazard analyses are needed during feature integration. More importantly, there are few tools that provide guidance to identify hazardous feature interactions and generate safety requirements based on results from hazard analyses.

1.2 Challenges in Designing Automotive Features

The automotive industry has been observing a trend of integrating new features into vehicle designs to provide more convenience and flexibility to customers. However, it can be challenging without the support of appropriate techniques and tools for hazard analysis and requirement engineering.

To begin with, new hazards may be introduced when changes are made [6]. For example, the "Keyless Go", a feature that allows the owner to start the engine without inserting the key into the ignition switch but by pushing an "Engine Start/Stop" button, introduced new hazards during operating the vehicle. Drivers have been reporting two hazards—carbon monoxide poisoning and unintended rollaway [7]. Both of them can be explained by design changes in the key-related system. On one hand, the keyless fob device plays no role in shutting down the engine and no longer has to do with the transmission range. On the other hand, if the driver still perceives the keyless "fob" (in the pocket) as an indication of engine status or transmission range, he or she may easily forget to check if the vehicle has been turned off and put into "park".

According to NHTSA [39], customer complaints about the unintended rollaway related to the "Keyless Go" feature have been increasing dramatically since 2002, as shown in Figure 1. Often, the driver forgets to shut down the engine or put the transmission in the "park" range before leaving the vehicle. Several recalls were conducted due to incidents or violation of regulations (e.g., FMVSS 114). Causal factors responsible for the rollaway incidents include incorrect assumptions about interactions with other features and unexpected driver behaviors. One contributing factor is that there is no standard engineering process to evaluate designs based on old assumptions when adding new components (e.g., Keyless Go) into the system.

In addition, assumptions for the old design may become obsolete and invalid after adding new features. Take the "shift-by-wire" concept, a system that substitutes traditional mechanical linkages between the shifter and the transmission linkage with electronic systems, as an example.

Some vehicle models with this feature can put themselves in "park" without input from the driver. However, those vehicles [8] are also designed to shift to "neutral" automatically under "abnormal" conditions. This decision is based on the assumption that in emergency situations, moving the vehicle out of a "danger area" is necessary if it experiences sudden deceleration due to engine failure when running on the highway. However, this assumption may become invalid if a driver who is not familiar with this new feature always expects the automatic engagement of "park", no matter what happens to the vehicle. In fact, such violations of design assumptions have resulted in accidents of unintended rollaway [8] that cause injuries to pedestrians.

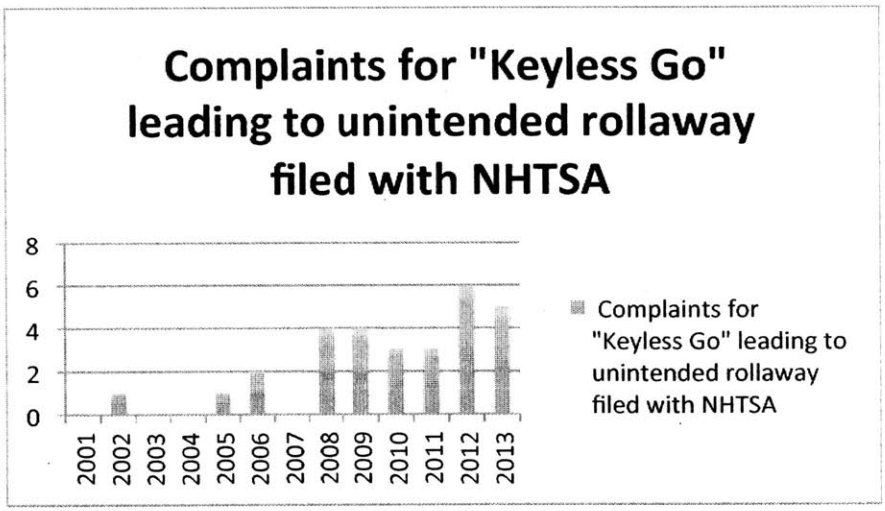


Figure 1: Customer complaints for Keyless Go leading to rollaway incidents

Furthermore, integrating new features may change the way that operators interact with computer systems. According to Leveson [9][10][11], "increased automation in complex systems has led to changes in human controller's role and to new types of technology-induced human error". While bringing convenience to the driver, substituting physical keys with electronic ones can also lead to the driver's omission of putting the transmission in "park" before leaving the vehicle. The traditional key, used to open the door and to start or stop the engine, also serves as an indicator for the transmission range and prevents the driver from removing the key from the ignition switch due to the interlock between the ignition switch and the transmission system. It is required by regulation (i.e., FMVSS 114) that "The starting system must prevent key removal unless the transmission or gear selection control is locked in "park" or becomes locked in "park" as the direct result of key removal". In several recalls of rollaway issues led by the "Keyless Go" feature, the driver can shut down the engine and leave the vehicle without the "park" range being engaged (a sequence indicating the violation of FMVSS 114).

Finally, introducing new features that involve changes to user interfaces can lead to inconsistencies in mental models between the operator and the designer. Feature designers often use their own models of the system to come up with the logic of automated features without realizing that their models can differ from those of operators, whose mental models can change dynamically during the operation. Again, take the recall related to the vehicle models with "Keyless Go" in 2009 [8] as an example. Unlike previous models where the "Engine Start/Stop" button only plays the role of turning the engine (ignition) on or off, certain models with the "shift-by-wire" feature can also put the transmission in "park" automatically when the "Engine Start/Stop" button is depressed. The problem is that the vehicle can also put the transmission in the "neutral" range automatically when certain "abnormal" conditions are met, as shown in Table 1 [8]. The driver, usually a novice in operating these new vehicle models, may get confused with the automated behaviors of the vehicle. Leveson [4] argues that software designers must pay attention to whether the automation behavior is transparent and consistent for human supervision, in addition to putting efforts into the internal design of these features.

Table 1: Two scenarios where N range (rather than P) will be achieved as "emergency function"

Condition	Driver input	Current transmission range	Abnormal conditions	Target range achieved in normal situation	Actual range achieved
1	Press push button	Drive/ Reverse	Driver pushes Start/Stop button 2x or 3x at intervals of 300–500ms	Park	Neutral
2	Press push button	Drive/ Reverse	Engine shuts down due to engine fault or empty tank	Park	Neutral

1.3 Goals and Objectives

Systems Theoretic Process Analysis (STPA) is a hazard analysis technique based on systems and control theory. It has been developed at MIT [4] and aims at capturing more causal factors leading to accidents including component interactions. So far, STPA has been successfully applied to various industries such as aerospace, aviation, railroad, automotive, medical and nuclear power [42][40][41][31][43]. While there are tools such as SpecTRM [15], A-STPA [27] and SafetyHAT [28] that allow engineers to document the results of hazard analysis based on STPA, there are no tools that provide guidance during the analysis. Also, although Thomas [31] demonstrated a method to generate requirements from an STPA analysis, no tools have been

developed to support that process. Therefore the main goal of this thesis is to examine how software tools can support an STPA analysis and potentially generate requirements automatically, rather than performing the analysis completely with tools alone because "the most effective safety-critical tools will assist rather than replace analysis by humans" [33]. Specifically, the objective of this thesis is to answer the research questions below:

- How can tools support hazard analysis using STPA?
- Can tools provide more guidance and checks during the hazard analysis? To evaluate the STPA tool, this thesis applies it to two new features—emergency braking assist and keyless ignition—that will be integrated into an existing system.
- How can tools help support requirements engineering when STPA is used? This includes how tool can assist the evaluation of completeness and the consistency of system specifications in early stages of the system development [10][11]. To provide more support for requirement engineering of the development of software-intensive systems, the thesis also investigates the potential of tool integration with other commercial toolsets.

The thesis is organized based on these objectives: Chapter 2 provides a survey of the state-of-the-art techniques used in hazard analysis and requirements engineering; Chapter 3 describes a prototype tool that was developed to support hazard analysis using STPA, generation of safety requirements, and the identification of conflicts arising from feature interactions; Chapter 4 applies the STPA tool to new automotive features—emergency braking assist and keyless ignition. In particular, this thesis illustrates how the tool can help in creating safety requirements based on SpecTRM-RL. With tool assistance, an integrated analysis is conducted in Chapter 5 to identify conflicts between new features and existing ones. The integrated analysis is based on the results of independent STPA analyses in Chapter 4. The results from the integrated analysis are also presented to show how violations of design assumptions can lead to conflicts and solutions for resolving these conflicts. Chapter 6 concludes by making suggestions for the system architecture of the automotive system based on the analyses results in the previous chapters.

2 Literature Review

2.1 Overview

This chapter briefly describes the techniques and toolsets used for hazard analysis. It starts by summarizing ISO 26262, the functional safety standard used in automotive industry. A review of traditional safety engineering methods including FMEA and FTA in ISO 26262 is given. Then, the chapter introduces the new hazard analysis technique STPA and its related accident causality model (i.e., STAMP). Given one of the objectives of this thesis is to illustrate how to generate safety requirements automatically from STPA analysis, SpecTRM [15], a toolset that supports the organization and documentation of safety requirements, is also presented.

2.2 Safety Engineering Methods in ISO 26262

ISO 26262 is an adaptation of functional safety standard IEC 61508 but specific to the automotive industry. It addresses safety-related automotive systems comprised of electrical, electronic, and software components [19]. Specifically, "ISO 26262 provides a standard for functional safety management for automotive applications, defining standards for overall organizational safety management as well as standards for a safety life cycle for the development and production of individual automotive products" [20]. As Hommes [19] suggests, ISO 26262 departs "from safety as an after-thought" and provides the framework and vocabulary for hazard elimination. It includes reliability engineering methods such as: Hardware Architecture Metrics, FMEA and FTA, etc. However, it is limited in providing guidance on how to identify and eliminate hazards in design.

As mentioned before, the increased use of electronic and digital components in automotive systems makes the vehicle much more complex. As a result, the number of recalls related to design flaws of software has been increasing dramatically [45][44]. Methods for building a reliable system become ineffective when dealing with systems that are software-intensive and involve interactions between human and automations. A review of these methods is critical to the understanding of the limitations of traditional methods and why we need STPA and tools that support related engineering processes to deal with complex systems in the automotive domain.

2.3 FMEA and FMECA

Failure mode and effect analysis (FMEA) was first developed to assess equipment reliability for military use [23]. It starts with failures of individual components and then checks what influences these failures will have on the system as a whole. Failure mode effects and criticality analysis (FMECA) extends FMEA by conducting a criticality analysis that "ranks each postulated failure mode according to the criticality of the effect on system operation and the probability of its occurrence" [23].

FMEA looks at single component failures and can help improve system reliability by selecting a design with a high probability of successful operation [23]. However, it doesn't consider accidents caused by multiple failures or unsafe interactions among components. Consider the recall related to keyless ignition and shift-by-wire, where pressing the push button three times (within three seconds) when the vehicle transmission is in "drive" or "reverse" and the speed is more than ten kilometers per hour can put the vehicle in "neutral" range automatically. In this case, no component failed. In fact, the shift-by-wire system worked as designed, by shifting to "neutral" automatically when the driver pressed the "Engine Start/Stop" button three times under the conditions above. This was regarded as an "emergency function" to make sure that the vehicle could be moved out of the "danger zone" [8]. Sotomayor [21] did a case study of a steer-by-wire system in which a comparison was made between FMEA and STPA. STPA found more causal scenarios that were missed by FMEA.
[The same type of limitations about human and software components of systems applies here.]

2.4 FTA

Fault tree analysis (FTA) is a top-down and deductive approach for fault (failure) analysis. It first defines the top-level event as the undesired condition of the system and then performs decompositions to create lower-level events using Boolean logic. It was initially applied to safety in the intercontinental ballistic missile (ICBM) and then used by the Boeing Company for manned aircraft design [24]. It involves six steps [24]:

- Define hazardous event: This is the event that can lead to loss of life or properties and must be prevented from happening.
- Acquire understanding of the system: Engineers should get a basic understanding of how the system operates and its intended use.
- Construct fault tree: It starts with the top-level hazardous event that has been defined and searches backward. At each level, all possible failures or faults that can trigger the one at

the higher level are identified. The "AND" or "OR" gate applies Boolean logic to the combination of failures and faults at the level below. "AND" means that the higher-level event will occur only if all of the events happen at the lower-level, while "OR" indicates that the event at the higher-level can happen if any of the lower-level ones occur. The ultimate goal is to identify all the possible combinations of failures and faults that can lead to the top-level event.

- Collect quantitative data: The purpose of this step is to determine the probability of the occurrence of the tree nodes and thus calculate that of the top-level hazard. These probabilities are often decided by previous experience and tests after the system is built. A challenge is verification of failure probability assumption of a brand new component, possibly to be manufactured in a new factory process.
- Evaluate fault tree probability: After probabilities have been assigned to each event at the lowest level (i.e., leaf), the probability of the top-level event can be determined by considering all paths in the tree.
- Analyze results: Although this technique can be applied to sub-systems during any phase of the system development, "the full merits of this technique are realized, however, only after the preliminary design phase, when the system reaches the component level" [24].

Although fault tree analysis follows a top-down procedure and has rigorous logical foundation, it is ineffective in dealing with complex systems for several reasons.

First, it provides little guidance to identify the causal factors leading to accidents. The quality of the fault tree can vary greatly depending on the expertise that engineers have in using FTA. "Both the model of the system being used by the analyst and the analysis itself are only in the analyst's head" [4].

Second, similar to FMEA, it focuses on component failure and assumes independence between elements. Take the power supply in the vehicle as an example. Insufficient power for assisting steering or braking can result from failure in the battery, the engine, or other linkages that transfer energies. However, as will be seen in Chapter 3, the interaction between new features related to the engine start and stop can also lead to the vehicle being shut down.

Third, FTA is ineffective in dealing with "human error" and with software. Fault tree analysis treats humans just like mechanical equipment by assigning a probability of "being able to perform

a certain required function" [24]. Often, these data are from simulations or predictions. However, as the surrounding environment changes, assumptions made during training become invalid. In fact, human error is not random and is influenced by the interplay between mental skills and physical abilities and by the environment in which the behavior occurs. Software does not "fail" and trying to determine all the causes of software misbehavior is impossible.

2.5 SpecTRM and SpecTRM-RL

SpecTRM is designed to assist in the development of software-intensive safety-critical systems. It provides support for intent specification [14], an improved way of organizing system specifications, and SpecTRM-RL, a formal language that describes the black-box behaviors of system components. All safety-related information is traceable in SpecTRM—i.e., rationales and assumptions for each design decision are included and can be linked to system-level goals and hazards. This is critical for the development of complex systems in the automotive domain because integrating multiple features can lead to conflicts between them.

SpecTRM-RL serves as an unambiguous interface between system engineers and components [15], making it an appropriate tool for integrated analysis of multiple-feature systems and a language of communication between different component designers and system integrators. In addition to symbolic or tabular notations, SpecTRM-RL also provides "the ability to create and hierarchically decompose a graphical model of the system" [16]. After engineers construct the SpecTRM-RL model of the system, it can be executed in SpecTRM for different types of formal analysis such as completeness and consistency checking [33][12].

Thomas [31] provides an example of the safety requirements described in a SpecTRM-RL table, as shown in Figure 2. The design logic of the train door controller is represented in the AND/OR table [31]. If any of the columns evaluate to true (OR), then the door controller must provide *Open*. Each column represents a condition for the logic to be true (AND). For example, the first column on the right describes that the controller must provide *open* when the train comes to a stop and is aligned with the platform, while the second one specifies that the same command must be provided if the train is stopped with an emergency being detected.

Although SpecTRM-RL provides a way to document results of an STPA hazard analysis and is executable for formal checking [12], it gives little guidance for performing STPA. Also,

SpecTRM cannot automatically generate safety requirements from an STPA analysis. A toolset that provides assistance in these aspects above can make engineers' work in hazard analysis and requirement engineering more efficient.

Train controller provides *"Open"*

			OR	
	Door State =	Not closing on person		
		Closing on person		
AND	**Train Position =**	Aligned with platform	T	
		Not aligned with platform		
	Train Motion =	Stopped	T	T
		Train is moving		
	Emergency =	No emergency		
		Emergency exists		T

Figure 2: An example of the SpecTRM-RL requirements for SCM

2.6 STAMP

Systems Theoretic Accident Model and Processes (STAMP) is an accident causality model based on systems theory. Developed by Professor Nancy Leveson at MIT, it aims to capture more causal factors (e.g., design flaws in the software, human errors and management and organizational factors, etc.) in addition to single component failures. Rather than considering accidents as the result of component failures or a chain of failures or faults, it treats safety as a control problem: accidents occur due to the violation of safety constraints enforced by the controller.

There are three basic concepts in STAMP: *safety constraints*, *safety control structure* and *process model*. Under these definitions, events can only lead to losses and incidents when safety constraints are not adequately enforced. For example, one constraint for vehicle operation is that the vehicle must be put into the "park" range before the driver turns off the engine and leaves the vehicle: otherwise, the vehicle may roll unexpectedly and damage itself or cause injuries to the driver or pedestrians nearby. The second concept, safety control structure, derives from systems theory in which "systems are viewed as hierarchical structures" [4]. To achieve safety goals, each level enforces safety constraints on the level below. Figure 3 is an example socio-technical control structure. The controller at each level has a downward reference channel to issue commands to processes in the level below, while it receives feedback from the upward channel.

The third concept, process model, is used by a controller for deciding whether or not it should issue commands. It represents a controller's understanding of the current status of the system and how it will operate under certain conditions. For example, the driver may use the feedback information about the vehicle motion, the transmission range and the engine status to decide whether he should turn off the vehicle when parking.

Process models play an important role not only in understanding why accidents occur, but also in designing safer systems. According to Leveson [4], "accidents often occur when the process model used by the controller does not match the process", as a result of which:

- The control actions necessary to enforce the associated safety constraints are not adequately enforced, leading to a hazard;
- Necessary control actions are provided but at wrong time leading to a hazard;
- Unsafe control actions are provided that cause a violation of the safety constraints;
- Appropriate control actions are provided but not followed, leading to a hazard.

STAMP is built on these three concepts above. In STAMP, systems are viewed as "interrelated components kept in a state of dynamic equilibrium by feedback control loops" [4]. Accidents result from "flawed processes that involve interactions among people, societal and organizational structures, engineering activities, and physical system components that lead to violating the system safety constraints. The processes leading to accidents is described in STAMP in terms of an adaptive feedback function that fails to maintain safety as system performance changes over time to meet a complex set of goals and values" [4].

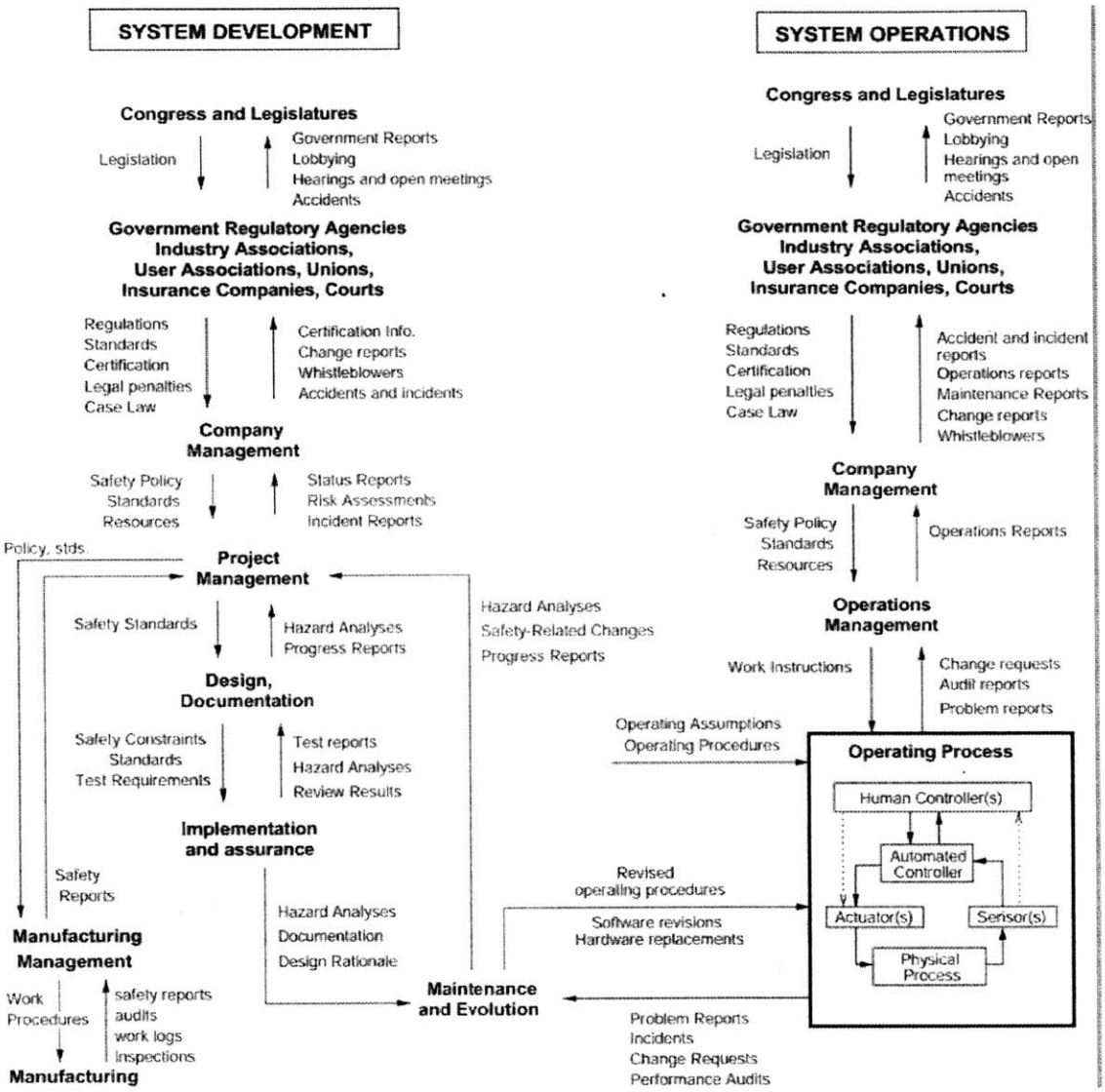


Figure 3: An example safety control structure of a socio-technical systems

2.7 STPA

STPA [4] is a new hazard analysis technique for complex systems in which hazards are caused by unsafe interactions between components (including humans) as well as component failures and faults. It is based on STAMP. It differs from traditional hazard analysis approaches in that it captures emergent behaviors that result from component interactions. It can also assist hazard analysis early in the conceptual design stage during system development, making it a powerful tool for early evaluation of safety impact led by feature integration.

STPA uses fundamental concepts in system engineering activities including accidents and losses, hazards, safety requirements and constraints, and the safety control structure. In particular, a hierarchical safety control structure is used to represent the functional components and the interactions (i.e., control actions and feedback) among them. Each hierarchical level can be regarded as a control loop in which the controller imposes safety constraints on the process being controlled in the lower level. Often, the controller issues commands through actuators and receives feedback from sensors, based upon which it updates the process model of the controlled process and makes decisions. If the controller is a human, he or she must also have a model of the automatic controller to make predictions about the automation behaviors.

There are two steps in STPA. At Step 1, unsafe control actions are identified and classified into four types:

- A control action required for safety is not provided or not followed;
- An unsafe control action is provided;
- A potentially safe control action is provided too early or too late;
- A control action required for safety is stopped too soon or applied too late.

After the unsafe control actions (UCAs) are identified, causal factors and scenarios that lead to UCAs are created based on the accident causality model described in STAMP.

It should be noted that unsafe interactions among multiple controllers could also contribute to hazards [4][40][42]. Thomas and Placke [5][18] proposed a method also to identify conflicts between different control actions arising from feature interactions and apply it to an automotive system that includes multiple features. In addition to assisting hazard analysis of each feature individually, the STPA toolset also provides support for feature interactions based on STPA. After using the STPA tool for hazard analysis of two new automotive features—emergency brake assist and keyless ignition—independently, this thesis continues to conduct the integrated analysis with tool assistance by combining these two features into an existing (automotive) system.

2.8 Summary

Given the strengths and limitations of different analysis techniques, it is necessary to choose approaches that are effective to deal with those challenges met during feature integrations. Table 2 provides a summary of different techniques/languages described in this chapter.

Table 2: Characteristics of different techniques and requirement languages

Techniques/ Languages	Categories/Characteristics
FMEA/FMECA	• Technique for hazard analysis • Based on chain-of-events model • Focuses on single component failures • Requires that designs are available • Assumes that humans and software fail randomly
FTA	• Technique for hazard analysis • Based on chain-of-events model • Identify combinations of component failures • Requires that designs are available • Assumes that humans and software fail randomly
STPA	• Technique for hazard analysis • Based on systems theoretic accident model • Considers component interactions • Includes humans and software without assuming random failure • Traceability between system goals/hazards and safety constraints • Can be applied to the early stage of conceptual development
SpecTRM-RL	• Formal requirements language • Executable and formal checking • Support for requirements reuse • Traceability between system goals/hazards and low-level requirements

As can be seen, traditional approaches such as FMEA and FTA are effective at preventing accidents resulting from component failures, while STPA can apply to accidents arising from unsafe component interactions as well as component failures. For these reasons, STPA and SpecTRM-RL are chosen to deal with those challenges presented in Chapter 1. Chapter 3 will introduce a software prototype that is built by the author and the research team to support hazard analysis based on STPA and requirement generation.

3 Tool-Assisted Hazard Analysis and Requirement Generation

This chapter describes an open-source software prototype for hazard analysis and requirements generation based on STPA in the Systems Engineering Research Laboratory (SERL) at MIT [29]. The chapter consists of five parts. It starts by introducing the characteristics and the architecture of the STPA tool. Then, the tool implementation for partially automating STPA and the guidance (checks) it provides for hazard analyses are presented. To further facilitate hazard analyses, this tool is capable of generating safety requirements automatically based on STPA Step 1 results. In addition, it also supports integrated analysis of multiple control systems, which is important for successful feature integrations. Finally, it concludes by discussing some reflections on building this prototype.

3.1 An Overview of the STPA Tool

The STPA tool is based on the outline presented by Thomas and Suo [29]. The Eclipse integrated development environment [38] was used to create the tool, as shown in Figure 4. One benefit of using this architecture is to make the tool extensible—i.e., it can be integrated into SpecTRM, a commercial tool designed to assist in the development of software-intensive safety critical systems and focuses on system requirements and specification.

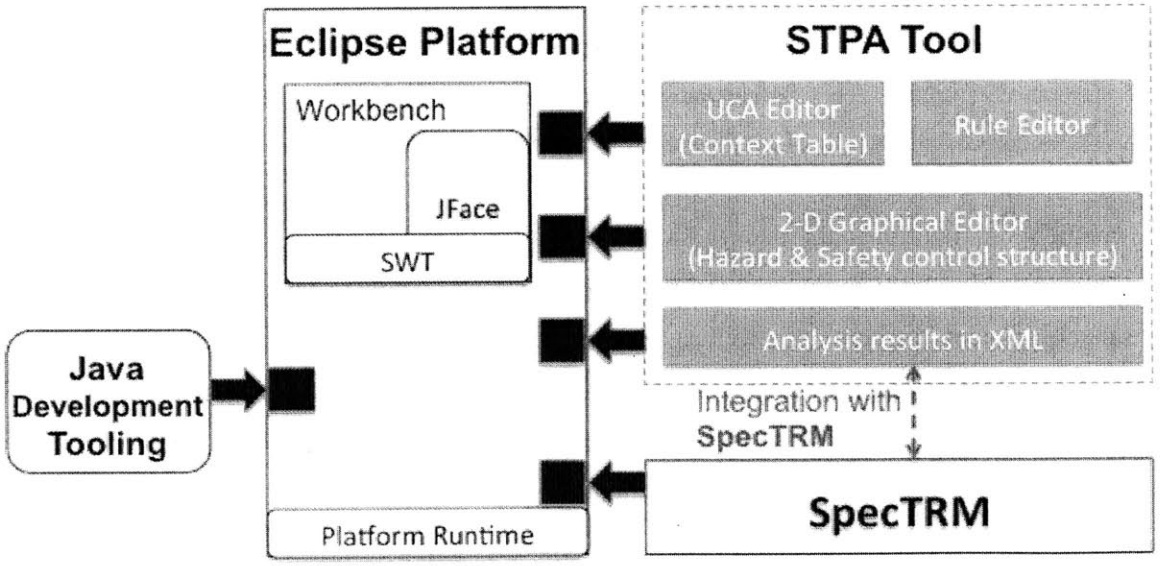


Figure 4: The architecture of the STPA tool

As mentioned before, this STPA toolset is created to explore the partial automation of STPA and the generation of requirements, and to provide assistance for the analysis of multiple control systems. Specifically, it guides engineers through the process of [29]:

- Specifying hazards
- Drawing the safety control structure
  - Adding controllers, controlled processes
  - Adding actuators and sensors
  - Adding control actions and feedback
  - Adding process model variables and values
- Performing STPA Step 1
  - Generating context table templates automatically based on control structure
  - Allowing engineers to specify unsafe control actions
  - Allowing engineers to define "Rules", which are used to automatically complete context tables and identify unsafe control actions
- Storing analysis results in XML files for interoperation with other tools
- Generating safety requirements for completeness and consistency checking in SpecTRM
- Performing integrated analysis of multiple controllers

The tool assistance for performing STPA Step 2, i.e., the generation of causal scenarios, is planned for the future.

3.2 Tool Assistance for STPA (System Engineering Foundations and Step 1)

The implementation of the tool assistance of STPA is based on recent research by Thomas that develops a formal framework for STPA, including algorithms that could be used to automate STPA Step 1 and generate executable requirements [31].

A 2-D graphical editor is included in the tool to support the basic system engineering foundations,i.e., defining hazards/goals and drawing safety control structure, as shown in Figure 5. The engineer can then choose components,i.e., controllers, processes, actuators or sensors, that he or she wants to add to the control structure by clicking on the corresponding labels in the tool bar at the center of the editor.

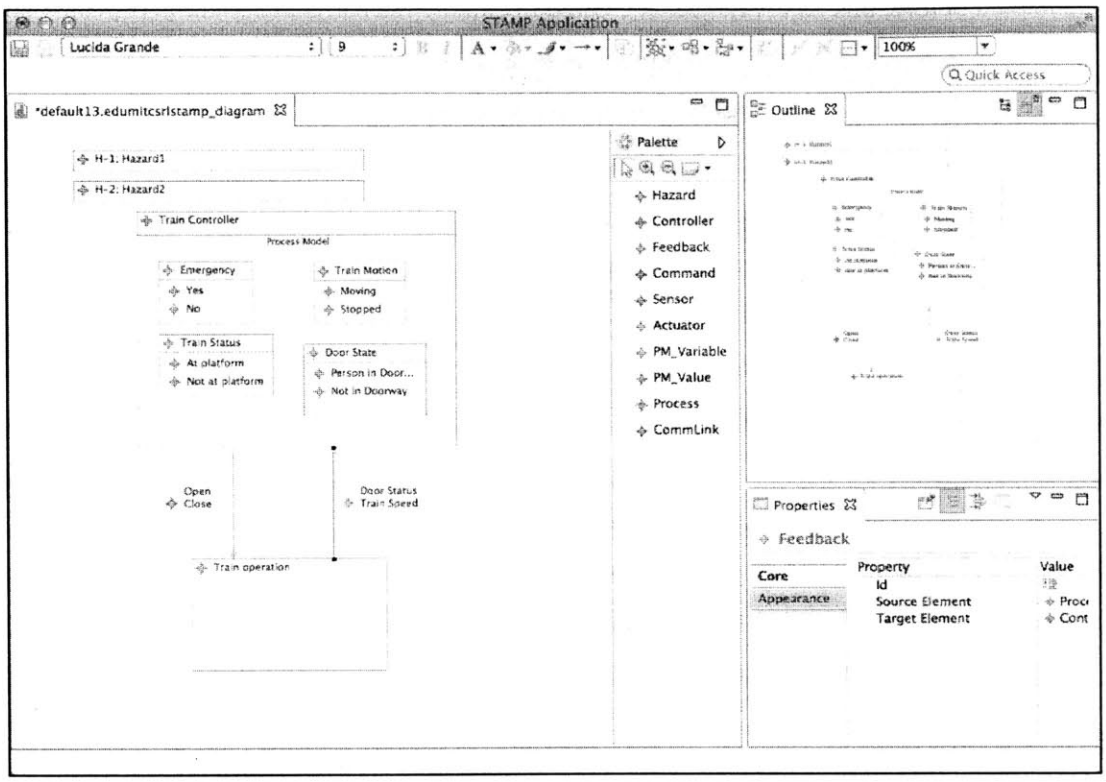


Figure 5: 2-D editor for specifying hazards and drawing safety control structure

After adding control actions to the control structure, the engineer can start to perform STPA Step 1 to identify unsafe control actions (UCAs). Our tool can also create context tables automatically to formalize UCAs [31]. Figure 6 illustrates the context tables for two control actions. The left column, the control action list, includes all control actions for the controller, while the right column displays the context table for the control action engineers' choice from the list on the left. Each row in the context table describes a condition in which a control action is provided or not provided. The engineer can brainstorm to decide whether providing or not providing that control action can cause hazards. The detailed UCAs after Step 1 can then be displayed in plain English as shown in Figure 7.

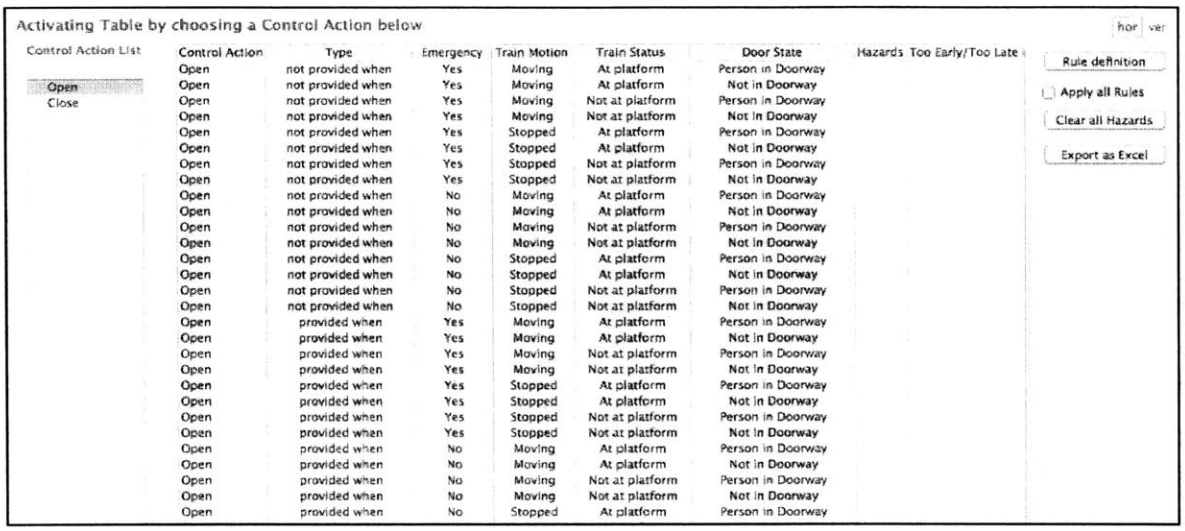


Figure 6: Context table created automatically

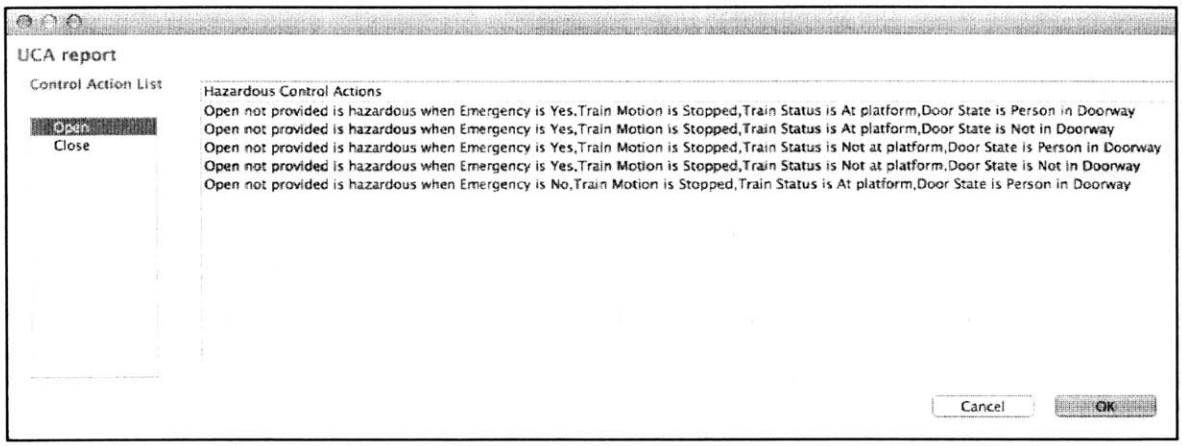


Figure 7: Detailed UCAs generated automatically from context tables

3.3 Guidance and Checks by STPA Tool

For complex systems, the size of the context table (i.e., the number of rows) can become large and engineers have to deal with a large number of variables when identifying detailed UCAs. Thomas [31] suggests that "Rules" be defined to simplify this process. As can be seen in Figure 8, each rule can be used to create multiple UCAs.

Rule 1: **Door open provided** is hazardous when train is moving

Context id	Control Action	Type	Train Motion	Emergency	Train Position	Hazardous
Context1	Door open	Provided	Moving	Yes	Not at Platform	Yes
Context2	Door open	Provided	Stopped	Yes	Not at Platform	
Context3	Door open	Provided	Moving	Yes	At Platform	Yes
Context4	Door open	Provided	Stopped	Yes	At Platform	
Context5	Door open	Provided	Moving	No	Not at platform	Yes
Context6	Door open	Provided	Stopped	No	Not at platform	
Context7	Door open	Provided	Moving	No	At Platform	Yes
Context8	Door open	Provided	Stopped	No	At Platform	

Figure 8: "Rules" for creating context tables

The STPA tool provides an editor for defining a rule based on its English description, as can be seen in Figure 9. Similar to the UCA editor, on the left is the list of control actions. Engineers can choose parameters for defining rules (i.e., provided/not provided, process model variables/values and related hazards) from the upper part. After clicking the button on the top right, a new rule with the English description will be added to the list of rules at the bottom. After engineers add a new rule through the rule editor, the tool will be automatically applied to fill out the right column to identify detailed UCAs and safety constraints, as shown in Figure 10.

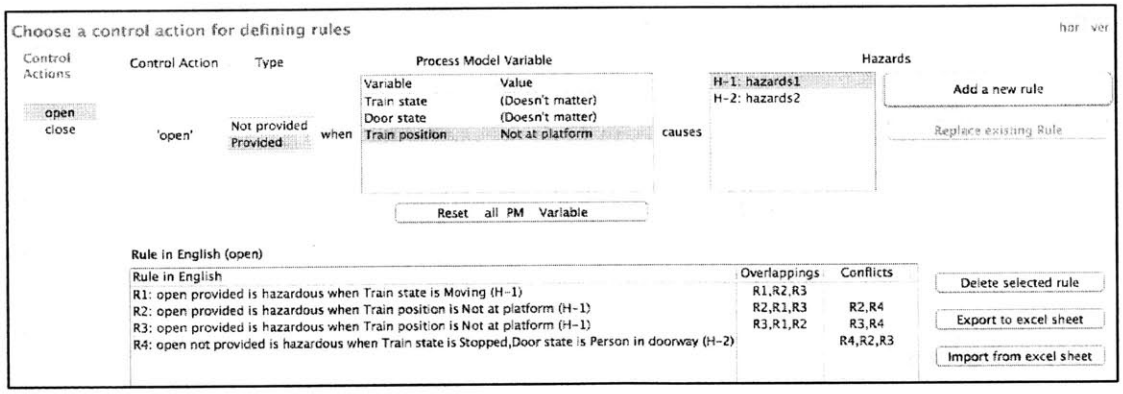
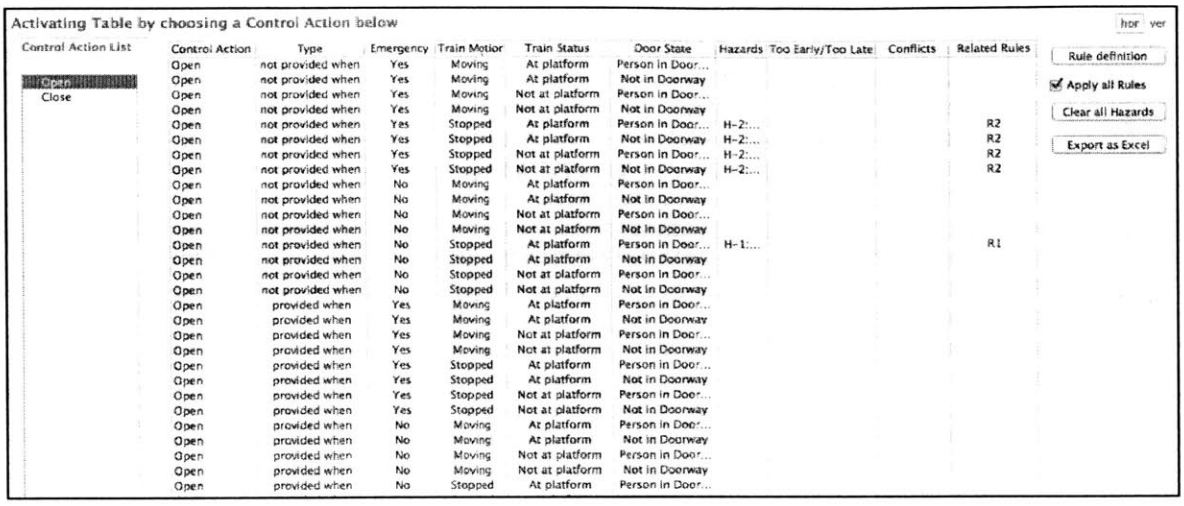


Figure 9: Editor for defining "Rules"



Control Action	Type	Emergency	Train Motion	Train Status	Door State	Hazards	Too Early/Too Late	Conflicts	Related Rules
Open	not provided when	Yes	Moving	At platform	Person in Door...				
Open	not provided when	Yes	Moving	At platform	Not in Doorway				
Open	not provided when	Yes	Moving	Not at platform	Person in Door...				
Open	not provided when	Yes	Moving	Not at platform	Not in Doorway				
Open	not provided when	Yes	Stopped	At platform	Person in Door...	H-2:...			R2
Open	not provided when	Yes	Stopped	At platform	Not in Doorway	H-2:...			R2
Open	not provided when	Yes	Stopped	Not at platform	Person in Door...	H-2:...			R2
Open	not provided when	Yes	Stopped	Not at platform	Not in Doorway	H-2:...			R2
Open	not provided when	No	Moving	At platform	Person in Door...				
Open	not provided when	No	Moving	At platform	Not in Doorway				
Open	not provided when	No	Moving	Not at platform	Person in Door...				
Open	not provided when	No	Moving	Not at platform	Not in Doorway				
Open	not provided when	No	Stopped	At platform	Person in Door...	H-1:...			R1
Open	not provided when	No	Stopped	At platform	Not in Doorway				
Open	not provided when	No	Stopped	Not at platform	Person in Door...				
Open	not provided when	No	Stopped	Not at platform	Not in Doorway				
Open	provided when	Yes	Moving	At platform	Person in Door...				
Open	provided when	Yes	Moving	At platform	Not in Doorway				
Open	provided when	Yes	Moving	Not at platform	Person in Door...				
Open	provided when	Yes	Moving	Not at platform	Not in Doorway				
Open	provided when	Yes	Stopped	At platform	Person in Door...				
Open	provided when	Yes	Stopped	At platform	Not in Doorway				
Open	provided when	Yes	Stopped	Not at platform	Person in Door...				
Open	provided when	Yes	Stopped	Not at platform	Not in Doorway				
Open	provided when	No	Moving	At platform	Person in Door...				
Open	provided when	No	Moving	At platform	Not in Doorway				
Open	provided when	No	Moving	Not at platform	Person in Door...				
Open	provided when	No	Moving	Not at platform	Not in Doorway				
Open	provided when	No	Stopped	At platform	Person in Door...				

Figure 10: Apply Rules to context table for identifying detailed UCAs

To create complete and correct safety constraints, it is critical to ensure that rules are devoid of missing UCAs and conflicts. The STPA tool provides guidance to engineers during this process by performing checks automatically.

The context table can help in deciding if there is a missing UCA. As shown in Figure 10, a cell in the last column specifies if any rule is applied to identify a UCA in the current row. No related rule suggests providing (or not providing) the control action will not cause hazards, or engineers missed a UCA. For example, the first row in Figure 10 describes the moment when the train is at the platform, has just started to move, and a person is still in the doorway. In this situation, not providing the *open* command can cause injuries to passengers. However, no rule in the last column suggests that there is a UCA. Therefore, engineers may decide a missing UCA is identified and new rules need to be added into the rule editor.

In addition, the rule editor in Figure 11 provides assistance for identifying conflicts between two rules. As can be seen, R1 specifies that providing the *open* command when the train is moving can lead to hazards, while R2 represents the situation that not providing the *open* command can also lead to a hazard if there is a passenger caught in the door. The safety constraint generated from R1 would require the controller to provide the *open* command, while the constraint from R2 would specify that the *open* command must not be provided in the same context. Obviously, the rules conflict in the sense that the system will enter hazardous states no matter what action the controller takes. This conflict is displayed in the last column of the table. The formal structure of conflicts and overlaps used in the tool implementation is also given below.

Rule in English (Open)		
Rule in English	Overlappings	Conflicts
R1: Open provided is hazardous when Train Motion is Moving (H-2)		R1,R2
R2: Open not provided is hazardous when Door State is Person in dooway (H-1)		R2,R1

Figure 11: Conflicts automatically detected by the STPA tool

Engineers can then resolve conflicts between rules by adding more detail to the context of the rule description. For example, if R2 is modified to include only those conditions where the train has been stopped, it will no longer be in conflict with R1, as shown in Figure 12.

Rule in English (Open)		
Rule in English	Overlappings	Conflicts
R1: Open provided is hazardous when Train Motion is Moving (H-2)		
R2: Open not provided is hazardous when Train Motion is Stopped,Door State is Person in dooway (H-1)		

Figure 12: Rule definition with conflicts resolved

Mathematical structures of Rules and conflicts used in the algorithms of the STPA tool are given below.

Definition 1. A **Rule** is a five-tuple $< Ce, Ca, C^T, Cg, Ha >$ where $Ce = \{Ce_1, Ce_2, ...Ce_i\}$ is a set of controllers which give commands; $Ca = \{Ca_1, Ca_2, ...Ca_i\}$ is a set of control actions involved; $C^T =< Ca, T >$ is a two-tuple, which represents the type of each control action in Ca, where $T = \{Provided, Not\ Provided\}$; $Cg \in Co$ is the subset of the whole contexts; $Ha = \{Ha_1, Ha_2, ..., Ha_i\}$ is a subset of hazards related to this Rule.

Definition 2. A **conflict** is a triple $< Co_i, L_{RP}, L_{RNP} >$ where Co_i is a context; $L_{RP} = \{R_1, R_2, ..., R_i\}$ is a set of Rules with the "provided" type that apply to context Co_i, $L_{RNP} = \{R_1, R_2, ..., R_i\}$ is a set of Rules with the "not provided" type that apply to context Co_i.

3.4 Tool Assistance for Requirement Generation

In addition to providing guidance for hazard analysis, a tool for hazard analysis can provide traceability between hazard analyses and system requirements if it is capable of generating safety requirements automatically based on STPA Step 1 results. Figure 13 shows the SpecTRM-RL requirements generated by the tool from the original UCAs list in Figure 7.

Our STPA tool seeks to improve the efficiency of hazard analyses by conducting logical simplifications so that engineers do not have to work on low-level SpecTRM-RL. The size of generated SpecTRM-RL tables will be automatically reduced to the simplest form.

Control algorithm for 'Open' cmd

Emergency=	Yes		T
	No	T	
Train Motion=	Moving		
	Stopped	T	T
Train Status=	At platform	T	
	Not at platform		
Door State=	Person in Doorway	T	
	Not in Doorway		

Figure 13: SpecTRM-RL table generated automatically from Step 1 results after logical simplification

As mentioned before, one of the goals of the STPA tool is to provide more guidance and checks to engineers during the early stage of the system development. Engineers who use our tool can store analysis results and SpecTRM-RL requirements as XML files that can be exported to SpecTRM for consistency and completeness checking, as shown in Figure 14.

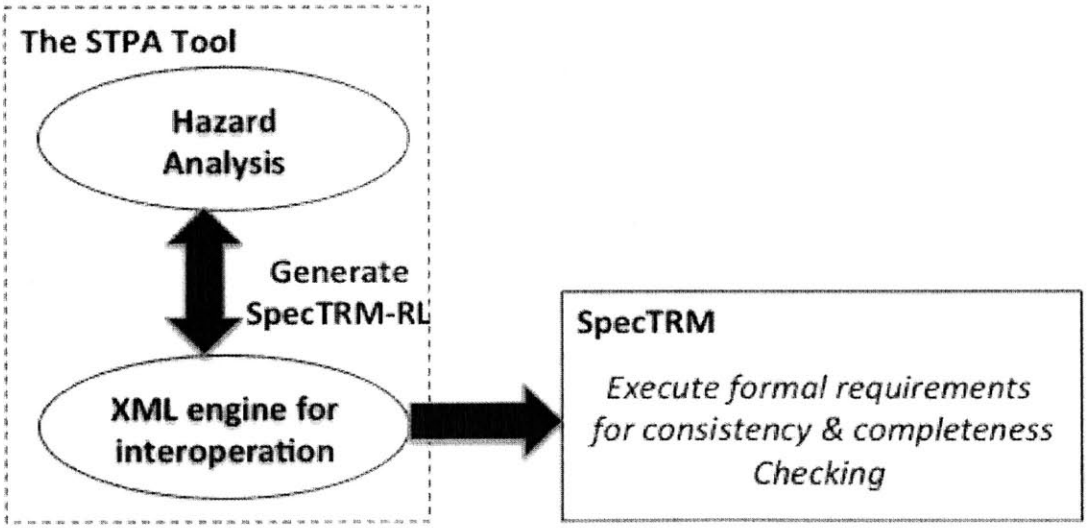


Figure 14: Tool integration with SpecTRM

3.5 Tool Assistance for Integrated Analysis of Multiple Control Systems

Thomas and Placke [18] propose a method to identify conflicts between multiple control systems with feature interactions based on results from STPA. This section discusses our tool assistance for this method and how to use results generated from automated tools to guide the design process.

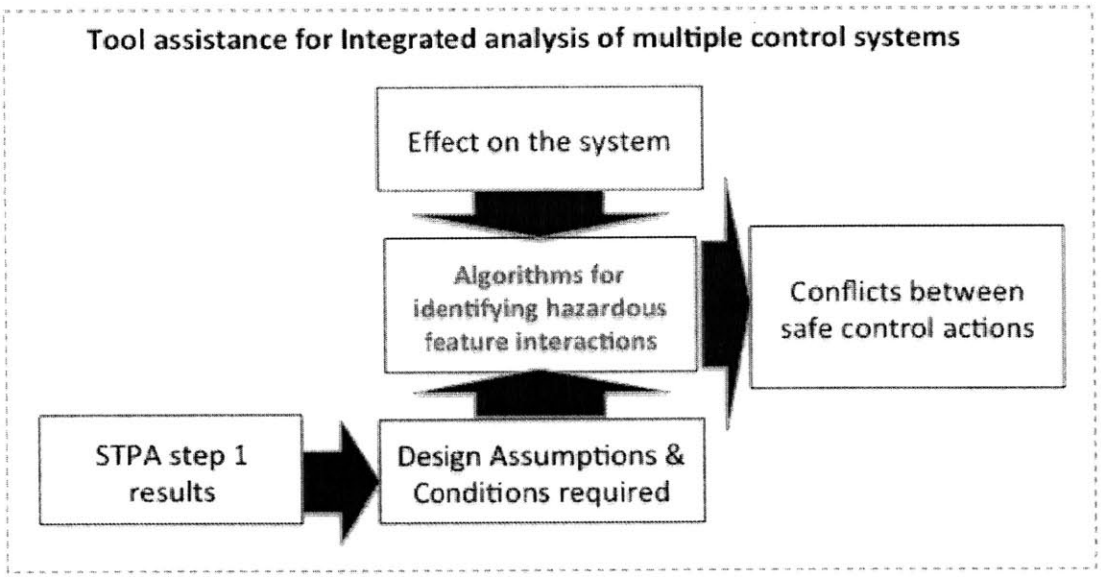


Figure 15: The process of identifying hazardous feature interactions

Our tool supports this process in three aspects (Figure 15):

- Generating design assumptions and conditions required for providing safe control actions based on STPA Step 1 results automatically, as shown in Figure 16. As Placke [5] suggested, STPA Step 1 results can be leveraged to create those assumptions for each control action that are necessary to achieve safety or functional goals.
- Providing an editor that allows engineers to define the effect of providing safe control actions on the system. As will be discussed later, issuing a command may have multiple influences on the system.
- Detecting conflicts between control actions by different controllers automatically. According to Thomas and Placke **Error! Reference source not found.**, one controller issuing a control action may lead to another controller issuing an unsafe control action. Conflicts can be identified by examining the effects resulting from issuing each command and comparing them with the conditions required by another. For complex systems, examining the large number of control actions and their design assumptions can be time-

consuming, but automated tools can improve the efficiency of the integrated analysis by detecting conflicts automatically. An algorithm that is scalable to integrated analysis of more than three controllers has been developed by Thomas and is implemented in our STPA tool. Figure 17 shows the editor that displays conflicts detected between two control actions. Chapter 4 will discuss how these generated results can provide more guidance to engineers during the design process.

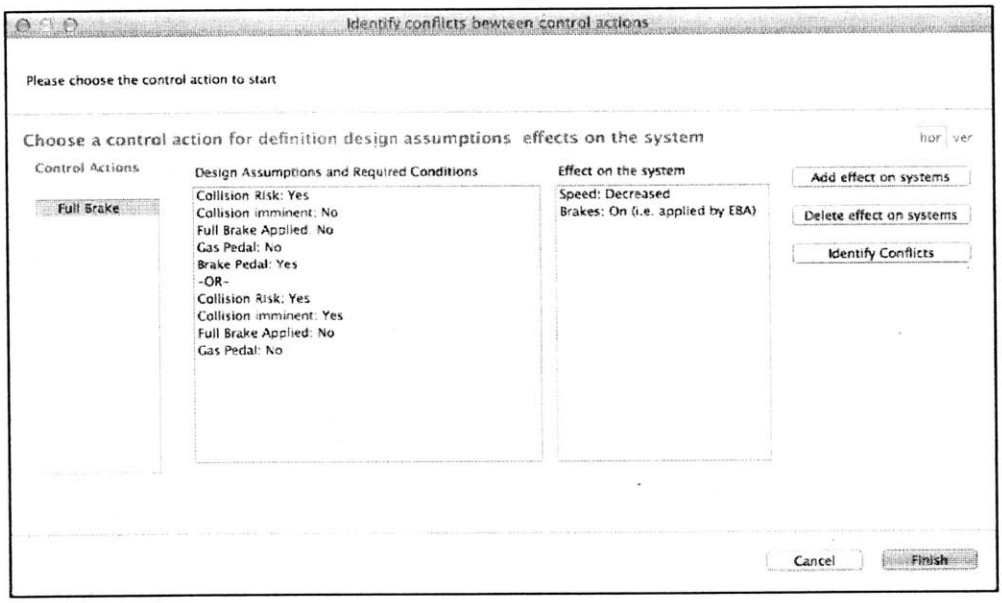


Figure 16: Interface for deciding design assumptions and effects on the system of safe control actions

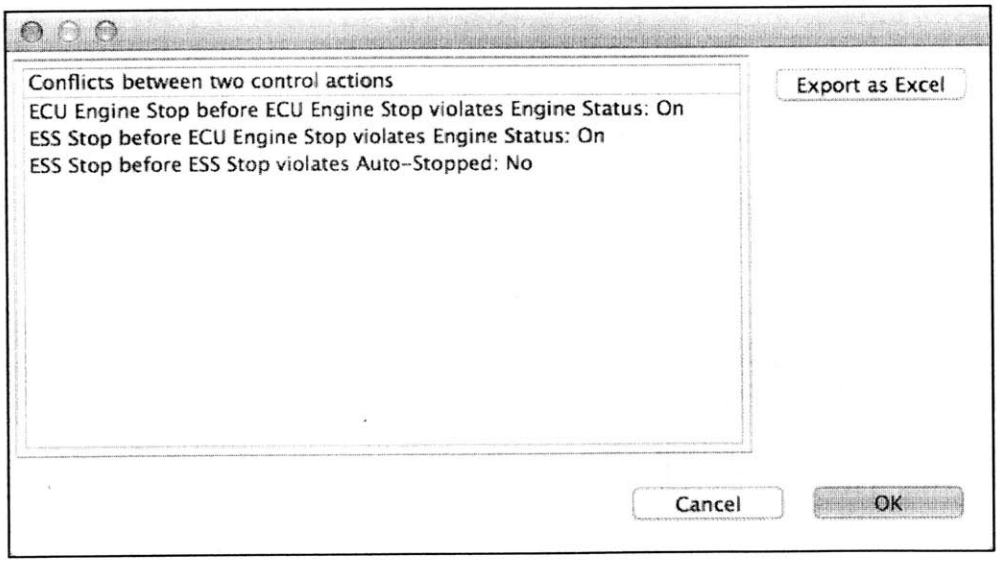


Figure 17: Conflicts detected by automated tools

3.6 Reflections on Building the STPA Tool

According to Modugno and Leveson [32], studying the process of how experts learn and use the analysis technique will help gain understanding of "how automation can be used to lessen the cognitive demands of the particular analysis technique". To evaluate the concepts of tool-assisted hazard analysis using STPA, real engineers were given a copy of the prototype tool to apply on real projects. Based on the feedback received, we suggest that three aspects be considered during the tool development.

First, allowing engineers to represent and model UCAs in multiple forms (languages) can help them learn STPA and make the hazard analysis more efficient. As Dulac, Viguier, and Leveson [34] suggest, "multiple visualizations generated from a common model will improve the requirements creation, reviewing and understanding process". Figure 18 illustrates three representations used in the STPA tool that assist engineers in creating UCAs and safety requirements. The characteristics of the three types of representation are also summarized in Table 3. For beginners or engineers who have not received training in formal logic, they can write UCAs with English descriptions instead of working on the context tables. For engineers who need to work on Excel files for permanent storage, they can export Step 1 results as context tables. For those who want to perform formal checking on requirements, SpecTRM-RL can be generated directly from STPA Step 1 results: it has rigorous mathematical foundations and is executable in automated tools.

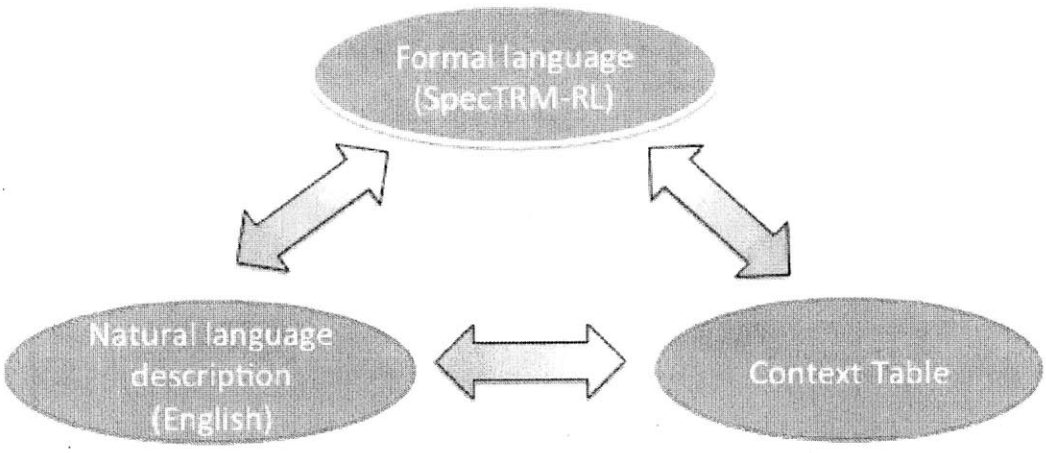


Figure 18: The interchange among three forms for describing UCAs

Table 3: Characteristics of three forms of language in the STPA tool for Step 1 and requirement generation

Forms of UCAs description	**Characteristics**
English description	• Easy to start for STPA beginners
Context table	• For engineers who have received training in formal logic • Can be exported to Excel files for storage
SpecTRM-RL	• Easy to learn with much less time for training • Executable and formal checking • Hierarchical modeling and scalable to complex systems

Second, tools should provide traceability between system-level goals and hazards, UCAs, safety constraints and causal scenarios, etc., so that engineers can be aware of the influence of design changes on the analysis results. Given that STPA is an interactive process [4], engineers often start the analysis at a high level during the early conceptual stage. They may then make changes in terms of hazards defined, control structure and process models for each controller as more design decisions are made. Take Step 1 as an example: analysis may be incomplete in the sense that certain conditions in which the control action is provided are missing and new variables must be added to the safety control structure. In this case, the contextual assumptions based on which UCAs are decided can be violated. If the engineer wants to check if an old analysis can still be used, instead of starting over from scratch, the results must be traceable so he or she can decide if providing or not providing the control action in a new but modified context can cause hazards or may violate system goals in the higher level.

Third, in addition to the traceability mentioned above, knowing rationales and assumptions for each safety requirement deriving from hazard analyses is important. As discussed in the first chapter, causal factors for hazards arising from feature interactions can be traced back to the violation of design assumptions. Often, each design team may be responsible for the development of a particular feature individually and rationales based on design decisions made by one team can be lost and ignored by another. Therefore, developers of tools for hazard analyses and

requirements generation should also take into account the documentation and the organization of as design assumptions and rationales for analysis results and generated requirements. We suggest that *intent specification* be used, since it includes all those characteristics discussed above. In fact, this is the main reason why we examine the potential of tool integrations with SpecTRM, a toolset that supports *intent specification*.

4 Application of the STPA Tool: A Case Study of Automotive Systems

This chapter demonstrates how the STPA tool can be applied to analyze automotive systems with multiple features, as shown in Figure 19. The automotive systems described in this chapter are fictional and do not reflect any specific manufacturer's design. They were created to be realistic in terms of overall complexity and representative of common functionalities being introduced into the automotive industry.

First, STPA is performed separately on keyless ignition and emergency braking assist to create high-level safety constraints and causal scenarios. Then, the STPA tool is used to refine and formalize the UCAs to generate SpecTRM-RL requirements for these two controllers. After the analysis of each new feature is finished individually, the STPA tool is used to perform integrated analysis to identify unsafe interactions between two new features (i.e., keyless ignition and emergency braking assist) and three existing features that have been analyzed before by Thomas and Placke [5][18], for a total of 5 features analyzed together.

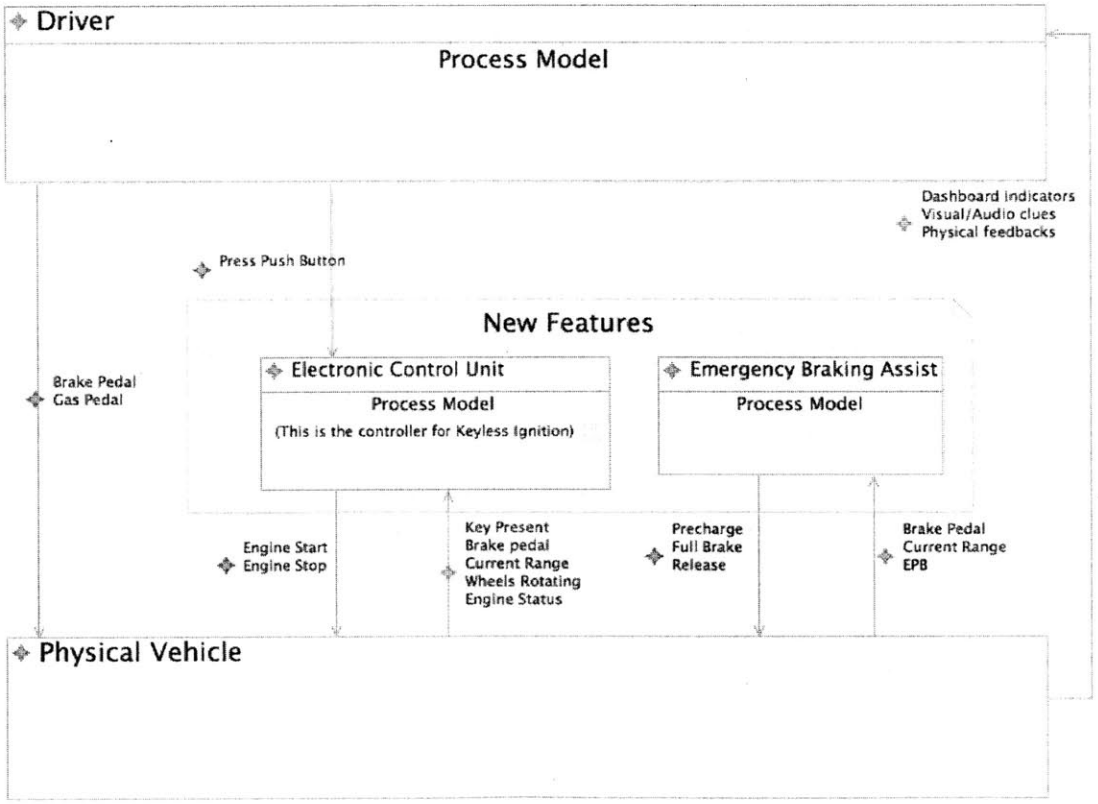


Figure 19: Keyless ignition and emergency braking assist

4.1 Tool-Assisted Hazard Analysis of Keyless Ignition

4.1.1 Initial STPA of keyless ignition

System Overview

Keyless ignition is a feature that allows the driver to start the vehicle (usually by pushing a button) without inserting the physical key into the ignition switch. Unlike the traditional ignition system, the keyless ignition system can detect the low frequency signals transmitted by the transponder within the physical key fob. Each key fob has a (unique) electronic key code embedded within it that can be detected by an electronic control unit (ECU). If the ECU decides the key code is correct, it will activate the key (i.e., electronic key code present) and disarm the immobilizer within the system that may prevent the starting of the engine. Under this condition, the starting system will start the vehicle if and only if the driver presses the start/stop button.

The architecture of keyless ignition can be different depending on actual implementation. For example, as Trinidad and Dixon [37] suggest, in addition to the ECU that monitors the detection of the electronic key, other components such as the power control unit (PCU), steering lock unit (SLU) and brake switch, etc., may also be required to enable the engine to start. This case study assumes that the ECU is the only controller in the high-level control structure that sends commands directly to the propulsion system that starts or stops the engine.

Before performing STPA Step 1, accidents and hazards are defined, as shown in Table 4 and Table 5 respectively—not only hazards that result in injuries or deaths of drivers, passengers, and pedestrians, but the violation of regulations is also considered because a lot of recalls [45] related to the starting system involve the violation of Federal Motor Vehicle Safety Standards (FMVSS) 102 and 114 [35][36].

Table 4: System-level accidents

Number	System Loss Description
A-1	Two or more vehicles collide
A-2	Vehicle collides with non-fixed obstacle
A-3	Vehicle crashes into terrain
A-4	Vehicle occupants injured without vehicle collision

Table 5: System-level hazards

Number	System Hazard Description
H-1	Vehicle does not maintain safe distance from nearby vehicles
H-2	Vehicle does not maintain safe distance from terrain and other obstacles
H-3	Vehicle occupants exposed to harmful effects and/or health hazards
R-1	Vehicle does not meet FMVSS regulations

Based on Federal Motor Vehicle Safety Standards (FMVSS), high-level requirements for the design of keyless ignition are shown in Table 6.

Table 6: System-level requirements for keyless ignition

Number	System Requirement Description
Requirement-1	According to FMVSS 102, "the engine starter shall be inoperative when the transmission shift lever is in a forward or reverse drive position." (Note that this only applies to driver activation.)
Requirement-2	According to FMVSS 114, "the starting system must prevent key removal unless the transmission or gear selection control is locked in 'park' or becomes locked in 'park' as the direct result of key removal."

Control Structure and Commands

The 2-D editor in Figure 20 represents the high-level safety control structure of keyless ignition. As shown in Figure 20, the ECU can detect if the driver has pressed the start/stop button. Then, when the start button is pressed, the ECU checks if it is safe to start the vehicle (based on feedback information such as key present, vehicle in standstill, current range, etc.) and issues commands to start or stop the engine. Two control actions provided by the ECU are shown in Table 7.

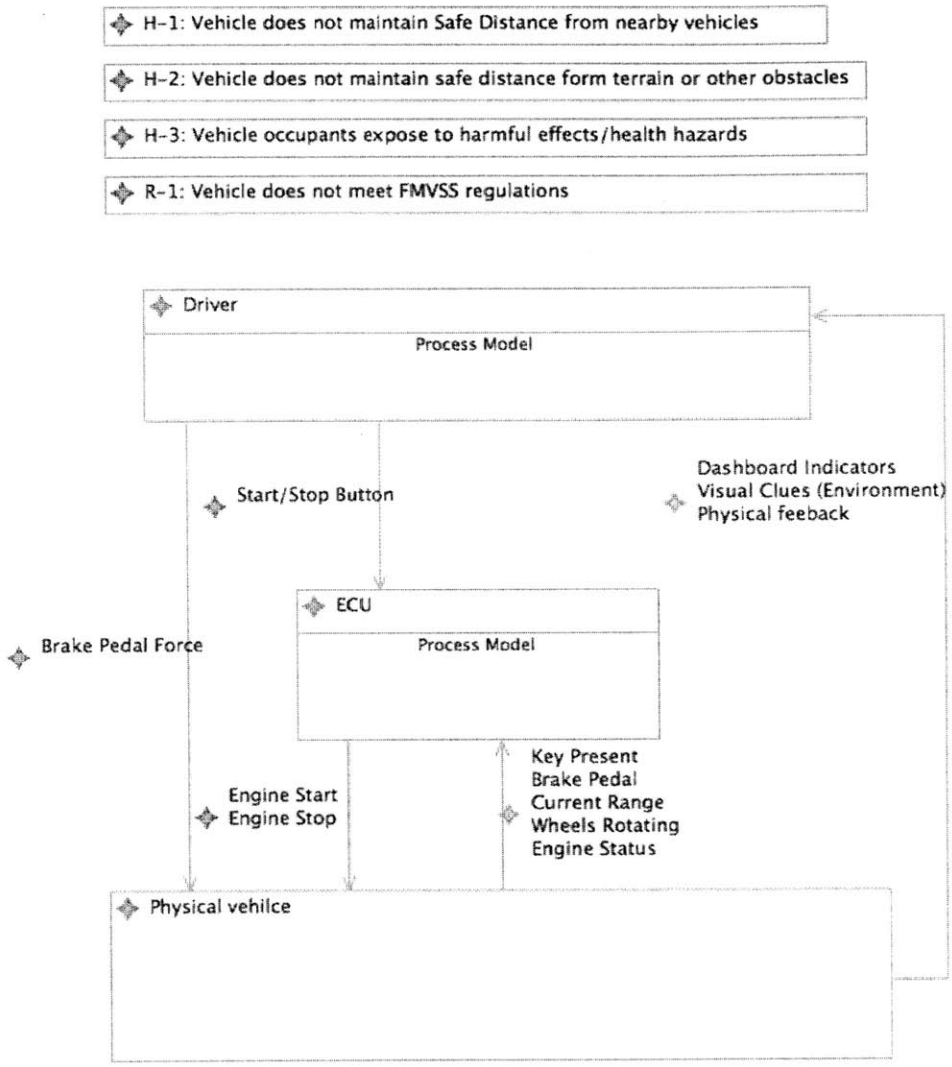


Figure 20: The high-level safety control structure of the keyless ignition

Table 7: Control actions of electronic control unit

Control Action	Description
Engine Start	The ECU sends commands to the propulsion system to start the engine.
Engine Stop	The ECU sends commands to the propulsion system to stop the engine.

Initial Unsafe Control Actions and Safety Constraints

After control actions are added to the control structure, engineers can perform STPA Step 1 to identify UCAs and create safety constraints, as shown in Table 8 and

Table 9.

Table 8: Initial UCAs of ECU

Control Action	Not providing the control action causes hazards	Providing the control action causes hazards	The timing or sequencing of control actions leads to hazards	The duration of a continuous control action leads to hazards
Engine Start		UCA-1: ECU provides Engine Start when the key is not present [R-1, H-1, H-2]	UCA-2: ECU provides Engine Start too late [H-3]	UCA-3: The duration of ECU providing Engine Start never ends [H-3]
		UCA-4: ECU provides Engine Start when the transmission is not in Park or Neutral [R-1, H-1, H-2]		
Engine Stop	UCA-5: ECU does not provide Engine Stop when the driver tries to shut down the engine (i.e., presses button) [H-1, H-2, H-3]	UCA-6: ECU provides Engine Stop when the driver is not shutting down the engine (i.e., pressing button) [H-1, H-2]	UCA-7: The ECU provides Engine Stop too late if the driver tries to turn off the engine [H-1, H-2]	N/A

Table 9: Initial safety constraints for ECU

Safety Constraints	Description
SC-1	ECU must not provide Engine Start if the key fob is not present.
SC-2	ECU must not provide Engine Start if the shift lever is in forward or reverse position.
SC-3	ECU must provide Engine Stop if the driver tries to turn off the engine.
SC-4	ECU must not provide Engine Stop unless the driver is turning off the engine.
SC-5	ECU must provide Engine Stop within TBD milliseconds after receiving the driver's selection of turning off the engine.

Identify basic Scenarios and causal factors

After STPA Step 1 is performed, STPA Step 2 can be conducted to identify basic scenarios in which safety constraints can be violated and causal factors contributing to each scenario. Detailed scenarios and causal factors can be constructed based on high-level ones when more design information becomes available.

- **Basic scenario 1**: ECU provides Engine Start because it incorrectly believes that the key is present.
  - **Possible causal factors**: this scenario can happen because the "key" for keyless ignition refers to the electronic key code, rather than the physical key fob. Therefore, if there is a failure in the sensor that detects the wireless signal from the physical key fob, the ECU may send signals to disarm the immobilizer in the engine starter. Then anyone (e.g. a thief) can start the engine.
- **Basic scenario 2**: ECU provides Engine Start because it incorrectly believes that vehicle is in "park" range.
  - **Possible causal factors**: This scenario can happen when there is a mistake in the transmission sensors or channel for the transmission status. If two or three bits are used to represent the transmission range (e.g., "00" park, "01" reverse, and "10" driver, etc.) without additional checks, a flip in a single bit can make the starter believe transmission is in "park" while is it in "reverse" or "drive".
- **Basic scenario 3**: ECU does not provide Engine Stop because it incorrectly believes that the driver has not turned off the engine because of inappropriate sensor sensitivity or resolution for the start/stop button.
  - **Possible causal factors**: The driver may have pressed the button, but not hard enough. If the threshold for "start/stop button pressed" is above the usual range, then the engine will not be shut down even if the driver presses the button when the vehicle comes to a stop and transmission is in "park". If this happens when the driver parks the car in the garage, he or she may leave the vehicle with the engine still running, resulting in the accumulation of toxic gases and carbon monoxide poisoning.
- **Basic scenario 4**: ECU provides Engine Stop because it incorrectly believes that the driver has pressed the start/stop button to turn off the engine.
  - **Possible causal factors**: this scenario can happen when the electronic key code is detected by the starting system. Since engine start does not requires the insertion of the physical key fob into the ignition switch, it is possible that a kid or an animal can

start the engine by touching the engine start/stop button if no protection mechanisms are added into the starting system.

- **Basic scenario 5**. ECU does not provide Engine Stop within TBD milliseconds after receiving the driver's selection because of delays in the transmission of the driver's commands.
  - **Possible causal factors**: in the event of emergencies such as malfunctions of the brake pedal, the driver may want to shut down the engine in order to reduce the speed. However, the engine torque is still applied to the wheels TBD seconds after he or she presses the start/stop button.

It is interesting to note that scenario 1 raises security concerns besides safety. Given that "key" means electronic code for the keyless ignition system, anyone who can make a copy of the key code can easily deceive the electronic control unit for the engine starter and activate the engine without the key fob.

The scenarios are used to generate additional (refined) requirements at the system level and requirements for the system components.

4.1.2 Formalize UCAs of ECU for generating SpecTRM-RL requirements

To generate SpecTRM-RL requirements, the UCAs can be formalized as described by Thomas [31]. Six process model variables (PMVs) for the ECU were identified, as shown in Table 10. These variables and their values describe conditions used by the ECU to provide control actions.

Table 10: Process model variables defined for keyless ignition

Push Button Pressed **Yes:** The starting system has just received the signal indicating engine start/stop button is pressed **No:** The starting system has not received the signal indicating engine start/stop button is pressed	**[Yes, No]**
Key Present ***Yes:*** *The electronic key code is inserted into the starting system* ***No:*** *The electronic key code is not present in the starting system*	**[Yes, No]**
Transmission Range ***P:*** *The transmission is in "park"* ***R:*** *The transmission is in "reverse"* ***N:*** *The transmission is in "neutral"* ***D:*** *The transmission is in "drive"* ***L:*** *The transmission is locked in first gear*	**[P, R, N, D, L]**
Driver Turns off Engine ***Yes:*** *The engine control module has received the driver's request to turn off the engine* ***No:*** *No request from the driver has been received to turn off the engine*	**[Yes, No]**
Vehicle Motion ***Stopped:*** *The vehicle is at a standstill* ***Moving:*** *The vehicle is not at a standstill*	**[Stopped, Moving]**
Engine Status ***On:*** *The power to the engine has been turned on and the application of the gas pedal will lead to input to the vehicle wheel* ***Off:*** *The power to the engine has been turned off and no input to the vehicle wheel will result from the application of the gas pedal*	**[On, Off]**
Brake Pedal (Pressed) ***Yes:*** *Brake pedal travel is at least X mm* ***No:*** *Brake pedal travel is less than the detectable X mm*	**[Yes, No]**

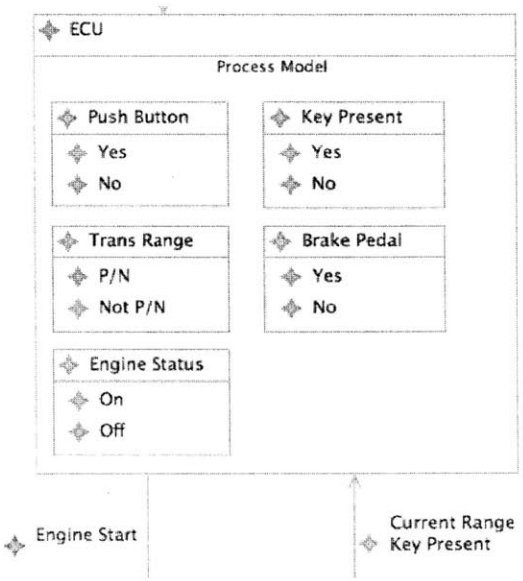


Figure 21: Adding process model variables to ECU

After adding these PMVs and their related values to the control structure in Figure 20, the STPA tool can create a low-level context table automatically for *Engine Start*, as shown in Table 11. Similarly, a low-level context table for *Engine Stop* can also be generated, as shown in

Table 12. Each row in Table 11 corresponds to a context in which the *Engine Start* command is provided by the ECU, while each row in

Table 12 corresponds to a context in which the *Engine Stop* command is provided. Engineers can use these tables to decide if providing or not providing *Engine Start* or *Engine Stop* in each context can cause hazards.

Table 11: Low-level context table created automatically for *Engine Start*

Control Action List	Control Action	Type	Push Button	Key Present	Trans Range	Brake Pedal	Engine Status	Hazards	Too Early/Too Late Hazards
Engine Start	Engine Start	not provided when	Yes	Yes	P/N	Yes	On		
	Engine Start	not provided when	Yes	Yes	P/N	Yes	Off		
	Engine Start	not provided when	Yes	Yes	P/N	No	On		
	Engine Start	not provided when	Yes	Yes	P/N	No	Off		
	Engine Start	not provided when	Yes	Yes	Not P/N	Yes	On		
	Engine Start	not provided when	Yes	Yes	Not P/N	Yes	Off		
	Engine Start	not provided when	Yes	Yes	Not P/N	No	On		
	Engine Start	not provided when	Yes	Yes	Not P/N	No	Off		
	Engine Start	not provided when	Yes	No	P/N	Yes	On		
	Engine Start	not provided when	Yes	No	P/N	Yes	Off		
	Engine Start	not provided when	Yes	No	P/N	No	On		
	Engine Start	not provided when	Yes	No	P/N	No	Off		
	Engine Start	not provided when	Yes	No	Not P/N	Yes	On		
	Engine Start	not provided when	Yes	No	Not P/N	Yes	Off		
	Engine Start	not provided when	Yes	No	Not P/N	No	On		
	Engine Start	not provided when	Yes	No	Not P/N	No	Off		
	Engine Start	not provided when	No	Yes	P/N	Yes	On		
	Engine Start	not provided when	No	Yes	P/N	Yes	Off		
	Engine Start	not provided when	No	Yes	P/N	No	On		
	Engine Start	not provided when	No	Yes	P/N	No	Off		
	Engine Start	not provided when	No	Yes	Not P/N	Yes	On		
	Engine Start	not provided when	No	Yes	Not P/N	Yes	Off		
	Engine Start	not provided when	No	Yes	Not P/N	No	On		
	Engine Start	not provided when	No	Yes	Not P/N	No	Off		
	Engine Start	not provided when	No	No	P/N	Yes	On		
	Engine Start	not provided when	No	No	P/N	Yes	Off		
	Engine Start	not provided when	No	No	P/N	No	On		
	Engine Start	not provided when	No	No	P/N	No	Off		
	Engine Start	not provided when	No	No	Not P/N	Yes	On		

Table 12: Low-level context cable created automatically for *Engine Stop*

Control Action List	Control Actior	Type	Driver Engine Off	Vehicle Motion	Trans Range	Engine Status	Hazards	Too Early/Too Late Hazards
	Engine Stop	not provid.	Yes	Moving	!=P	On		
Engine Stop	Engine Stop	not provid.	Yes	Moving	!=P	Off		
	Engine Stop	not provid.	Yes	Moving	P	On		
	Engine Stop	not provid.	Yes	Moving	P	Off		
	Engine Stop	not provid.	Yes	Stopped	!=P	On		
	Engine Stop	not provid.	Yes	Stopped	!=P	Off		
	Engine Stop	not provid.	Yes	Stopped	P	On		
	Engine Stop	not provid.	Yes	Stopped	P	Off		
	Engine Stop	not provid.	No	Moving	!=P	On		
	Engine Stop	not provid.	No	Moving	!=P	Off		
	Engine Stop	not provid.	No	Moving	P	On		
	Engine Stop	not provid.	No	Moving	P	Off		
	Engine Stop	not provid.	No	Stopped	!=P	On		
	Engine Stop	not provid.	No	Stopped	!=P	Off		
	Engine Stop	not provid.	No	Stopped	P	On		
	Engine Stop	not provid.	No	Stopped	P	Off		
	Engine Stop	provided w.	Yes	Moving	!=P	On		
	Engine Stop	provided w.	Yes	Moving	!=P	Off		
	Engine Stop	provided w.	Yes	Moving	P	On		
	Engine Stop	provided w.	Yes	Moving	P	Off		
	Engine Stop	provided w.	Yes	Stopped	!=P	On		
	Engine Stop	provided w.	Yes	Stopped	!=P	Off		
	Engine Stop	provided w.	Yes	Stopped	P	On		
	Engine Stop	provided w.	Yes	Stopped	P	Off		
	Engine Stop	provided w.	No	Moving	!=P	On		
	Engine Stop	provided w.	No	Moving	!=P	Off		
	Engine Stop	provided w.	No	Moving	P	On		
	Engine Stop	provided w.	No	Moving	P	Off		
	Engine Stop	provided w.	No	Stopped	!=P	On		

As mentioned in Section 3.3, rather than manually filling out original context tables, as shown in Table 11 and

Table 12, engineers can work on the editor provided by the STPA tool to define a small set of rules in natural language form. The STPA tool will then generate complete context tables, extract all relevant UCAs, and generate SpecTRM-RL requirements automatically.

Figure 22 includes all the rules defined for the *Engine Start* command by the ECU. As can be seen, R1 is needed for achieving safety goals, because starting the engine without the driver's input can expose humans to harmful effects such as carbon monoxide poisoning. R2 is needed to ensure that the starting system conforms to safety regulations. To prevent inadvertent activation of the engine, FMVSS 114 states that "the starting system must prevent key removal unless the transmission or gear selection control is locked in 'park' or becomes locked in 'park' as the direct result of key removal." After all rules are defined, the STPA tool can automatically fill out the right column of the *Engine Start* command, as shown in

Table 13.

Rule in English (Engine Start)
Rule in English
R1: Engine Start provided is hazardous when Push Button Pressed is No (H-3)
R2: Engine Start provided is hazardous when Push Button Pressed is Yes,Key Present is No (R-1)
R3: Engine Start provided is hazardous when Push Button Pressed is Yes,Key Present is Yes,Engine Status is On (H-3)
R4: Engine Start provided is hazardous when Push Button Pressed is Yes,Key Present is Yes,Brake Pedal is No,Engine Status is Off (H-1,H-2)
R5: Engine Start provided is hazardous when Push Button Pressed is Yes,Key Present is Yes,Trans Range is Not P/N,Brake Pedal is Yes,Engine Status is Off (R-1)
R6: Engine Start not provided is hazardous when Push Button Pressed is Yes,Key Present is Yes,Trans Range is P/N,Brake Pedal is Yes,Engine Status is Off (H-1,H-2)

Figure 22: Rule definitions for *Engine Start* command

Table 13: Context table for *Engine Start* automatically filled out based on "Rules" (Figure 22)

Control Action	Type	Push Button	Key Present	Trans Range	Brake Pedal	Engine Status	Hazards	Too Early/Too Late Hazards	Related Rules
Engine Start	not provided when	Yes	Yes	P/N	Yes	On			
Engine Start	not provided when	Yes	Yes	P/N	Yes	Off	H-1: Vehic		R6
Engine Start	not provided when	Yes	Yes	P/N	No	On			
Engine Start	not provided when	Yes	Yes	P/N	No	Off			
Engine Start	not provided when	Yes	Yes	Not P/N	Yes	On			
Engine Start	not provided when	Yes	Yes	Not P/N	Yes	Off			
Engine Start	not provided when	Yes	Yes	Not P/N	No	On			
Engine Start	not provided when	Yes	Yes	Not P/N	No	Off			
Engine Start	not provided when	Yes	No	P/N	Yes	On			
Engine Start	not provided when	Yes	No	P/N	Yes	Off			
Engine Start	not provided when	Yes	No	P/N	No	On			
Engine Start	not provided when	Yes	No	P/N	No	Off			
Engine Start	not provided when	Yes	No	Not P/N	Yes	On			
Engine Start	not provided when	Yes	No	Not P/N	Yes	Off			
Engine Start	not provided when	Yes	No	Not P/N	No	On			
Engine Start	not provided when	Yes	No	Not P/N	No	Off			
Engine Start	not provided when	No	Yes	P/N	Yes	On			
Engine Start	not provided when	No	Yes	P/N	Yes	Off			
Engine Start	not provided when	No	Yes	P/N	No	On			
Engine Start	not provided when	No	Yes	P/N	No	Off			
Engine Start	not provided when	No	Yes	Not P/N	Yes	On			
Engine Start	not provided when	No	Yes	Not P/N	Yes	Off			
Engine Start	not provided when	No	Yes	Not P/N	No	On			
Engine Start	not provided when	No	Yes	Not P/N	No	Off			
Engine Start	not provided when	No	No	P/N	Yes	On			
Engine Start	not provided when	No	No	P/N	Yes	Off			
Engine Start	not provided when	No	No	P/N	No	On			
Engine Start	not provided when	No	No	P/N	No	Off			
Engine Start	not provided when	No	No	Not P/N	Yes	On			
Engine Start	not provided when	No	No	Not P/N	Yes	Off			
Engine Start	not provided when	No	No	Not P/N	No	On			
Engine Start	not provided when	No	No	Not P/N	No	Off			
Engine Start	provided when	Yes	Yes	P/N	Yes	On	H-3: Vehic	H-3: Vehicle oc	R3
Engine Start	provided when	Yes	Yes	P/N	Yes	Off			
Engine Start	provided when	Yes	Yes	P/N	No	On	H-3: Vehic	H-3: Vehicle oc	R3
Engine Start	provided when	Yes	Yes	P/N	No	Off	H-1: Vehic	H-1: Vehicle do	R4
Engine Start	provided when	Yes	Yes	Not P/N	Yes	On	H-3: Vehic	H-3: Vehicle oc	R3
Engine Start	provided when	Yes	Yes	Not P/N	Yes	Off	R-1: Vehic	R-1: Vehicle do	R5
Engine Start	provided when	Yes	Yes	Not P/N	No	On	H-3: Vehic	H-3: Vehicle oc	R3
Engine Start	provided when	Yes	Yes	Not P/N	No	Off	H-1: Vehic	H-1: Vehicle do	R4
Engine Start	provided when	Yes	No	P/N	Yes	On	R-1: Vehic	R-1: Vehicle do	R2
Engine Start	provided when	Yes	No	P/N	Yes	Off	R-1: Vehic	R-1: Vehicle do	R2
Engine Start	provided when	Yes	No	P/N	No	On	R-1: Vehic	R-1: Vehicle do	R2
Engine Start	provided when	Yes	No	P/N	No	Off	R-1: Vehic	R-1: Vehicle do	R2
Engine Start	provided when	Yes	No	Not P/N	Yes	On	R-1: Vehic	R-1: Vehicle do	R2
Engine Start	provided when	Yes	No	Not P/N	Yes	Off	R-1: Vehic	R-1: Vehicle do	R2
Engine Start	provided when	Yes	No	Not P/N	No	On	R-1: Vehic	R-1: Vehicle do	R2
Engine Start	provided when	Yes	No	Not P/N	No	Off	R-1: Vehic	R-1: Vehicle do	R2
Engine Start	provided when	No	Yes	P/N	Yes	On	H-3: Vehic	H-3: Vehicle oc	R1
Engine Start	provided when	No	Yes	P/N	Yes	Off	H-3: Vehic	H-3: Vehicle oc	R1
Engine Start	provided when	No	Yes	P/N	No	On	H-3: Vehic	H-3: Vehicle oc	R1
Engine Start	provided when	No	Yes	P/N	No	Off	H-3: Vehic	H-3: Vehicle oc	R1
Engine Start	provided when	No	Yes	Not P/N	Yes	On	H-3: Vehic	H-3: Vehicle oc	R1
Engine Start	provided when	No	Yes	Not P/N	Yes	Off	H-3: Vehic	H-3: Vehicle oc	R1
Engine Start	provided when	No	Yes	Not P/N	No	On	H-3: Vehic	H-3: Vehicle oc	R1
Engine Start	provided when	No	Yes	Not P/N	No	Off	H-3: Vehic	H-3: Vehicle oc	R1
Engine Start	provided when	No	No	P/N	Yes	On	H-3: Vehic	H-3: Vehicle oc	R1
Engine Start	provided when	No	No	P/N	Yes	Off	H-3: Vehic	H-3: Vehicle oc	R1
Engine Start	provided when	No	No	P/N	No	On	H-3: Vehic	H-3: Vehicle oc	R1
Engine Start	provided when	No	No	P/N	No	Off	H-3: Vehic	H-3: Vehicle oc	R1
Engine Start	provided when	No	No	Not P/N	Yes	On	H-3: Vehic	H-3: Vehicle oc	R1
Engine Start	provided when	No	No	Not P/N	Yes	Off	H-3: Vehic	H-3: Vehicle oc	R1
Engine Start	provided when	No	No	Not P/N	No	On	H-3: Vehic	H-3: Vehicle oc	R1
Engine Start	provided when	No	No	Not P/N	No	Off	H-3: Vehic	H-3: Vehicle oc	R1

The tool also generates simplified context tables and SpecTRM-RL requirements for the *Engine Start* command, as can be seen in Table 14 and Figure **23**. Note that although the full context table in Table 13 contains a lot of information with 64 rows, the tool was able to apply logical simplification and automatically generate the simplified requirement in Figure **23**. The simplified requirement is easily understandable and addresses all of the UCAs identified in Table 13. For example, the UCA ECU-Start-3 refers to an unsafe command to start the engine when the engine is already started. As can be seen in Figure **23**, the requirement will not allow the engine start command when the engine is on.

Table 14: Simplified context table of *Engine Start* command generated by the STPA tool

Controller: Electronic Control Unit								
Control Action	Context ID	Driver Selection	Key Present	Trans Range	Brake Pedal	Engine Status	Not Providing causes Hazards	Providing causes Hazards
Engine Start	ECU-Start-1	No	*	*	*	*		Yes
	ECU-Start-2	Yes	No	*	*	*		Yes
	ECU-Start-3	Yes	Yes	*	*	On		Yes
	ECU-Start-4	Yes	Yes	*	No	Off		Yes
	ECU-Start-5	Yes	Yes	Not P/N	Yes	Off		Yes
	ECU-Start-6	Yes	Yes	P/N	Yes	Off	Yes	

Control algorithm for 'Engine Start' cmd		
Push Button=	Yes	T
	No	
Key Present=	Yes	T
	No	
Trans Range=	P/N	T
	Not P/N	
Brake Pedal=	Yes	T
	No	
Engine Status=	On	
	Off	T

Figure 23: SpecTRM-RL for *Engine Start* command generated by the STPA tool

Similarly, engineers can define rules for the *Engine Stop* command, as shown in Figure 24. Table 15 gives the context table for the *Engine Stop* command that is automatically filled out by the STPA tool. The simplified context table and corresponding SpecTRM-RL requirements are also generated automatically, as shown in Table 16 and Figure 25.

Rule in English (Engine Stop)
Rule in English
R1: Engine Stop provided is hazardous when Driver Engine Off is No (H-1,H-2)
R2: Engine Stop provided is hazardous when Driver Engine Off is Yes,Vehicle Motion is Stopped,Trans Range is !=P,Engine Status is On (R-1)
R3: Engine Stop not provided is hazardous when Driver Engine Off is Yes,Vehicle Motion is Stopped,Trans Range is P,Engine Status is On (H-3)

Figure 24: Rule definitions for *Engine Stop* command

Table 15: Context table for *Engine Stop* command automatically filled out based on "Rules" (Figure 24)

Control Action	Type	Driver Engine Off	Wheels Rotating	Trans Range	Engine Status	Hazards	Too Early/Too Late Hazards	Related Rules
Engine Stop	not provided when	Yes	Yes	!=P	On			
Engine Stop	not provided when	Yes	Yes	!=P	Off			
Engine Stop	not provided when	Yes	Yes	P	On			
Engine Stop	not provided when	Yes	Yes	P	Off			
Engine Stop	not provided when	Yes	No	!=P	On			
Engine Stop	not provided when	Yes	No	!=P	Off			
Engine Stop	not provided when	Yes	No	P	On	H-3: Vehi		R3
Engine Stop	not provided when	Yes	No	P	Off			
Engine Stop	not provided when	No	Yes	!=P	On			
Engine Stop	not provided when	No	Yes	!=P	Off			
Engine Stop	not provided when	No	Yes	P	On			
Engine Stop	not provided when	No	Yes	P	Off			
Engine Stop	not provided when	No	No	!=P	On			
Engine Stop	not provided when	No	No	!=P	Off			
Engine Stop	not provided when	No	No	P	On			
Engine Stop	not provided when	No	No	P	Off			
Engine Stop	provided when	Yes	Yes	!=P	On			
Engine Stop	provided when	Yes	Yes	!=P	Off			
Engine Stop	provided when	Yes	Yes	P	On			
Engine Stop	provided when	Yes	Yes	P	Off			
Engine Stop	provided when	Yes	No	!=P	On	R-1: Vehi	R-1: Vehicle doe	R2
Engine Stop	provided when	Yes	No	!=P	Off			
Engine Stop	provided when	Yes	No	P	On			
Engine Stop	provided when	Yes	No	P	Off			
Engine Stop	provided when	No	Yes	!=P	On	H-1: Vehi	H-1: Vehicle doe	R1
Engine Stop	provided when	No	Yes	!=P	Off	H-1: Vehi	H-1: Vehicle doe	R1
Engine Stop	provided when	No	Yes	P	On	H-1: Vehi	H-1: Vehicle doe	R1
Engine Stop	provided when	No	Yes	P	Off	H-1: Vehi	H-1: Vehicle doe	R1
Engine Stop	provided when	No	No	!=P	On	H-1: Vehi	H-1: Vehicle doe	R1
Engine Stop	provided when	No	No	!=P	Off	H-1: Vehi	H-1: Vehicle doe	R1
Engine Stop	provided when	No	No	P	On	H-1: Vehi	H-1: Vehicle doe	R1
Engine Stop	provided when	No	No	P	Off	H-1: Vehi	H-1: Vehicle doe	R1

Table 16: Simplified context table of *Engine Stop* generated by the STPA tool

Controller: Electronic Control Unit							
Control Action	Context ID	Driver Turns Off Engine	Vehicle Motion	Trans Range	Engine Status	Not Providing causes Hazards	Providing causes Hazards
Engine Stop	ECU-Stop-1	No	*	*	*		Yes
	ECU-Stop-2	Yes	Moving	*	*		
	ECU-Stop-3	Yes	Stopped	*	Off		
	ECU-Stop-4	Yes	Stopped	Not P	On		Yes
	ECU-Stop-5	Yes	Stopped	P	On	Yes	

Control algorithm for 'Engine Stop' cmd		
Driver Engine Off=	Yes	T
	No	
Vehicle Motion=	Moving	
	Stopped	T
Trans Range=	!=P	
	P	T
Engine Status=	On	T
	Off	

Figure 25: SpecTRM-RL for *Engine Stop* generated by the STPA tool

4.2 Tool-Assisted Analysis of EBA

4.2.1 Initial STPA of EBA

System Overview

Emergency braking assist (EBA) is a feature that provides braking support to the driver to avoid front-end collisions of the current vehicle in emergencies. With advanced sensors for object recognition such as radar sensors, EBA is able to monitor stationary/moving objects ahead and decide the range (i.e., distance) and range rate (i.e., relative speed) of the current vehicle. The EBA detects a collision risk if the current vehicle is moving with a high range rate and does not keep a safe distance from the vehicle in front. After potential risks of collision are detected, the system alerts the driver by visual or audio signals and pre-charges the braking system so that the full-powered brake is available during an emergency brake. Whenever the driver presses the brake pedal after a collision risk is detected, the EBA will apply the full-powered brake to assist the driver. Then, if the driver does not react and the gap distance between the two vehicles continues to decrease to the point where a front-end collision is unavoidable unless the full-powered brake is applied, the EBA will initiate the full brake automatically to reduce the braking distance. To summarize, the EBA is able to initiate full-powered brake under two situations:

- Driver presses the brake pedal and there is a collision risk OR
- Collision is imminent (even without input from the driver)

Since EBA is designed to prevent front-end collision and detailed information about designs is not available at the high-level analysis, it is reasonable to make the assumption that the EBA will not release the brake until the vehicle comes to a stop after initiating the brake automatically. In addition, the EBA must not compromise the driver's ability to control the vehicle if there is no collision risk. Specifically, the EBA must not take control authority over the brake if there is no emergency. Even if the system detects a collision risk and alerts the driver by warning signals, it should give the driver enough time to respond (before providing full brake automatically) as long as the collision can be prevented.

Control Structure and Control Actions

Figure 26 gives the high-level safety control structure of EBA. As mentioned before, the driver may press the brake pedal to reduce the speed after receiving warning signals indicating a collision risk. The automated controller of EBA detects collision risks based on feedback information from the radar (i.e., range and range rate). If a collision risk is detected, EBA will alert the driver and *Pre-Charge* the brake. If the driver does not respond when the collision is imminent, the EBA will issue the *Full Brake* command automatically to reduce the braking distance during emergencies. In addition, it must be aware of the motion of the vehicle and detect if other features (e.g. transmission is in "park" or electronic parking brake is applied) are holding the vehicle before releasing the brake. Table 17 summarizes three control actions provided by the EBA.

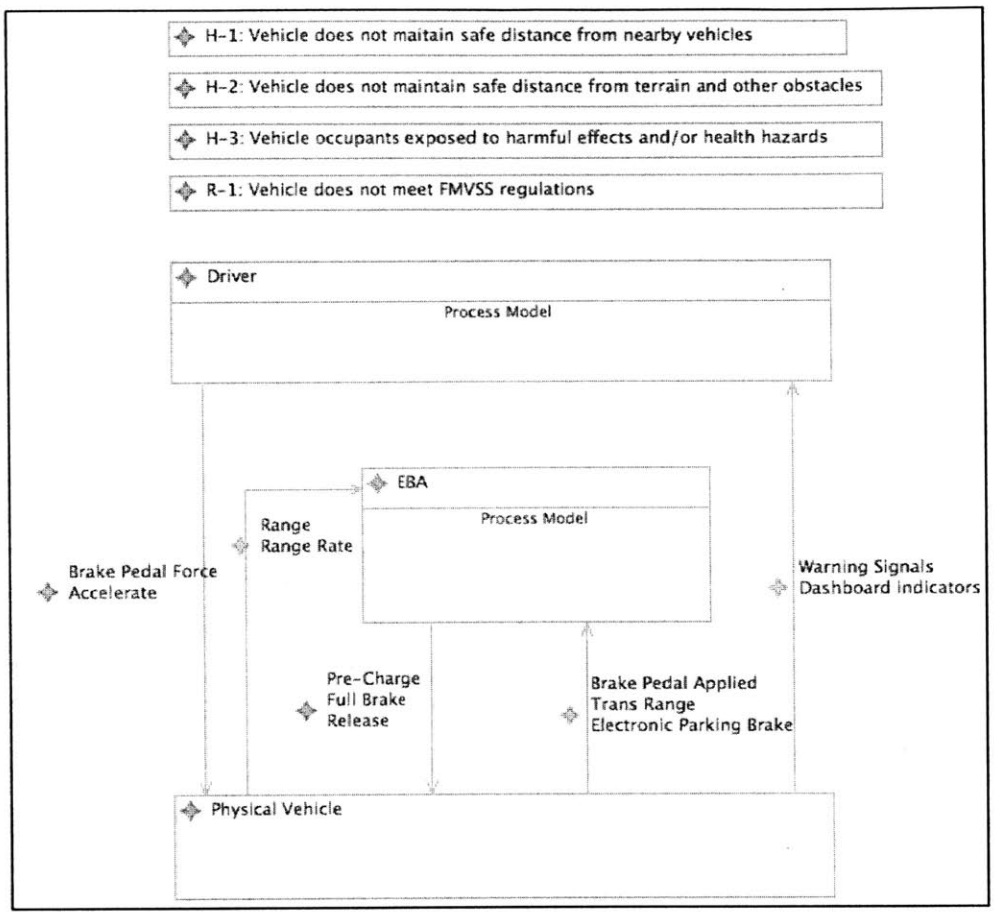


Figure 26: The high-level safety control structure of the emergency braking assist

Table 17: Control actions provided by the emergency braking assist

Control Action	Description
Pre-Charge	Prepares the braking system for emergency braking with full power. When the brakes are pre-charged, pressing the brake pedal by any amount will result in full power braking force.
Full Brake	Performs emergency braking will full power braking.
Release	Disengages the braking system from full power brake and gives the control of the brake to other controllers.

Initial Unsafe Control Actions and Safety Constraints

As with keyless ignition, engineers can perform STPA Step 1 to identify UCAs and create safety constraints for EBA. Table 18 shows how providing (or not providing) *Pre-charge*, *Full Brake*, or *Release* can cause hazards. Based on the identified UCA, the safety constraints of EBA can be created, as shown in Table 19.

Table 18: Initial UCAs of EBA

Control Action	Not providing the control action causes hazards	Providing the control action causes hazards	The timing or sequencing of control actions leads to hazards	The duration of a continuous control action leads to hazards
Pre-charge	UCA-1: Emergency brake assist does not provide pre-charge to prepare the brake when a potential collision risk is detected [H-1,H-2]	UCA-2: Emergency brake assist provides pre-charge to prepare the brake without collision risks [H-3]	UCA-3: Emergency brake assist provides pre-charge to prepare the brake for emergency brake too late [H-1,H-2]	N/A
Full Brake	UCA-4: Emergency brake assist does not provide full brake when the brake pedal is depressed and there is a collision risk [H-1,H-2]	UCA-5: Emergency brake assist initiates full power brake without collision risks and brake pedal is depressed [H-1, H-2]	UCA-6: Emergency brake assist initiates full power brake too late after the brake pedal is depressed and there is a collision risk [H-1, H-2]	N/A
	UCA-7: The emergency brake assist does not initiate full brake power automatically when a collision is imminent [H-1, H-2]	UCA-8: Emergency brake assist provides full power brake without collision risks [H-1, H-2]	UCA-9: Emergency brake assist provides full power brake too late to prevent the vehicle from front-end collision [H-1, H-2]	
Release	UCA-10: Emergency brake assist does not release the brake when the vehicle comes to a stop after an emergency brake and is held [H-1, H-2]	UCA-11: Emergency brake assist releases the brake during the emergency brake when a full powered brake is necessary [H-1, H-2]	UCA-12: Emergency brake assist releases the brake too early during emergency brake [H-1, H-2]	N/A
		UCA-13: Emergency brake assist releases the brake when the vehicle is at a standstill but is not held [H-1, H-2]		

Table 19: Initial safety constraints for EBA

Safety Constraints	Description
SC-1	EBA must pre-charge the brake when a collision risk is detected
SC-2	EBA must not pre-charge the brake if there is no collision risk
SC-3	EBA must pre-charge the brake within x milliseconds after a collision risk is detected
SC-4	EBA must provide full power brake when the brake pedal is depressed and there is a collision risk
SC-5	EBA must not provide full power brake without collision risk and the brake pedal is depressed
SC-6	EBA must provide full power brake within x milliseconds after brake pedal is depressed and there is a collision risk
SC-7	EBA must initiate the full power brake when the collision is imminent
SC-8	EBA must not initiate the full power brake without collision risk
SC-9	EBA must provide full power brake within x milliseconds if a collision is imminent
SC-10	EBA must release the brake when the vehicle comes to a stop and is held by other features or the driver
SC-11	EBA must not release the brake during the emergency brake when the full powered brake is necessary
SC-12	EBA must not release the brake during an emergency brake
SC-13	EBA must not release the brake unless the vehicle is held by other features or the driver

Identify basic scenarios and causal factors

As with keyless ignition, high-level basic scenarios can be derived from safety constraints in Table 19. Causal factors that contribute to each scenario are also given. Detailed scenarios can be constructed when more design information becomes available.

- **Basic scenario 1**: EBA does not pre-charge the brake because it incorrectly believes that there is no collision risk.
  - **Possible causal factors**: If the radar sensor malfunctions, it may provide inconsistent feedback of range and range rate. If this happens, the vehicle may not detect a collision risk even when the safe distance is violated. Then a full powered brake will not be available under the emergency brake because the braking system has not been pre-charged.
- **Basic scenario 2**: EBA pre-charges the brake because it incorrectly believes that there is a collision risk.

- **Possible causal factors**: If there is a flaw in the algorithm within the electronic controller, the EBA may decide to alert the driver and pre-charge the brake when there is no collision risk. Then, when the driver wants a "normal" brake and presses the brake pedal only slightly, he or she may be surprised by a full powered brake.

- **Basic scenario 3**: EBA pre-charges the brake too late because of delays in the braking mechanism.
  - **Possible causal factors**: The EBA does not pre-charge the brake when a collision risk is identified. However, after the collision risk disappears (this is possible if the driver of the vehicle in front applies the gas pedal to accelerate), the EBA pre-charges the brake because of delays in the transmission of the *Pre-charge* command. Then, when the driver applies the brake pedal TBD seconds later for a normal brake, he or she unexpectedly gets a full powered brake, which increases the chance of a rear-end collision.
- **Basic scenario 4**: Similar to pre-charging the brake, EBA does not provide a full power brake because it incorrectly believes that there is no collision risk.
  - **Possible causal factors**: Incorrect feedback about the position of the brake pedal can also lead to this scenario. The feedback from the position sensor for the brake pedal becomes erroneous or not as sensitive as required due to inappropriate installment. If the driver does not press the brake pedal hard enough under emergencies, the sensor may not detect that the brake pedal is applied.
- **Basic scenario 5**: EBA initiates the full powered brake because it incorrectly believes that the brake pedal is depressed and there is a collision risk.
  - **Possible causal factors**: If there is a delay in feedback channels for range and range rate, the EBA may detect a collision risk when the vehicle in front has already moved out of the "danger zone". Then, when a driver slightly presses the brake pedal for a normal brake, he or she will inexpertly get a full powered brake, which increases the chance of a rear-end collision.
- **Basic scenario 6**: EBA does not release the brake because it incorrectly believes that the emergency brake has not been finished or the vehicle is not held.
  - **Possible causal factors**: If the radar sensor malfunctions, or the algorithms for the EBA are flawed, the EBA may still believe there is a collision risk even after the vehicle comes to a stop. If this happens, the EBA will continue to apply the brake and hold the vehicle even though there is engine torque to resume the motion.

- **Basic scenario 7**: EBA releases the brake because it incorrectly believes that the vehicle has come to a stop and there is no collision risk.
  - **Possible causal factors**: If the speed sensor provides incorrect feedback about wheel speed to the EBA, it may disengage the brake when the vehicle is still moving. If this happens, the current vehicle may crash into the vehicle in front.
- **Basic scenario 8**: EBA releases the brake because it incorrectly believes that the vehicle has been held.
  - **Possible causal factors**: If the range sensor sends incorrect feedback information the current transmission range, the EBA may believe that vehicle is in the "park" range when it is in "reverse", "drive" or "neutral" range. If this happens, the EBA will release the brake because "park" range indicates that the vehicle is held.

These scenarios can help engineers come up with design solutions to prevent hazards when detailed design information is still not available. Take causal scenario 6 as an example. According to the safety constraint related to this scenario (i.e., SC 7&9), EBA may initiate the full powered brake when necessary to prevent a front-end collision if it believes that a collision is imminent. However, if feedback signals are incorrect about range or range rate, initiating the brake can violate SC 8 because the EBA must not apply the full powered brake when there is no collision risk. Therefore, the engineer may consider adding other sensors for object recognition. For example, a camera may also be used for detecting a stationary or moving object in front. Also, design flaws in the algorithm of the EBA can lead to this scenario—the EBA does not give the driver enough time to respond to the warning before the full brake. This may not necessarily lead to hazards, but the driver will get confused about the automated behaviors. As mentioned in section 4.2.1, although EBA should have the highest priority and control authority over the brake during emergencies, it should not compromise the driver's ability to control the vehicle unless rear-end collisions are about to occur. A potential solution may be adding other timing constraints for the EBA before the full powered brake is applied automatically.

4.2.2 Formalize UCAs of EBA for generating SpecTRM-RL requirements

This section starts with the definition of PMVs related to conditions in which the EBA provides the *Full Brake* and *Release* commands, as shown in Table 20. The STPA tool creates low-level context tables for *Full Brake* and *Release* automatically based on the PMVs defined.

Table 20: Process model variables defined for EBA

Collision Risk **Yes:** The (distance to speed) rate is less than the threshold and deceleration is required to prevent collision **No:** The vehicle is not at risk of rear-end collision based on the current speed and the distance from the vehicle in front	**[Yes, No]**
Collision Imminent ***Yes:*** *A rear-end collision is unavoidable unless full powered brake is applied* ***No:*** *The vehicle is at risk of a potential collision, but it can be avoided as long as the driver applies the brake*	**[Yes, No]**
Acceleration ***Yes:*** *The vehicle is moving forward and going through an acceleration due to the driver's input (i.e., gas pedal) or the engagement of other features (e.g. adaptive cruise control)* ***No:*** *The vehicle is at a standstill, reversing, moving at a constant speed or decelerating*	**[Yes, No]**
Brake Pedal ***Pressed:*** *Brake pedal travel is at least X mm* ***Not Pressed:*** *Brake pedal travel is less than the detectable X mm*	**[Pressed, Not Pressed]**
Vehicle Motion ***Stopped:*** *The vehicle is at a standstill* ***Moving:*** *The vehicle is not at a standstill*	**[Stopped, Moving]**
Full Brake Applied ***Yes:*** *The full powered brake is applied by the EBA* ***No:*** *The full powered brake has not been applied by the EBA*	**[Yes, No]**
Vehicle Held ***Yes:*** *The vehicle is held by other features or the driver (by pressing the brake pedal)* ***No:*** *The vehicle is not held by other features or the driver*	**[Yes, No]**

The context table for *Pre-charge* is not given since it only involves one high-level variable (i.e., collision risk).

Table 21 and

Table 22 show the low-level context tables for *Full Brake* and *Release* provided by EBA.

Table 21: Low-level context table created automatically for *Full Brake*

Control Action List	Control Action	Type	Collision Risk	Collision Imminent	Gas Pedal	Full Brake Applied	Brake Pedal	Hazards	Too Early/Too Late Hazards
	Full Brake	not provided when	Yes	Yes	Yes	Yes	Yes		
Full Brake	Full Brake	not provided when	Yes	Yes	Yes	Yes	No		
	Full Brake	not provided when	Yes	Yes	Yes	No	Yes		
	Full Brake	not provided when	Yes	Yes	Yes	No	No		
	Full Brake	not provided when	Yes	Yes	No	Yes	Yes		
	Full Brake	not provided when	Yes	Yes	No	Yes	No		
	Full Brake	not provided when	Yes	Yes	No	No	Yes		
	Full Brake	not provided when	Yes	Yes	No	No	No		
	Full Brake	not provided when	Yes	No	Yes	Yes	Yes		
	Full Brake	not provided when	Yes	No	Yes	Yes	No		
	Full Brake	not provided when	Yes	No	Yes	No	Yes		
	Full Brake	not provided when	Yes	No	No	Yes	No		
	Full Brake	not provided when	Yes	No	No	Yes	Yes		
	Full Brake	not provided when	Yes	No	No	No	No		
	Full Brake	not provided when	No	Yes	Yes	Yes	Yes		
	Full Brake	not provided when	No	Yes	Yes	Yes	No		
	Full Brake	not provided when	No	Yes	Yes	No	Yes		
	Full Brake	not provided when	No	Yes	Yes	No	No		
	Full Brake	not provided when	No	Yes	No	Yes	Yes		
	Full Brake	not provided when	No	Yes	No	Yes	No		
	Full Brake	not provided when	No	Yes	No	No	Yes		
	Full Brake	not provided when	No	Yes	No	No	No		
	Full Brake	not provided when	No	No	Yes	Yes	Yes		
	Full Brake	not provided when	No	No	Yes	Yes	No		
	Full Brake	not provided when	No	No	Yes	No	Yes		
	Full Brake	not provided when	No	No	Yes	No	No		
	Full Brake	not provided when	No	No	No	Yes	Yes		

Table 22: Low-level context table created automatically for *Release*

Control Action List	Control Action	Type	Vehicle Motion	Driver Present	Full Brake Applied	Vehicle Held	Gas Pedal	Hazards	Too Early/Too Late Hazard
	Release	not provided when	Moving	Yes	Yes	Yes	Yes		
Release	Release	not provided when	Moving	Yes	Yes	Yes	No		
	Release	not provided when	Moving	Yes	Yes	No	Yes		
	Release	not provided when	Moving	Yes	Yes	No	No		
	Release	not provided when	Moving	Yes	No	Yes	Yes		
	Release	not provided when	Moving	Yes	No	Yes	No		
	Release	not provided when	Moving	Yes	No	No	Yes		
	Release	not provided when	Moving	Yes	No	No	No		
	Release	not provided when	Moving	No	Yes	Yes	Yes		
	Release	not provided when	Moving	No	Yes	Yes	No		
	Release	not provided when	Moving	No	Yes	No	Yes		
	Release	not provided when	Moving	No	Yes	No	No		
	Release	not provided when	Moving	No	No	Yes	Yes		
	Release	not provided when	Moving	No	No	No	No		
	Release	not provided when	Moving	No	No	No	Yes		
	Release	not provided when	Moving	No	No	No	No		
	Release	not provided when	Stopped	Yes	Yes	Yes	Yes		
	Release	not provided when	Stopped	Yes	Yes	Yes	No		
	Release	not provided when	Stopped	Yes	Yes	No	Yes		
	Release	not provided when	Stopped	Yes	Yes	No	No		
	Release	not provided when	Stopped	Yes	No	Yes	Yes		
	Release	not provided when	Stopped	Yes	No	Yes	No		
	Release	not provided when	Stopped	Yes	No	No	Yes		
	Release	not provided when	Stopped	Yes	No	No	No		
	Release	not provided when	Stopped	No	Yes	Yes	Yes		
	Release	not provided when	Stopped	No	Yes	Yes	No		
	Release	not provided when	Stopped	No	Yes	No	Yes		
	Release	not provided when	Stopped	No	Yes	No	No		
	Release	not provided when	Stopped	No	No	Yes	Yes		

Similar to the analysis of keyless ignition, rather than working on low-level context tables, the STPA tool can help engineers define rules to simplify the process of identifying UCAs and create formal requirements (SpecTRM-RL) automatically.

Figure 27 includes rules defined for the *Full Brake* command provided by EBA. The rule editor of the STPA tool provides more guidance on identifying explicit assumptions and resolving potential conflicts. For example, the EBA must provide the emergency brake while conditions are met according to the safety constraints from the initial Step 1. However, engineers may also consider the possibility that the *Full Brake* command has already been applied either by the driver or other features that are capable of engaging the braking mechanism. In this case, they must also think about whether providing the redundant *Full Brake* command may lead to resetting or canceling the emergency procedure. As another example, issuing the *Full Brake* command can make the vehicle difficult to control if other controllers are applying the gas pedal at the same time.

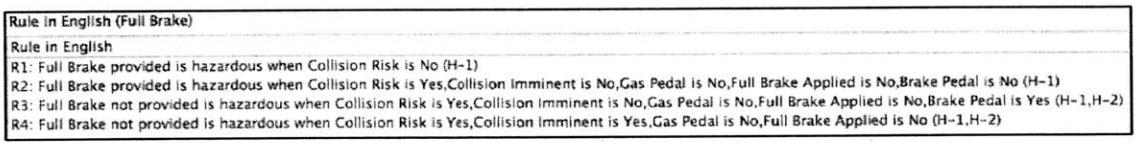


Figure 27: Rule definitions for *Full Brake*

After all rules are defined, the STPA tool automatically fills out the right column of the original context table for the *Full Brake* command, as shown in Table 23. The tool also generates a simplified context table including SpecTRM-RL requirements for *Full Brake*, as shown in Table 24 and Figure 28.

Table 23: Context table for *Full Brake* command automatically filled out based on "Rules" (Figure 27)

Control Action	Type	Collision Risk	Collision imminent	Full Brake Applied	Gas Pedal	Brake Pedal	Hazards	Too Early/Too Late Hazards	Related Rules
Full Brake	not provided when	Yes	Yes	Yes	Yes	Yes			
Full Brake	not provided when	Yes	Yes	Yes	Yes	No			
Full Brake	not provided when	Yes	Yes	Yes	No	Yes			
Full Brake	not provided when	Yes	Yes	Yes	No	No			
Full Brake	not provided when	Yes	Yes	No	Yes	Yes			
Full Brake	not provided when	Yes	Yes	No	Yes	No			
Full Brake	not provided when	Yes	Yes	No	No	Yes	H-2: Vehi		R4
Full Brake	not provided when	Yes	Yes	No	No	No	H-2: Vehi		R4
Full Brake	not provided when	Yes	No	Yes	Yes	Yes			
Full Brake	not provided when	Yes	No	Yes	Yes	No			
Full Brake	not provided when	Yes	No	Yes	No	Yes			
Full Brake	not provided when	Yes	No	Yes	No	No			
Full Brake	not provided when	Yes	No	No	Yes	Yes			
Full Brake	not provided when	Yes	No	No	Yes	No			
Full Brake	not provided when	Yes	No	No	No	Yes	H-1: Vehi		R3
Full Brake	not provided when	Yes	No	No	No	No			
Full Brake	not provided when	No	Yes	Yes	Yes	Yes			
Full Brake	not provided when	No	Yes	Yes	Yes	No			
Full Brake	not provided when	No	Yes	Yes	No	Yes			
Full Brake	not provided when	No	Yes	Yes	No	No			
Full Brake	not provided when	No	Yes	No	Yes	Yes			
Full Brake	not provided when	No	Yes	No	Yes	No			
Full Brake	not provided when	No	Yes	No	No	Yes			
Full Brake	not provided when	No	Yes	No	No	No			
Full Brake	not provided when	No	No	Yes	Yes	Yes			
Full Brake	not provided when	No	No	Yes	Yes	No			
Full Brake	not provided when	No	No	Yes	No	Yes			
Full Brake	not provided when	No	No	Yes	No	No			
Full Brake	not provided when	No	No	No	Yes	Yes			
Full Brake	not provided when	No	No	No	Yes	No			
Full Brake	not provided when	No	No	No	No	Yes			
Full Brake	not provided when	No	No	No	No	No			
Full Brake	provided when	Yes	Yes	Yes	Yes	Yes			
Full Brake	provided when	Yes	Yes	Yes	Yes	No			
Full Brake	provided when	Yes	Yes	Yes	No	Yes			
Full Brake	provided when	Yes	Yes	Yes	No	No			
Full Brake	provided when	Yes	Yes	No	Yes	Yes			
Full Brake	provided when	Yes	Yes	No	Yes	No			
Full Brake	provided when	Yes	Yes	No	No	Yes			
Full Brake	provided when	Yes	Yes	No	No	No			
Full Brake	provided when	Yes	No	Yes	Yes	Yes			
Full Brake	provided when	Yes	No	Yes	Yes	No			
Full Brake	provided when	Yes	No	Yes	No	Yes			
Full Brake	provided when	Yes	No	Yes	No	No			
Full Brake	provided when	Yes	No	No	Yes	Yes			
Full Brake	provided when	Yes	No	No	Yes	No			
Full Brake	provided when	Yes	No	No	No	Yes			
Full Brake	provided when	Yes	No	No	No	No	H-2: Vehi	H-2: Vehicle do	R2
Full Brake	provided when	No	Yes	Yes	Yes	Yes	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	Yes	Yes	Yes	No	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	Yes	Yes	No	Yes	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	Yes	Yes	No	No	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	Yes	No	Yes	Yes	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	Yes	No	Yes	No	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	Yes	No	No	Yes	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	Yes	No	No	No	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	No	Yes	Yes	Yes	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	No	Yes	Yes	No	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	No	Yes	No	Yes	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	No	Yes	No	No	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	No	No	Yes	Yes	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	No	No	Yes	No	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	No	No	No	Yes	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	No	No	No	No	H-2: Vehi	H-2: Vehicle do	R1

Table 24: Simplified context table for *Full Brake*

Controller: EBA								
Control Action	Context ID	Collision Risk	Collision Imminent	Gas Pedal Applied	Full Brake Applied	Brake Pedal	Not Providing causes Hazards	Providing causes Hazards
Full Brake	EBA-FB-1	No	*	*	*	*		Yes
	EBA-FB-2	Yes	*	Yes	*	*		
	EBA-FB-3	Yes	*	No	Yes	*		
	EBA-FB-4	Yes	No	No	No	Pressed	Yes	
	EBA-FB-5	Yes	No	No	No	Not Pressed		Yes
	EBA-FB-6	Yes	Yes	No	No	*	Yes	

Control algorithm for 'Full Brake' cmd			
Collision Risk=	Yes	T	T
	No		
Collision Immine...	Yes		T
	No	T	
Gas Pedal=	Yes		
	No	T	T
Full Brake Applied=	Yes		
	No	T	T
Brake Pedal=	Yes	T	
	No		

Figure 28: SpecTRM-RL for *Full Brake* generated automatically

Similarly, rules can be defined for the *Release* command, based on which the original context table for the *Release* command can be automatically filled out, as shown in Figure 29 and Table 25. The simplified context table and SpecTRM-RL requirements generated by the STPA tool are shown in Table 26 and Figure 30.

Rule in English (Release)
Rule in English
R1: Release provided is hazardous when Vehicle Motion is Moving,Full Brake Applied is Yes (H-1,H-2)
R2: Release provided is hazardous when Vehicle Motion is Stopped,Driver Present is No,Full Brake Applied is Yes,Vehicle Held is No (H-1,H-2)
R3: Release provided is hazardous when Vehicle Motion is Stopped,Driver Present is Yes,Full Brake Applied is Yes,Vehicle Held is No,Gas Pedal is No (H-1,H-2)
R4: Release not provided is hazardous when Vehicle Motion is Stopped,Driver Present is Yes,Full Brake Applied is Yes,Vehicle Held is Yes (H-1,H-2)
R5: Release not provided is hazardous when Vehicle Motion is Stopped,Driver Present is Yes,Full Brake Applied is Yes,Vehicle Held is No,Gas Pedal is Yes (H-1,H-2)

Figure 29: Rule definitions for *Release*

Table 25: Context table for *Release* command automatically filled out based on "Rules" (Figure 29)

Control Action	Type	Vehicle Motion	Driver Present	Full Brake Applied	Vehicle Held	Gas Pedal	Hazards	Too Early/Too Late Hazards	Related Rules
Release	not provided when	Moving	Yes	Yes	Yes	Yes			
Release	not provided when	Moving	Yes	Yes	Yes	No			
Release	not provided when	Moving	Yes	Yes	No	Yes			
Release	not provided when	Moving	Yes	Yes	No	No			
Release	not provided when	Moving	Yes	No	Yes	Yes			
Release	not provided when	Moving	Yes	No	Yes	No			
Release	not provided when	Moving	Yes	No	No	Yes			
Release	not provided when	Moving	Yes	No	No	No			
Release	not provided when	Moving	No	Yes	Yes	Yes			
Release	not provided when	Moving	No	Yes	Yes	No			
Release	not provided when	Moving	No	Yes	No	Yes			
Release	not provided when	Moving	No	Yes	No	No			
Release	not provided when	Moving	No	No	Yes	Yes			
Release	not provided when	Moving	No	No	Yes	No			
Release	not provided when	Moving	No	No	No	Yes			
Release	not provided when	Moving	No	No	No	No			
Release	not provided when	Stopped	Yes	Yes	Yes	Yes	H-1: Vehicl		R4
Release	not provided when	Stopped	Yes	Yes	Yes	No	H-1: Vehicl		R4
Release	not provided when	Stopped	Yes	Yes	No	Yes	H-1: Vehicl		R5
Release	not provided when	Stopped	Yes	Yes	No	No			
Release	not provided when	Stopped	Yes	No	Yes	Yes			
Release	not provided when	Stopped	Yes	No	Yes	No			
Release	not provided when	Stopped	Yes	No	No	Yes			
Release	not provided when	Stopped	Yes	No	No	No			
Release	not provided when	Stopped	No	Yes	Yes	Yes			
Release	not provided when	Stopped	No	Yes	Yes	No			
Release	not provided when	Stopped	No	Yes	No	Yes			
Release	not provided when	Stopped	No	Yes	No	No			
Release	not provided when	Stopped	No	No	Yes	Yes			
Release	not provided when	Stopped	No	No	Yes	No			
Release	not provided when	Stopped	No	No	No	Yes			
Release	not provided when	Stopped	No	No	No	No			
Release	provided when	Moving	Yes	Yes	Yes	Yes	H-1: Vehicl	H-1: Vehicle dc	R1
Release	provided when	Moving	Yes	Yes	Yes	No	H-1: Vehicl	H-1: Vehicle dc	R1
Release	provided when	Moving	Yes	Yes	No	Yes	H-1: Vehicl	H-1: Vehicle dc	R1
Release	provided when	Moving	Yes	Yes	No	No	H-1: Vehicl	H-1: Vehicle dc	R1
Release	provided when	Moving	Yes	No	Yes	Yes			
Release	provided when	Moving	Yes	No	Yes	No			
Release	provided when	Moving	Yes	No	No	Yes			
Release	provided when	Moving	Yes	No	No	No			
Release	provided when	Moving	No	Yes	Yes	Yes	H-1: Vehicl	H-1: Vehicle dc	R1
Release	provided when	Moving	No	Yes	Yes	No	H-1: Vehicl	H-1: Vehicle dc	R1
Release	provided when	Moving	No	Yes	No	Yes	H-1: Vehicl	H-1: Vehicle dc	R1
Release	provided when	Moving	No	Yes	No	No	H-1: Vehicl	H-1: Vehicle dc	R1
Release	provided when	Moving	No	No	Yes	Yes			
Release	provided when	Moving	No	No	Yes	No			
Release	provided when	Moving	No	No	No	Yes			
Release	provided when	Moving	No	No	No	No			
Release	provided when	Stopped	Yes	Yes	Yes	Yes			
Release	provided when	Stopped	Yes	Yes	Yes	No			
Release	provided when	Stopped	Yes	Yes	No	Yes			
Release	provided when	Stopped	Yes	Yes	No	No	H-1: Vehicl	H-1: Vehicle dc	R3
Release	provided when	Stopped	Yes	No	Yes	Yes			
Release	provided when	Stopped	Yes	No	Yes	No			
Release	provided when	Stopped	Yes	No	No	Yes			
Release	provided when	Stopped	Yes	No	No	No			
Release	provided when	Stopped	No	Yes	Yes	Yes			
Release	provided when	Stopped	No	Yes	Yes	No			
Release	provided when	Stopped	No	Yes	No	Yes	H-1: Vehicl	H-1: Vehicle dc	R2
Release	provided when	Stopped	No	Yes	No	No	H-1: Vehicl	H-1: Vehicle dc	R2
Release	provided when	Stopped	No	No	Yes	Yes			
Release	provided when	Stopped	No	No	Yes	No			
Release	provided when	Stopped	No	No	No	Yes			
Release	provided when	Stopped	No	No	No	No			

Table 26: Simplified context table for *Release*

Controller: EBA								
Control Action	Context ID	Vehicle Motion	Driver Present	Full Brake Applied	Vehicle Held	Gas Pedal Applied	Not Providing causes Hazards	Providing causes Hazards
Release	EBA-R-1	Moving	*	No	*	*		
	EBA-R-2	Moving	*	Yes	*	*		Yes
	EBA-R-3	Stopped	No	Yes	No	*		Yes
	EBA-R-4	Stopped	No	Yes	Yes	*		
	EBA-R-5	Stopped	No	No	*	*		
	EBA-R-6	Stopped	Yes	Yes	Yes	*	Yes	
	EBA-R-7	Stopped	Yes	Yes	No	No		Yes
	EBA-R-8	Stopped	Yes	Yes	No	Yes	Yes	
	EBA-R-9	Stopped	Yes	No	*	*		

Control algorithm for 'Release' cmd			
Vehicle Motion=	Moving		
	Stopped	T	T
Driver Present=	Yes	T	T
	No		
Full Brake Applied=	Yes	T	T
	No		
Vehicle Held=	Yes	T	
	No		T
Gas Pedal=	Yes		T
	No		

Figure 30: SpecTRM-RL for *Release* generated automatically

4.3 Integrated Analysis for Identifying Hazardous Feature Interactions

4.3.1 Feature integrations in multiple control systems

This section applies the STPA tool to the integrated analysis of multiple control systems, as shown in Figure 31. In particular, it shows tool assistance to identify unsafe interactions among multiple controllers. In addition to two features presented in this chapter, the STPA tool uses the analysis results from the analysis of "Auto-Hold", "Engine Start/Stop", and "Adaptive Cruise Control" that have been done previously by Thomas and Placke [5][18].

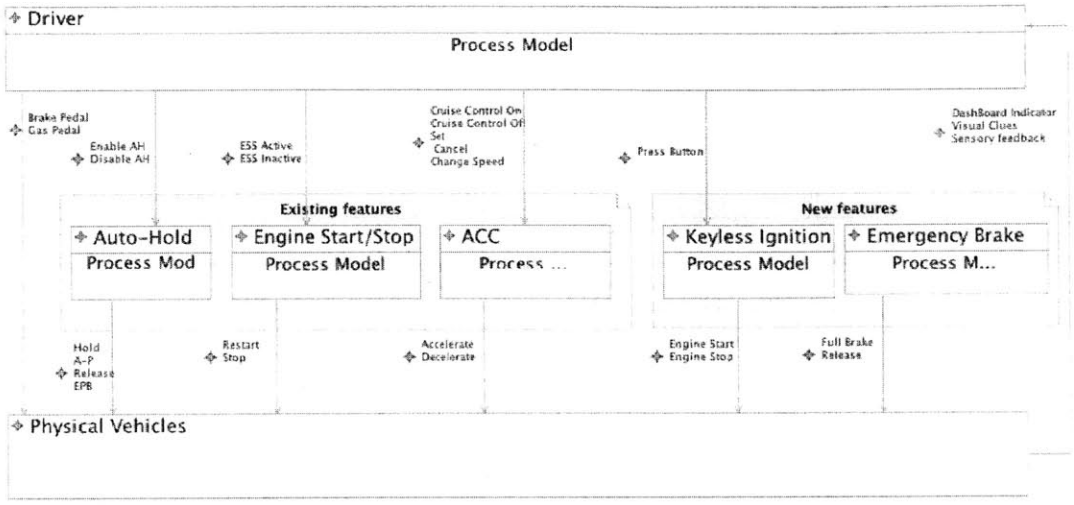


Figure 31: Integrating keyless ignition and EBA into the existing system

The STPA tool uses Step 1 results to generate design assumptions and conditions required for each control action. Engineers can then define the effects on systems of providing each control action. The tool will detect conflicts between the control actions of different controllers automatically. These results can help engineers identify unsafe interactions among controllers and make trade-offs in design decisions.

4.3.2 Create design assumptions and effects on systems

As described in section 3.5, the STPA tool leverages Step 1 results to generate design assumptions and the required conditions for keyless ignition and emergency braking assist automatically. Table 27 shows the conditions under which the *Engine Start* command must be provided by keyless ignition and its effects on the vehicle. Specifically, the second column specifies that keyless ignition must provide the command to start the engine when the driver pushes the button, the key is present in the vehicle, the transmission is in "park" or "neutral" range, and the engine status is off. The third column means that if the *Engine Start* command is provided under its required conditions, the engine will be turned on. Similarly, the required conditions and effects on the system for the *Engine Stop*, *Full Brake*, and *Release* commands are shown in Table 28, Table 29, and Table 30 respectively.

Table 27: Conditions table of *Engine Start* (keyless ignition)

Control Actions	Design Assumptions and Required Conditions	Effect on the system
Engine Start	Driver Selection: Yes Key Present: Yes Trans Range: P/N Engine Status: Off	Engine Status: On

Table 28: Conditions table of *Engine Stop* (keyless ignition)

Control Actions	Design Assumptions and Required Conditions	Effect on the system
Engine Stop	Driver Engine Off: Yes Vehicle Motion: Stopped Trans Range: P Engine Status: On	Key Present: No Engine Status: Off

Table 29: Conditions table of *Full Brake* (EBA)

Control Actions	Design Assumptions and Required Conditions	Effect on the system
Full Brake	Collision Risk: Yes Collision imminent: No Full Brake Applied: No Gas Pedal: No Brake Pedal: Yes -OR- Collision Risk: Yes Collision imminent: Yes Full Brake Applied: No Gas Pedal: No	Speed: Decreased Brakes: On (i.e. applied by EBA)

Table 30: Conditions table of *Release* (EBA)

Control Actions	Design Assumptions and Required Conditions	Effect on the system
Release	Vehicle Motion: Stopped Driver Present: Yes Full Brake Applied: Yes Vehicle Held: Yes -OR- Vehicle Motion: Stopped Driver Present: Yes Full Brake Applied: Yes Vehicle Held: No Gas Pedal: Yes	Brakes: Off (i.e. not applied by EBA)

4.3.3 Conflicts detected automatically by STPA tool

After design assumptions and effects on the system for each control action are available, the STPA tool detects conflicts between different controllers, as shown in Figure 32 and Figure 33.

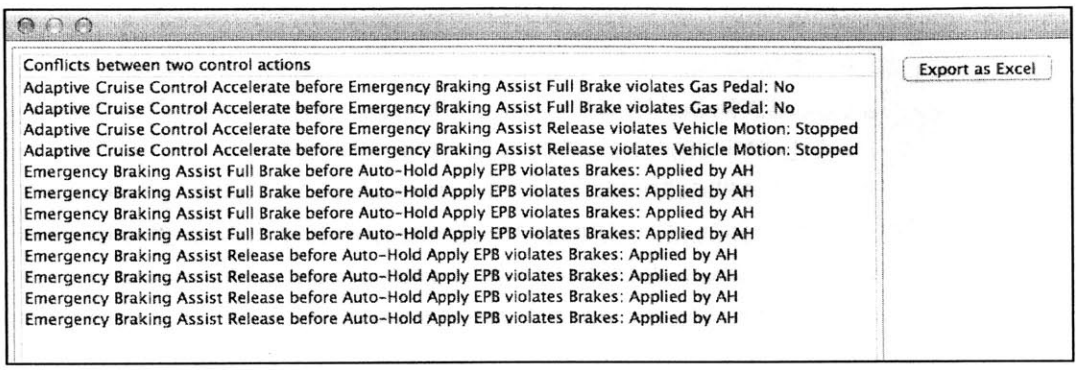


Figure 32: Conflicts between emergency brake assist and other features

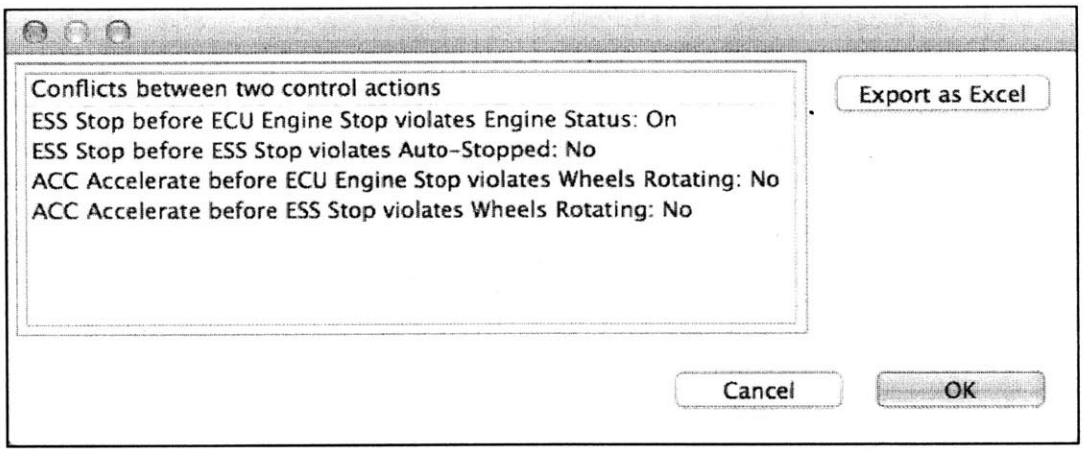


Figure 33: Conflicts between keyless ignition and other features

The conflict list displayed in the STPA tool can then be used to identify scenarios that will trigger unsafe interactions among the controllers. Three examples are given below.

Conflict between EBA and ACC

ACC providing *Accelerate* command before EBA provides *Release* command will violate the required condition *Vehicle Motion: Stopped*

Possible scenario:

- EBA initiates full powered brake automatically because a collision is "imminent"
- Vehicle speed and range rate decrease
- ACC applies the gas to increase vehicle speed before EBA releases the brake
- ACC and EBA fight with each other

Outcome: Vehicle has increased stopping distance when a collision is imminent

This conflict must be controlled or mitigated if EBA and ACC are provided on the same vehicles. As mentioned before, EBA not only initiates the full powered brake if the driver presses the brake

pedal and there is a collision risk, but also engages the emergency braking automatically even when the driver does not react if it believes the collision is imminent. In the latter case, EBA is designed such that it will not release the brake until the vehicle comes to a stop, according to the assumption made for this analysis (**Error! Reference source not found.**). At the same time, ACC continues to apply the gas pedal to increase vehicle speed if the range and range rate are below the predefined level in order to achieve its functional goals. As a consequence, the vehicle will have increased stopping distance in an emergency because one feature is applying the full powered brake while another one is accelerating the vehicle.

Conflicts between EBA and Auto-Hold

EBA providing *Full Brake* command before Auto-Hold provides EPB will violate the required condition *Brakes: Applied by AH*

EBA providing *Release* command before Auto-Hold provides EPB will violate the required condition *Brakes: Applied by AH*

Possible scenario:

- EBA initiates full powered brake and assumes control authority
- Vehicle comes to a stop
- EBA releases the brake before Auto-Hold captures the braking pressure
- Driver opens the door and leaves the vehicle
- Auto-hold does not apply the electronic parking brake as designed, since Auto-Hold has not been engaged

Outcome: Vehicle starts to roll

This interaction must be controlled when EBA and Auto-Hold are provided in the same vehicle. Auto-hold is designed to hold the vehicle and prevent it from unintended rollback. In situations where the driver leaves the vehicle without applying the parking brake, it can apply the electronic parking brake automatically to hold the vehicle [5]. However, the engagement of the EBA violates the required condition of Auto-Hold assuming that EPB,that is, auto-hold,must have control authority over the brake in order to capture the remaining braking pressure after the vehicle comes to a stop. If this happens, the EBA will prevent the Auto-Hold from holding the vehicle and applying the electronic parking brake. It should be noted that these conflicts would not occur in reality if Auto-Hold were still able to capture the braking pressure when EBA is engaged. However, during conceptual development, when detailed design information is not

available, it is necessary for the STPA tool to generate all conflicts to ensure that they will all be addressed.

Conflict between keyless ignition and engine start/stop (ESS):
ESS providing *STOP* command before keyless ignition provides *STOP* command will violate the required condition *Engine: On*
Possible scenario:

- The vehicle comes to a stop on an uphill
- ESS turns off the engine temporarily
- Driver shifts to "park" and presses the "Engine Start/Stop" button (accidentally/on purpose)
- Vehicle turns off (instead of engine restarting)
- Driver releases the brake pedal (may expect the vehicle to resume the motion)
- Engine is not restarted; vehicle rolls backwards

Outcome: The vehicle starts to roll backwards downhill

This interaction must be controlled if keyless ignition and ESS are installed in the same vehicle. It is interesting to note that this feature involves conflicts of two non-safety goals. Keyless ignition is designed to provide more flexibility and convenience to the driver, while ESS is a "green" feature that can save fuel consumption by turning off the engine automatically when the vehicle comes to a stop and restarting it before motion resumes [5]. The engine can also be restarted by commands from the ESS if it is turned off automatically rather than by input from the driver.

The conflict above can cause the driver to be unaware of the real status of the engine. He or she is not only confused by the automated behavior of the ESS in shutting down the engine, but also cannot deduce the status of the vehicle because of design changes in the ignition system. For the driver who is not familiar with this automated feature, activating or deactivating the engine automatically can make him believe that the engine is off while it is not and he or she may try to restart it by pressing the push button. If this happens when the engine is temporarily shut down by the ESS, the vehicle will not be restarted by the ESS before motion resumes. In addition, the driver cannot tell the engine status based on the information of whether the key fob is in the ignition, which is available in traditional vehicle designs.

5 Discussion and Conclusions

This thesis answers the research questions mentioned in section 1.3 by building an open-source software tool that supports STPA and requirement generation and applies it to a case study of automotive systems.

Chapter 3 shows how engineers can perform STPA with tool assistance. As illustrated in the case study of EBA and keyless ignition, the STPA tool partially automates Step 1 analysis and requirement generation. For complex systems that operate in multiple states and have different hazards (goals) to eliminate (achieve), engineers can identify unsafe control actions with less time and effort.

In addition to providing guidance for identifying unsafe control actions, the STPA tool can also help engineers make decisions at the architectural level. In particular, this tool could be used as a way to check whether conflicts between features have been resolved and to identify exactly what decisions should be made jointly between multiple design teams. Take the conflict between EBA and Auto-Hold detected by the tool in section 4.3.3 as an example. Engineers must consider the potential consequences caused by this conflict and resolve it. If the available design information indicates that Auto-Hold will capture the braking pressure before EBA releases the brake, this conflict can be ignored. Otherwise, engineers who integrate EBA into vehicle models may have to make arbitrary decisions about control authorities over the braking system when ACC, EBA, and other features that have control authority over the brake engage simultaneously. This is reasonable since both ACC and EBA use feedback information (i.e., range and range rate) from the radar for the speed control of vehicles.

For tool assistance during requirements engineering, the STPA tool can apply logical simplification to the original context tables and automatically generate the simplified requirement in SpecTRM-RL. The simplified requirement is easily understandable and addresses all of the UCAs identified in the original context tables, as shown in section 4.1.2. In addition, the SpecTRM-RL requirements of EBA and keyless ignition that are generated automatically by the STPA tool can be executed in existing tools like SpecTRM and they can be formally checked for other properties like completeness. Every time new changes are made, these SpecTRM-RL models can be updated and checked again for any new problems.

In the future, the concept and work related to tool-assisted hazard analysis may be improved further in several respects:

- **Extend the tool to provide assistance for Step 2 of STPA.** It would be helpful to engineers who perform Step 2 if the STPA tool can provide more guidance for identifying causal scenarios contributing to UCAs. Specifically, the STPA tool may generate causal scenarios related to process model flaws and unsafe interactions among controllers and refined requirements for detailed design decisions. In addition, the tool can help maintain the traceability between high-level goals (hazards) and UCAs and causal scenarios in the lower level. This would be useful when engineers come up with design solutions to mitigate hazards and want to evaluate their impact on system-level goals.
- **Use requirements generated by the STPA tool to guide the design process.** Designers can use these safety requirements to design algorithms for automation and user interfaces that interact directly with human operators. SpecTRM-RL models can be updated as more design decisions are made and refined for formal checking at a more detailed level.
- **Provide more guidance for engineers who perform hazard analysis using STPA.** Previous work [5][18] illustrates that many automotive systems not only share a generic safety control structure, but also common process model variables. Engineers can use this "general" information stored on the STPA tool to start the analysis and refine the safety control structure when more design information is available.
- **Extend tool-assisted hazard analysis techniques to look at other system emergent properties such as security, quality, and sustainability, etc.** Previous work [5][31] has applied a systems theoretic approach to ensure that functional goals can be achieved while also preventing hazards. Given that all these goals are emergent properties of the system, it is possible to treat them by top-down systems engineering approaches such as STPA. The STPA tool can be shaped and extended to meet special requirements for such analysis.

Bibliography

[1] Weiss, Kathryn A., Elwin C. Ong, and Nancy G. Leveson. "Reusable specification components for model-driven development." *Proceedings of the INCOSE 2003 International Symposium,* Crystal City, VA. 2003.

[2] Leveson, Nancy G., and Kathryn Anne Weiss. "Making embedded software reuse practical and safe." In *ACM SIGSOFT Software Engineering Notes*, vol. 29, no. 6, pp. 171–178. ACM, 2004.

[3] Thomas, John, John Sgueglia, Dajiang Suo, Nancy Leveson, Mark Vernacchia, and Padma Sundaram. *An Integrated Approach to Requirements Development and Hazard Analysis.* No. 2015-01-0274. SAE Technical Paper, 2015.

[4] Leveson, Nancy G. "Engineering a safer world." *Systems Thinking Applied to Safety*, The MIT Press, Cambridge, MA, 2011.

[5] Placke, Matthew Seth. Application of STPA to the integration of multiple control systems: A case study and new approach. *Massachusetts Institute of Technology*, 2014.

[6] Leveson, Nancy. "A systems approach to risk management through leading safety indicators." *Reliability Engineering & System Safety* 136 (2015): 17–34.

[7] Safety Research & Strategies Inc. *NHTSA Opens Smart Key Compliance Probe.* Available at: http://www.safetyresearch.net/blog/articles/nhtsa-opens-smart-key-compliance-probe

[8] NHTSA Defect Investigation Results PE11-025. Available at: http://www-odi.nhtsa.dot.gov/cars/problems/defect/results.cfm?action_number=PE11025&SearchType=QuickSearch&summary=true

[9] N. Leveson, L. Pinnel, S. Sandys, S. Koga, and J. Reese. "Analyzing software specifications for mode confusion potential." In *Proceedings of a Workshop on Human Error and System Development*, pp. 132–146, 1997.

[10] N. Leveson, and E. Palmer. "Designing automation to reduce operator errors." In *Proceedings of the IEEE Systems, Man, and Cybernetics Conference,* October 1997.

[11] R. Mario, M. Zimmerman, M. Katahira, M. D. Villepin, B. Ingram, and N. Leveson. "Identifying mode confusion potential in software design." In *Digital Avionics Systems Conference, 2000 Proceedings.* DASC, 19, vol. 2, pp. 5D2–1. IEEE, 2000.

[12] M. Heimdahl, and N. Leveson. "Completeness and consistency in hierarchical state-based requirements." *Software Engineering, IEEE Transactions* 22, 6, 363–377, 1996.

[13] N. Leveson, M. Heimdahl, H. Hildreth, and J. Reese. "Requirements specification for process-control systems." *Software Engineering, IEEE Transactions* 20, 9, 684–707, 1994.

[14] N. Leveson. "Intent specifications: An approach to building human-centered specifications." In *Requirements Engineering, 1998. Proceedings.* 1998 Third International Conference, pp. 204-213. IEEE, 1998.

[15] J. Howard. "Preserving System Safety across the Boundary between System Integrator and Software Contractor," *SAE Technical Paper* 2004-01-1663, 2004, doi:10.4271/2004-01-1663.

[16] M. Zimmerman, M. Rodriguez, B. Ingram, M. Katahira, M. D. Villepin, and N. Leveson. "Making formal methods practical." In *Digital Avionics Systems Conference, 2000 Proceedings.* DASC 19, 1, pp. 1B2–1. IEEE, 2000.

[17] B. Molly, and N. Leveson. "Modeling controller tasks for safety analysis." In *Workshop on Human Error and System Development*, 1998.

[18] S. Placke, J. Thomas, and D. Suo. Integration of Multiple Active Safety Systems using STPA. No. 2015-01-0277. *SAE Technical Paper*, 2015.

[19] Q. D. V. E. Hommes. Assessment of the ISO 26262 Standard, "Road Vehicles – Functional Safety". *SAE 2012 Government/Industry Meeting*, 2012.

[20] ISO 26262, https://en.wikipedia.org/wiki/ISO_26262#Vocabulary

[21] R. S. Martinez. *Comparing STPA and FMEA on an Automotive Electric Power Steering System.* Master's Thesis, 2015.

[22] N. Leveson. *Safeware System Safety and Computers*, Boston MA: Addison-Wesley, 1995.

[23] Project Reliability Group. *Jet Propulsion Laboratory Reliability Analysis Handbook*, 1990.

[24] Vesely, William E., et al. Fault tree handbook. No. NUREG-0492. Nuclear Regulatory Commission Washington DC, 1981.

[25] I. Navarro, K. Lundqvist, and N. Leveson. "An intent-specifications model for a robotic software control system." In *Digital Avionics Systems, 2001.* DASC. 20th Conference, vol. 2, pp. 8E1–1. IEEE, 2001.

[26] I. Takuto, N. Leveson, C. Fleming, M. Katahira, Y. Miyamoto, and H. Nakao. "Multiple Controller Contributions to Hazards." In *5th IAASS Conference*, Versailles, France, 2011.

[27] A. Adhulkhaleq and S. Wagner. "Tool Support for STPA," in *STAMP Workshop 2014*, Cambridge, MA, 2014.

[28] Q. D. V. E. Hommes. "The Volpe STPA Tool," in *STAMP Workshop 2014*, Cambridge, MA, 2014.

[29] J. Thomas, and D. Suo. "An STPA Tool," in *STAMP Workshop 2014*, Cambridge, MA, 2014.

[30] F. Budinsky. *Eclipse modeling framework: A developer's guide.* Addison-Wesley Professional, 2004.

[31] J. Thomas. *Extending and Automating a Systems-Theoretic Hazard Analysis for Requirements Generation and Analysis*, Ph.D. Thesis, Engineering Systems Division, Massachusetts Institute of Technology, 2013.

[32] F. Modugno, and N. Leveson. "Integrated safety analysis of requirements specifications." *Requirements Engineering* 2, no.2, 65–78, 1997.

[33] N. Leveson. "Completeness in formal specification language design for process-control systems." In *Proceedings of the Third Workshop on Formal Methods in Software Practice*, pp. 75–87. ACM, 2000.

[34] N. Dulac, T. Viguier, and N. Leveson. "On the use of visualization in formal requirements specification." In *Requirements Engineering*, 2002.

[35] National Highway Traffic Safety Administration (NHTSA), *Federal Motor Vehicle Safety Standards* 114.

[36] National Highway Traffic Safety Administration (NHTSA), *Federal Motor Vehicle Safety Standards* 102.

[37] Trinidad, Omar and Dixon, Matt. "Push Button Start: The new ignition switch" (2009). *Presentations. Paper 9.* http://opensiuc.lib.siu.edu/auto_pres/9

[38] The Eclipse projects. Available at: http://www.eclipse.org

[39] National Highway Traffic Safety Administration (NHTSA). *Recalls and Defects.* Available at: http://www.nhtsa.gov/Vehicle+Safety/Recalls+&+Defects

[40] C. Fleming. *Safety-Driven Early Concept Analysis and Development*, Ph.D. Thesis, Massachusetts Institute of Technology, 2015.

[41] B. Antoine. *Systems Theoretic Hazard Analysis (STPA) Applied to the Risk Review of Complex Systems: An Example from the Medical Device Industry*, Ph.D. Thesis, Engineering Systems Division, Massachusetts Institute of Technology, 2013.

[42] T. Ishimatsu, N. Leveson, J. Thomas, C. Fleming, M. Katahira, Y. Miyamoto, R. Ujiie, H. Nakao, and N. Hoshino. "Hazard Analysis of Complex Spacecraft using Systems-Theoretic Process Analysis," *Journal of Spacecraft and Rockets*, vol. 51, no. 2, pp. 509–522, 2014.

[43] S. Kawakami. *Application of a Systems-Theoretic Approach to Risk Analysis of High-speed Rail Project Management in the US*, Master's Thesis, Engineering Systems Division, Massachusetts Institute of Technology, 2013.

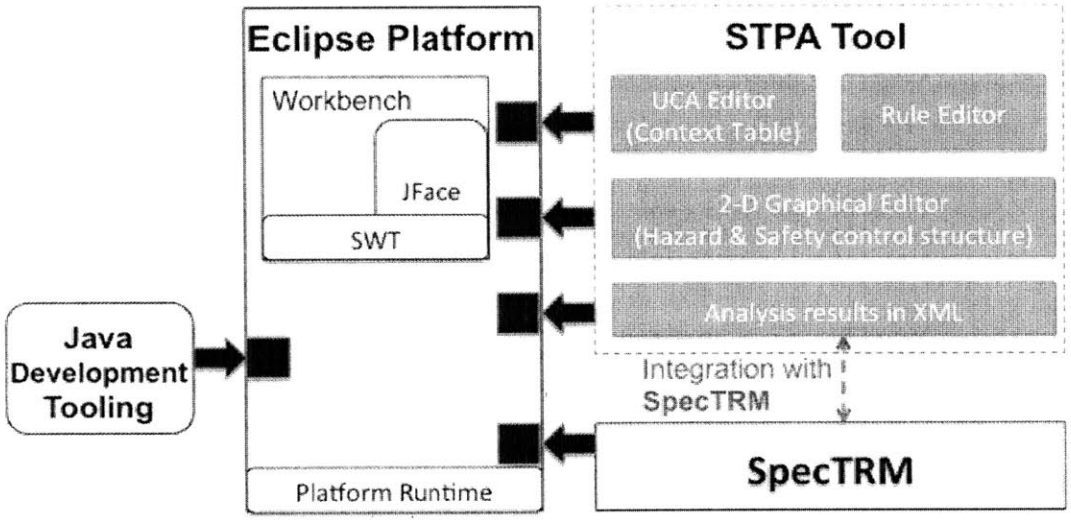
[44] Q. D. V. E. Hommes. "Applying STAMP Framework to Analyze Automotive Recalls," in *STAMP Workshop 2014*, Cambridge, MA, 2014

[45] National Highway Traffic Safety Administration (NHTSA), information about vehicle recalls, available at: http://www-odi.nhtsa.dot.gov/owners/SearchSafetyIssues

Appendix A: A Quick Guide to the STPA Tool

Disclaimer: The STPA tool described in the thesis was not developed for professional use. It is just a prototype proof of concept meant to demonstrate the ideas in the author's thesis and was created by one Master's student, rather than a team of professional software developers. The author does not guarantee that it does not include any bugs. This tool was not created to satisfy any of the standards for tool development and the author suggests that users should not rely on it for the analysis of safety-critical or security-critical systems. Whoever tries the STPA tool must only use it for non-commercial purposes.

Overview of the STPA tool



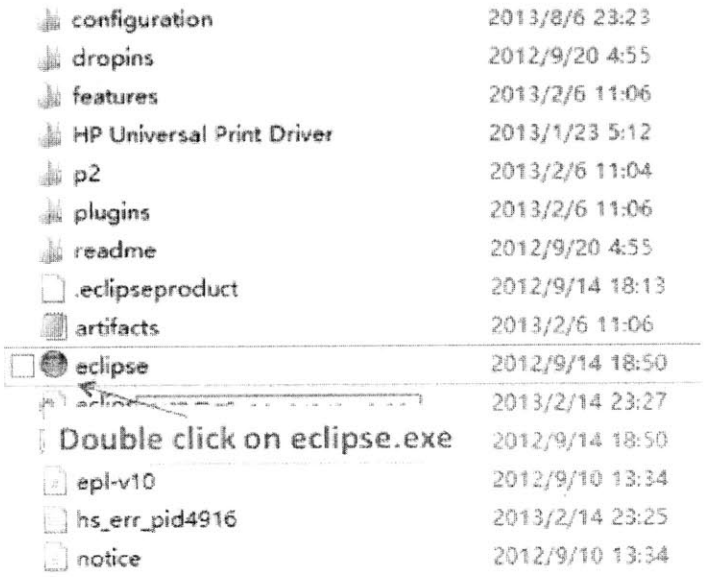
The STPA tool allows engineers to:

- Specifying hazards
- Drawing the safety control structure
  - Adding controllers, controlled processes
  - Adding actuators and sensors
  - Adding control actions and feedback
  - Adding process model variables and values
- Performing STPA Step 1
  - Generating context table templates automatically based on control structure
  - Allowing engineers to specify unsafe control actions
  - Allowing engineers to define "Rules", which are used to automatically complete context tables and identify unsafe control actions

- Storing analysis results in XML files for interoperation with other tools
- Generating safety requirements for completeness and consistency checking in SpecTRM
- Performing integrated analysis of multiple controllers

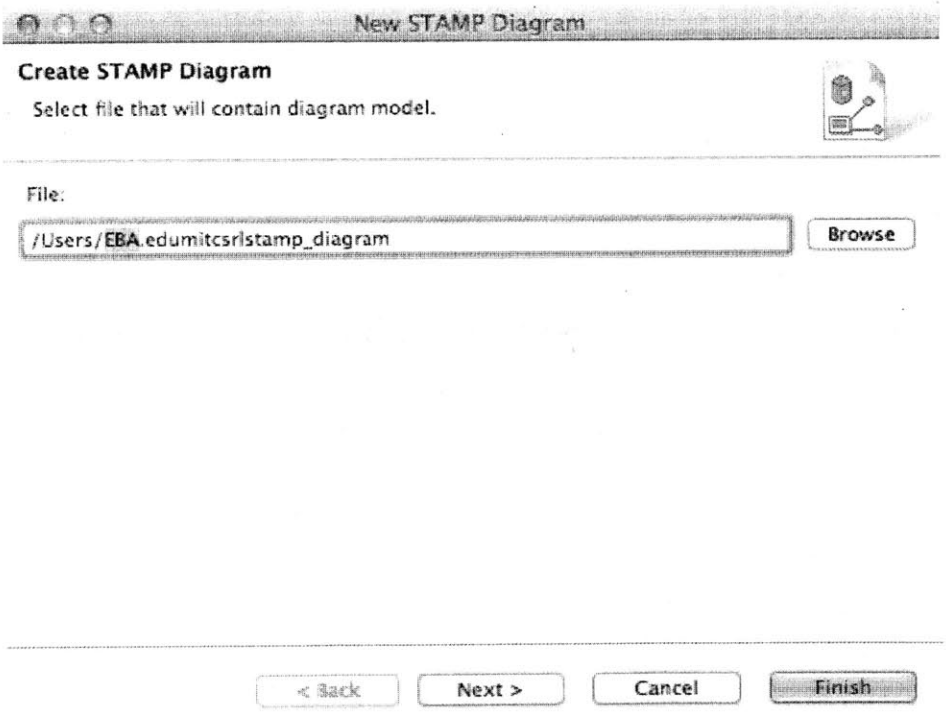
The tool assistance for performing STPA Step 2, i.e., the generation of causal scenarios, is planned for the future.

Launch the STPA Tool



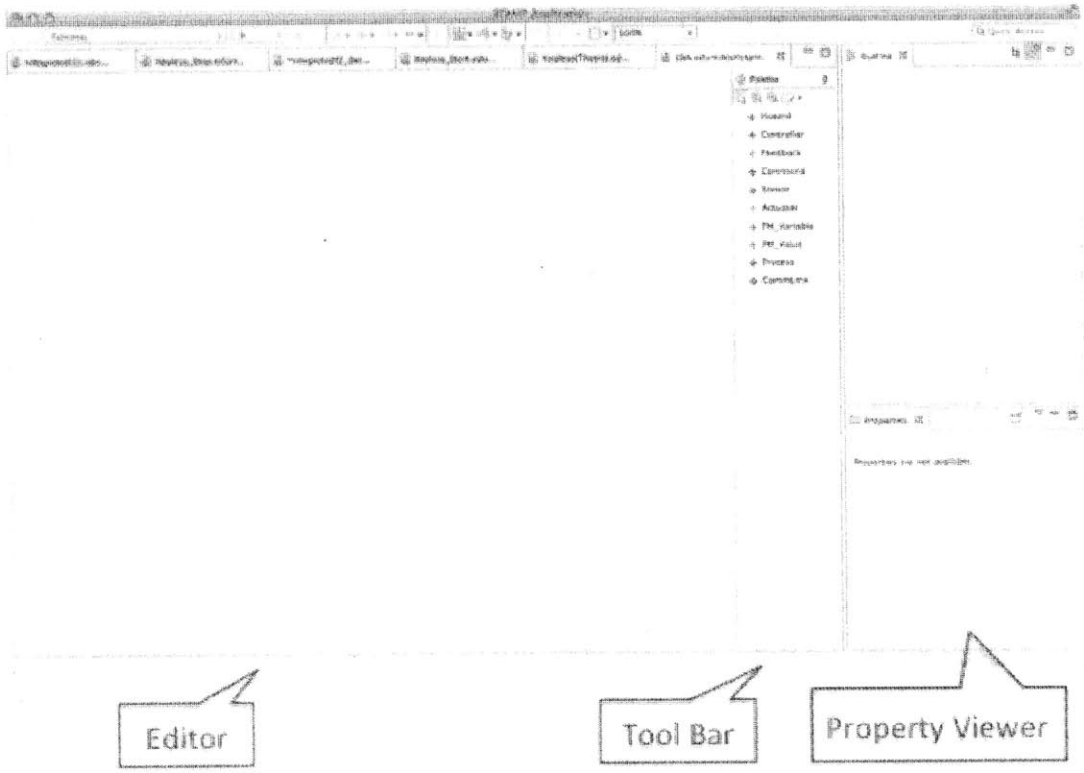
Launch STPA tool by double clicking on eclipse.exe(Win)/eclipse.app(Mac)

Set up Diagram & Model Files



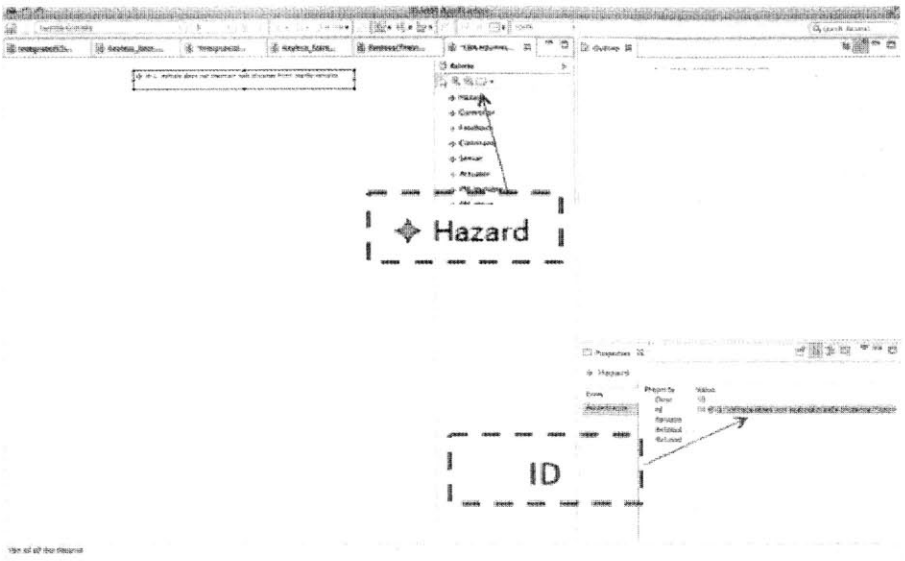
- Create a new STAMP Diagram by left click on File->New->STAMP Diagram
- Rename the diagram file, but do not change the suffix (This is the file to store graphical information such as size and position)
- Click on "Finish" on the bottom of the dialog (New STAMP Model)

2-D Editor for drawing Safety Control Structure



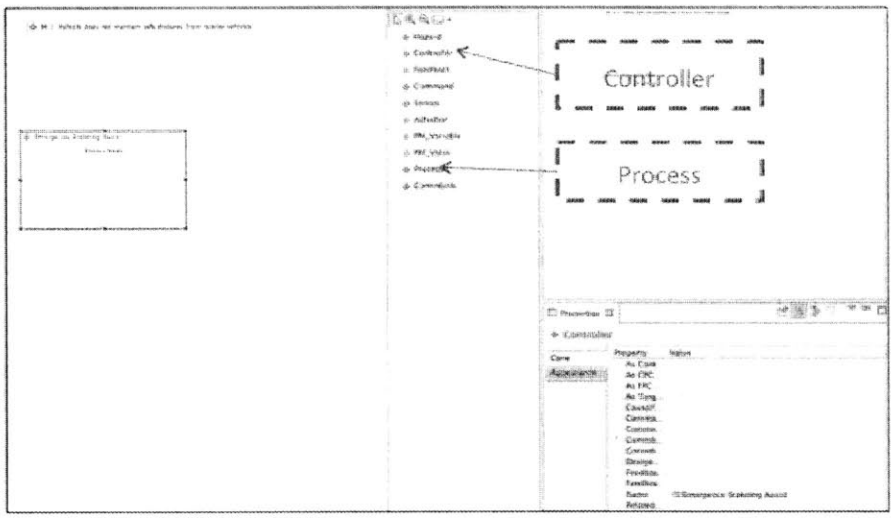
- **Editor** is used to place the safety control structure;
- **Tool Bar** is used to choose the components or links to add;
- **Property Viewer** is used to edit the properties of components such as name or Id.

Add System Hazards



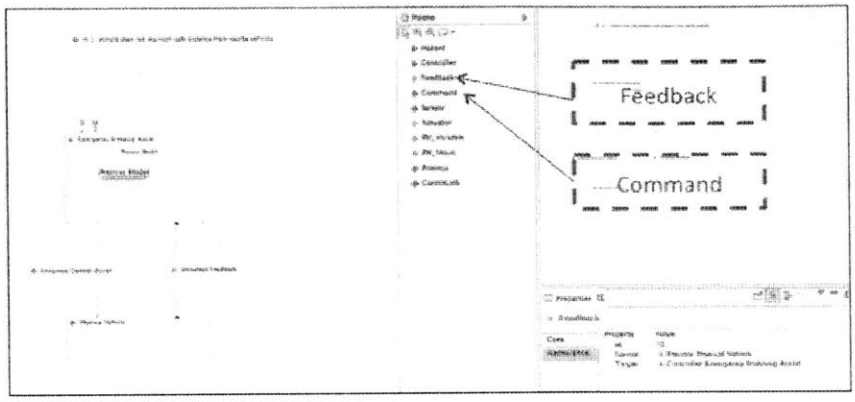
- To create hazards, left click on the icon of "Hazard" in the tool bar (center), then left click on any place in the Editor (left) you want to put the Hazard frame.
- Drug the frame to adjust it to the preferred size.
- Editing the "Id" of hazard in the "Properties Viewer" (bottom right).

Add Components to Safety Control Structure



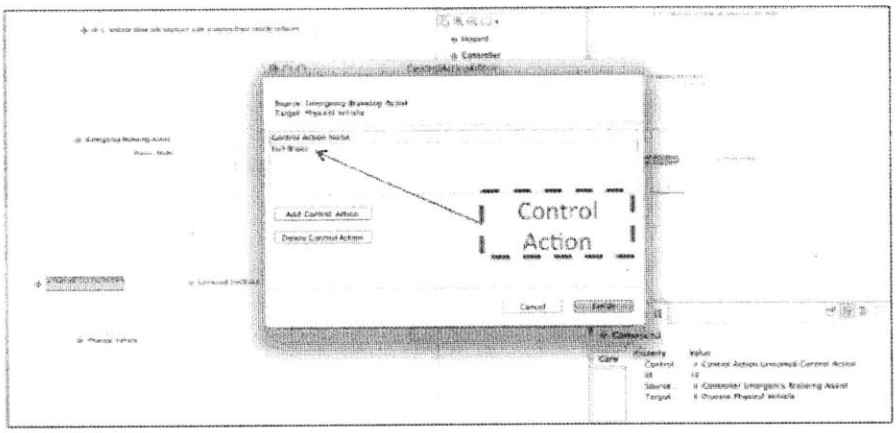
- To create components, left click on the icon such as "Controller" or "Process" in the tool bar (center), then left click on any place in the Editor (left).
- Drag the frame to adjust it to your preferred size.
- Edit the "Name" of different components in the "Properties Viewer" (bottom right).

Add Command&Feedback links to Safety Control Structure



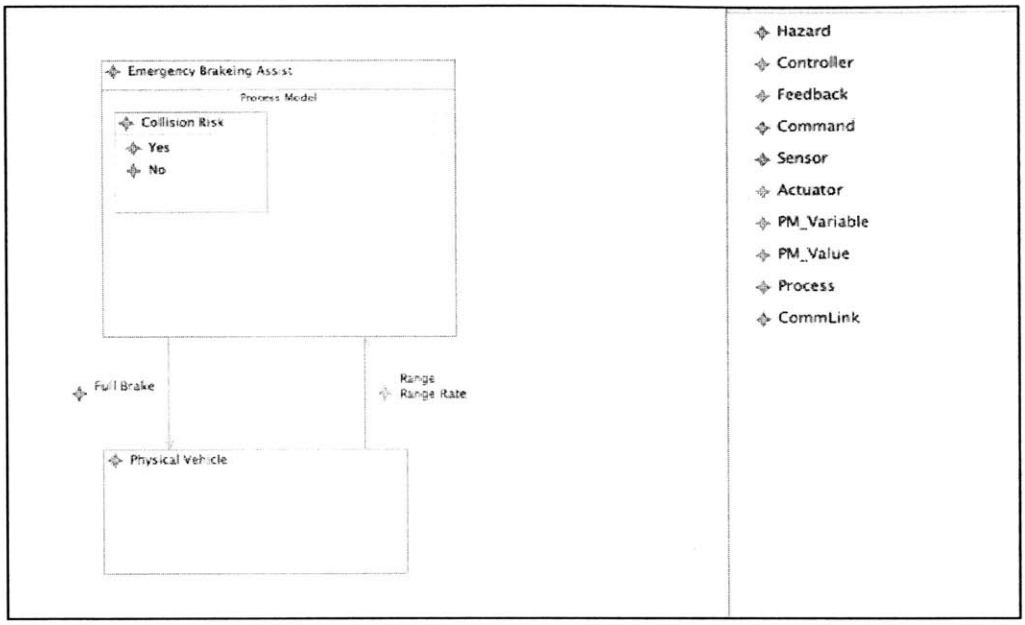
- To create links, left click on the icon of "Feedback" or " Command" in the tool bar (center), then left click on the source component (do not release), then drag to the target component.
- Make sure to differentiate between feedback and command.

Add Control Actions



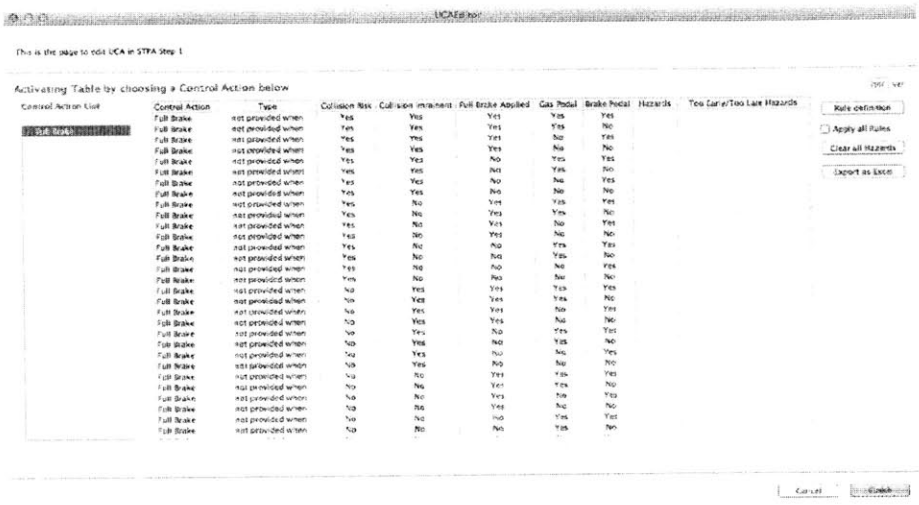
- Activate the Control Action Editor by right clicking on the label of Command link("Unnamed Control Action") ->Add Control Action
- Left Click on the "Unnamed Control Action" to change the name of Control Action
- Click on "Add Control Action" Button to add new Control Action
- Repeat step 2 to change its name
- To delete a Control Action, first select it in the Table, then click on "Delete Control Action"

Add Process Model Variables and Values



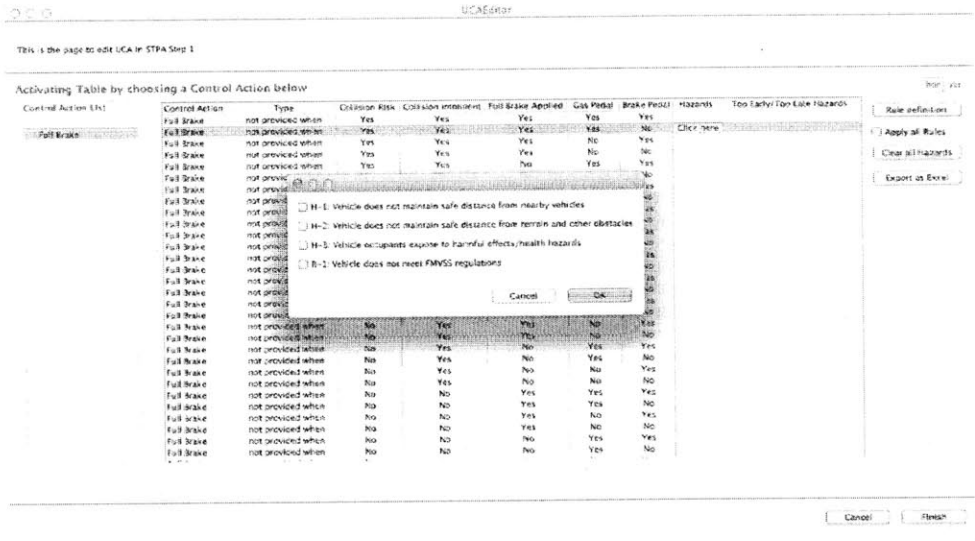
- To add PMV to the controller (in this case - "Train Controller"), left click on the icon of "PM_Varaible", then left click on the center of the controller(not the title part), then edit the name of PMV in the "Properties Viewer" (bottom right).
- To add Value to PMV, left click on the icon of "PM_Value", then left click on the center of the PMV object (not the title part), then edit the name of PV_Value in the "Properties Viewer" (bottom right).

Generate Low-level Context Tables Automatically



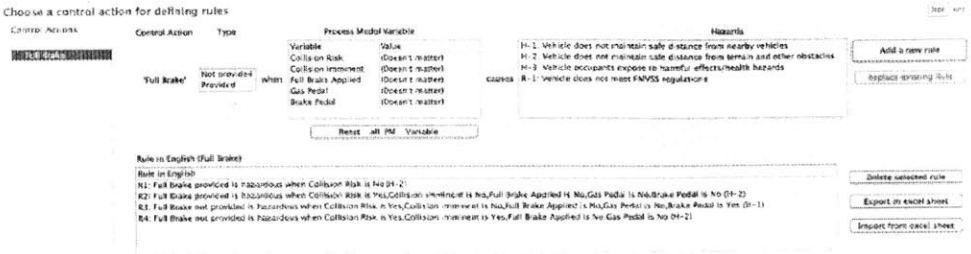
- To generate Context Table automatically, right click on the controller (in this case "Train Controller") ->Idenfity UCA
- Choose a Control Action (on the left) you would like to analyze

Identify UCAs based on Low-level Context Tables



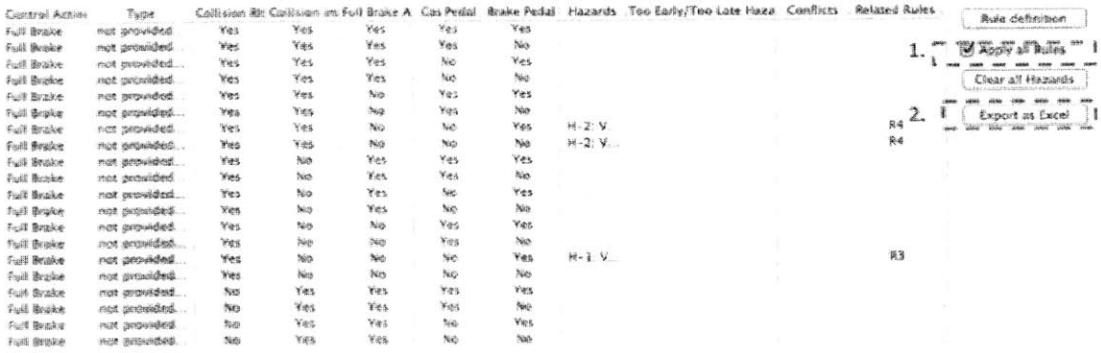
- To open the dialog for assigning Hazards, click on the cell in the "Hazards" column
- Click on the check box of the Hazard you would assign to this UCA

Define Rules in the *Rule Definition Editor*

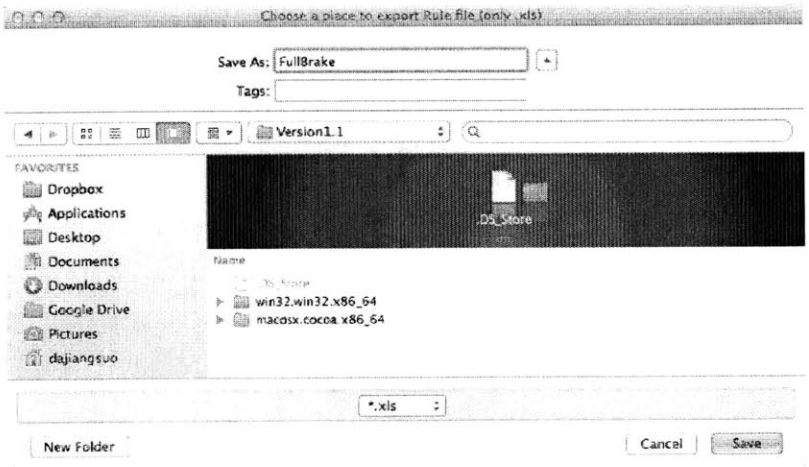


- To activate Rule Definition Editor, click on the "Rule Definition" Button in the UCA Editor
- Create Rule by choosing Type (Not Provided/Provide), values of PMV and related Hazards and clicking "Add a new Rule" button

Apply Rules to generate Low-level Context Tables



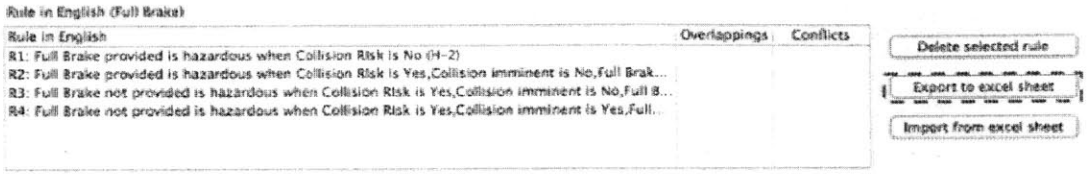
- Go back to the UCA Editor by clicking on the "Finish" button in Rule Editor
- Click on the check box "Apply Rules" to complete the Hazards column automatically
- Click on the button "Export as Excel" and input the name file for the context table



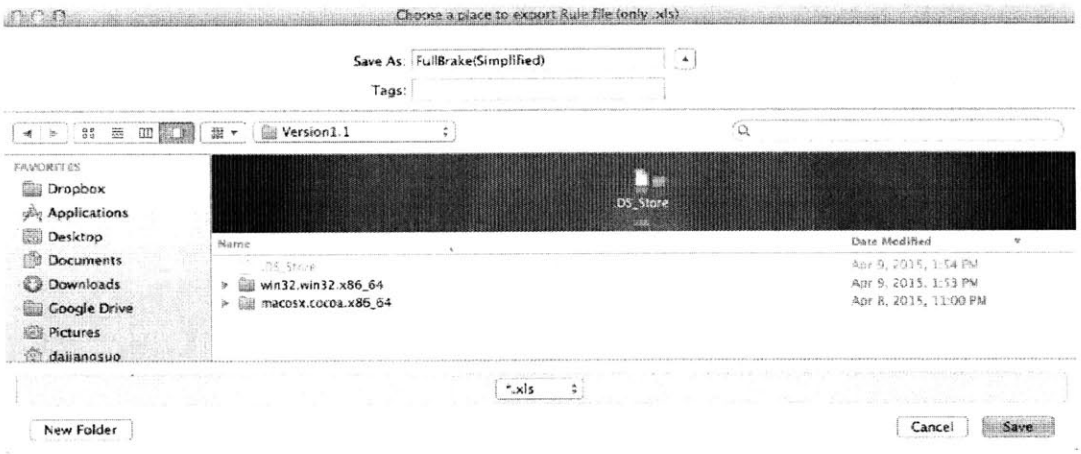
- Open the excel file for the low-level context table

Control Action	Type	Collision Risk	Collision imminent	Full Brake Applied	Gas Pedal	Brake Pedal	Hazards	Too Early/Too Late Hazards	Related Rules
Full Brake	not provided when	Yes	Yes	Yes	Yes	Yes			
Full Brake	not provided when	Yes	Yes	Yes	Yes	No			
Full Brake	not provided when	Yes	Yes	Yes	No	Yes			
Full Brake	not provided when	Yes	Yes	Yes	No	No			
Full Brake	not provided when	Yes	Yes	No	Yes	Yes			
Full Brake	not provided when	Yes	Yes	No	Yes	No			
Full Brake	not provided when	Yes	Yes	No	No	Yes	H-2: Vehi		R4
Full Brake	not provided when	Yes	Yes	No	No	No	H-2: Vehi		R4
Full Brake	not provided when	Yes	No	Yes	Yes	Yes			
Full Brake	not provided when	Yes	No	Yes	Yes	No			
Full Brake	not provided when	Yes	No	Yes	No	Yes			
Full Brake	not provided when	Yes	No	Yes	No	No			
Full Brake	not provided when	Yes	No	No	Yes	Yes			
Full Brake	not provided when	Yes	No	No	Yes	No			
Full Brake	not provided when	Yes	No	No	No	Yes	H-1: Vehi		R3
Full Brake	not provided when	Yes	No	No	No	No			
Full Brake	not provided when	No	Yes	Yes	Yes	Yes			
Full Brake	not provided when	No	Yes	Yes	Yes	No			
Full Brake	not provided when	No	Yes	Yes	No	Yes			
Full Brake	not provided when	No	Yes	Yes	No	No			
Full Brake	not provided when	No	Yes	No	Yes	Yes			
Full Brake	not provided when	No	Yes	No	Yes	No			
Full Brake	not provided when	No	Yes	No	No	Yes			
Full Brake	not provided when	No	Yes	No	No	No			
Full Brake	not provided when	No	No	Yes	Yes	Yes			
Full Brake	not provided when	No	No	Yes	Yes	No			
Full Brake	not provided when	No	No	Yes	No	Yes			
Full Brake	not provided when	No	No	Yes	No	No			
Full Brake	not provided when	No	No	No	Yes	Yes			
Full Brake	not provided when	No	No	No	Yes	No			
Full Brake	not provided when	No	No	No	No	Yes			
Full Brake	not provided when	No	No	No	No	No			
Full Brake	provided when	Yes	Yes	Yes	Yes	Yes			
Full Brake	provided when	Yes	Yes	Yes	Yes	No			
Full Brake	provided when	Yes	Yes	Yes	No	Yes			
Full Brake	provided when	Yes	Yes	Yes	No	No			
Full Brake	provided when	Yes	Yes	No	Yes	Yes			
Full Brake	provided when	Yes	Yes	No	Yes	No			
Full Brake	provided when	Yes	Yes	No	No	Yes			
Full Brake	provided when	Yes	Yes	No	No	No			
Full Brake	provided when	Yes	No	Yes	Yes	Yes			
Full Brake	provided when	Yes	No	Yes	Yes	No			
Full Brake	provided when	Yes	No	Yes	No	Yes			
Full Brake	provided when	Yes	No	Yes	No	No			
Full Brake	provided when	Yes	No	No	Yes	Yes			
Full Brake	provided when	Yes	No	No	Yes	No			
Full Brake	provided when	Yes	No	No	No	Yes			
Full Brake	provided when	Yes	No	No	No	No	H-2: Vehi	H-2: Vehicle do	R2
Full Brake	provided when	No	Yes	Yes	Yes	Yes	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	Yes	Yes	Yes	No	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	Yes	Yes	No	Yes	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	Yes	Yes	No	No	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	Yes	No	Yes	Yes	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	Yes	No	Yes	No	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	Yes	No	No	Yes	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	Yes	No	No	No	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	No	Yes	Yes	Yes	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	No	Yes	Yes	No	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	No	Yes	No	Yes	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	No	Yes	No	No	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	No	No	Yes	Yes	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	No	No	Yes	No	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	No	No	No	Yes	H-2: Vehi	H-2: Vehicle do	R1
Full Brake	provided when	No	No	No	No	No	H-2: Vehi	H-2: Vehicle do	R1

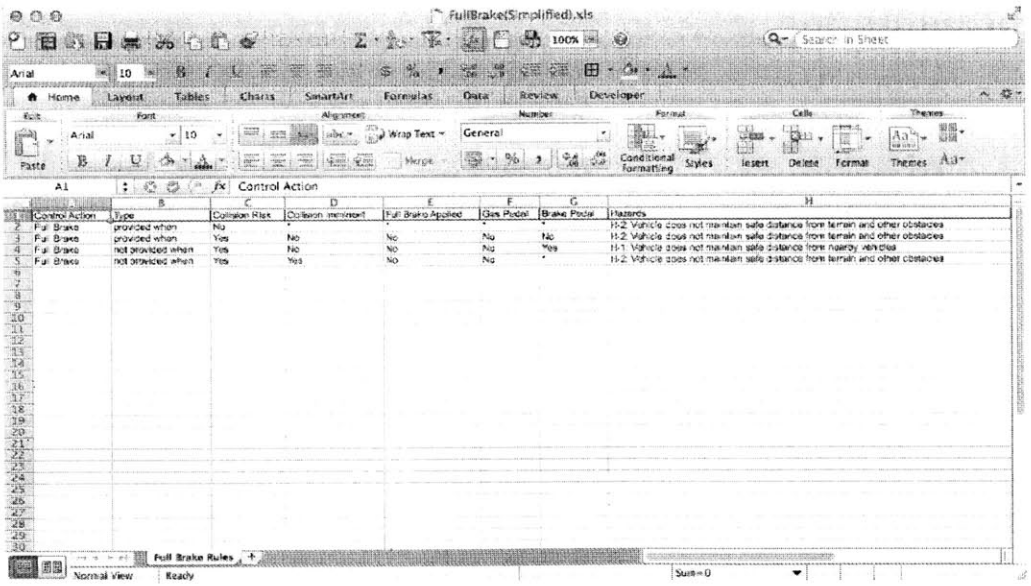
Generate simplified Context Tables and SpecTRM-RL



- In the Rule definition editor, click on "Export to excel sheet"
- Input the file name for the simplified context table



- Open the excel file for the simplified context table



Rule in And/Or Table

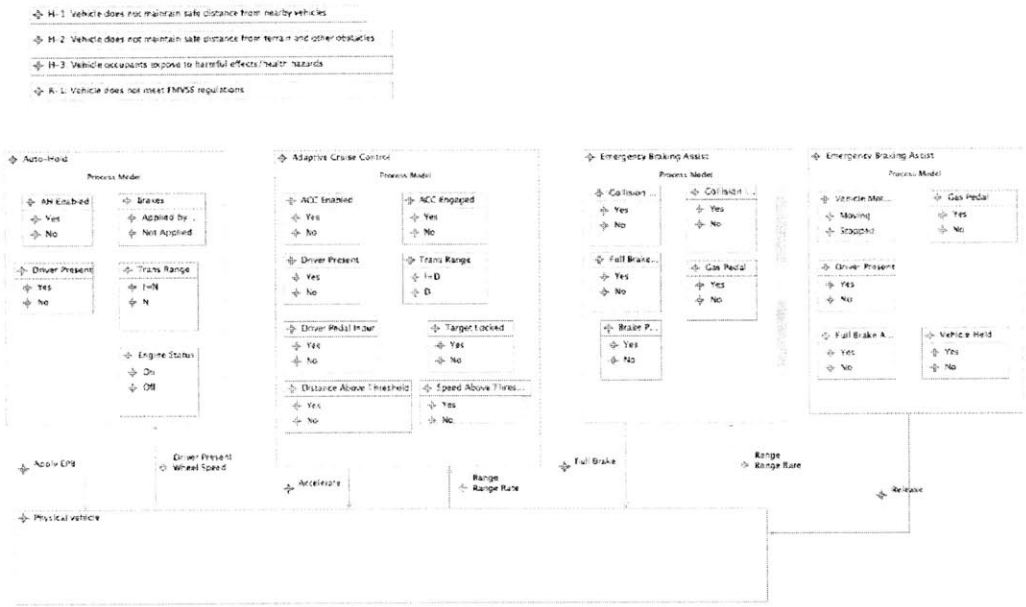
Control algorithm for 'Full Brake' cmd

Collision Risk=	Yes	T	T	
	No			
Collision immine...	Yes		T	
	No	T		
Full Brake Applied=	Yes			
	No	T	T	
Gas Pedal=	Yes			
	No	T	T	
Brake Pedal=	Yes	T		
	No			

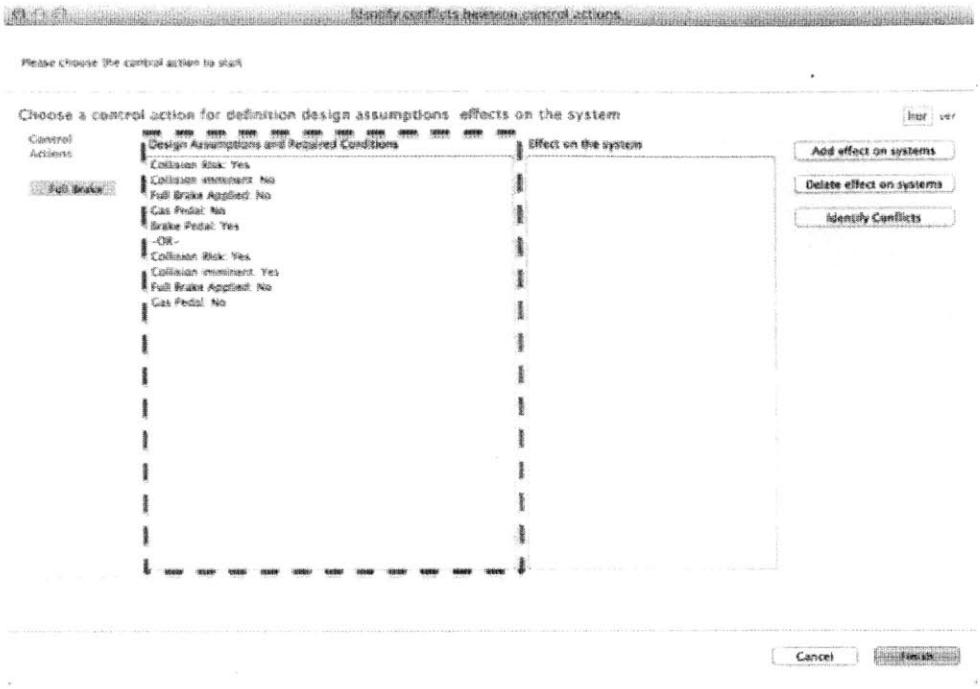
- The SpecTRM-RL requirements are also generated automatically in the Rule Editor

Integrated Analysis for identifying unsafe interactions

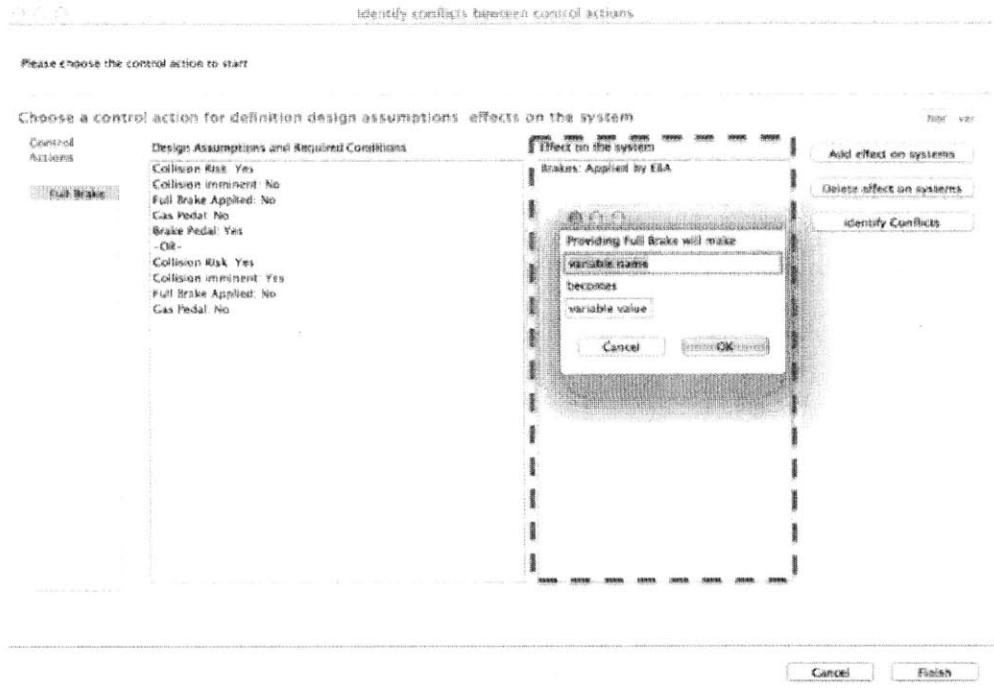
- Finish formalized STPA Step 1 for each controller individually



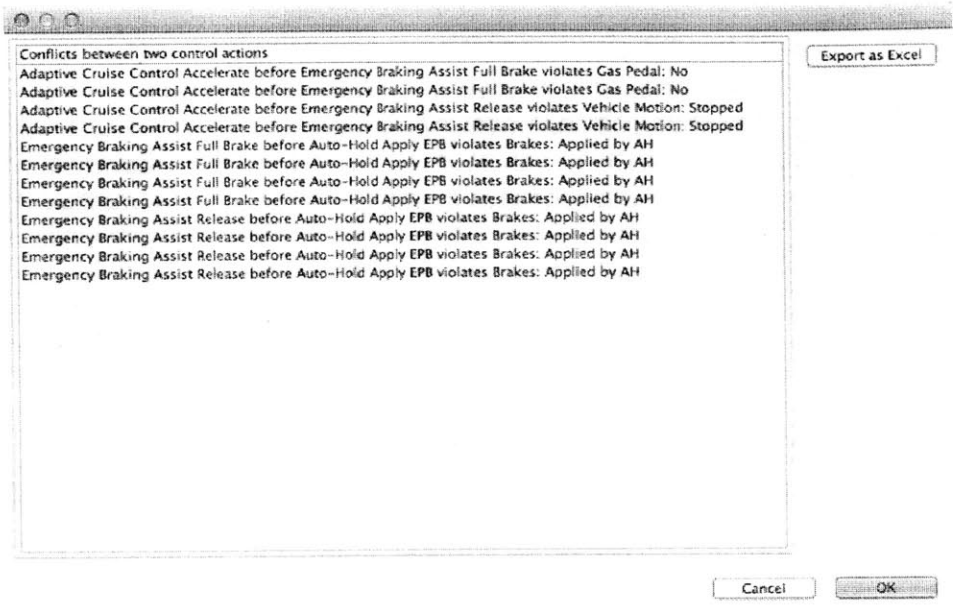
- To open the editor for integrated analysis of multiple control systems, right click on any controller in the 2-D editor->Identify conflicts



- The Design Assumptions & Required Conditions are automatically generated



- Click on the button Add effect on system to define Effects by control actions
- To identify unsafe interactions, click on the button "*Identify Conflicts*" in the previous editor



- Conflicts between controllers are automatically identified and displayed

Engineers may decide if these conflicts must be controlled in order to prevent unsafe interactions

Appendix B: Intent Specification (Keyless & EBA)

Appendix B includes the simplified formal requirements generated by the STPA Tool. Intent Specification is adopted to document system specifications for the keyless ignition and the emergency brake assist that are analyzed in Chapter 4.

Simplified context tables and SpecTRM-RL requirements generated by the STPA tool represent the black-box behaviors of keyless ignition and EBA based on Step 1 analysis. The traceability to the system hazards, the assumptions and the design constraints for the automotive system are also included. In the future, the integration of the STPA tool with SpecTRM can provide more support for requirement engineering and the documenting of system specifications based on Intent Specification.

Level 1: System Purpose

System Goals

G1: The vehicle controls should provide convenience to allow the driver to start and stop the vehicle/engine without the use of physical key fob

G2: The vehicle controls should provide brake assist to the driver to prevent front-end collision or reduce the braking distance in an event of emergency.

System Accidents

A1: Two or more vehicles collide

A2: Vehicle collides with non-fixed obstacle

A3: Vehicle crashes into terrain

A4: Vehicle occupants injured without vehicle collision

System Hazards

H1: Vehicle does not maintain safe distance from nearby vehicles

H2: Vehicle does not maintain safe distance from terrain and other obstacles

H3: Vehicle occupants exposed to harmful effects and/or health hazards

R1: Vehicle does not meet FMVSS regulations

System Design Constraints:

DC1. The vehicle controls must not hinder the driver's normal operation of the vehicle by providing conflicting commands

Level 2: System Design

Keyless Start Goals:

Keyless-G1: Keyless Ignition should allow the driver to start the vehicle/engine by pushing a button (e.g. "*Engine start/stop*") without rotating the metal key in the ignition switch

Keyless-G1.1: An Electronic Control Unit is used to monitor if the key fob is presented in the vehicle (i.e. detect electronic key code)

Keyless_G1.2: The Electronic Control Unit may disarm the immobilizer that may prevent the start of the engine unless the key code is detected

Keyless Start Assumptions:

Keyless-A1: Issuing engine stop will cause loss of propulsion

Keyless-A2: It is assumed conventional non-HEV, non-EV, without engine idle stop/start[1]

Keyless-A3: Electronic Control Unit will not issue engine start/stop commands unless the driver press the "engine start/stop" button

Keyless-A4: Brake pedal must be pressed before activating the engine

Keyless Start Design Constraints:

Keyless-R1: Power-assisted steering must remain functional after engine stop command when vehicle is moving

Keyless-R2: Power-assisted braking must remain functional after engine stop command when vehicle is not in park

---

[1] It should be noted that this assumption might be violated during feature integrations although it is true for independent analysis of the *Keyless Start* feature – i.e. engine can be started

Emergency Braking Assist Goals:

EBA-G1: The EBA should detect collision risks by monitoring stationary/moving objects ahead by using feedbacks (e.g. Range, Range Rate) from the Radar

EBA-G2: The EBA should alerts the driver by visual and audio signals when a collision risk is identified

EBA-G3: The EBA should pre-charge the braking system for full-powered brake

EBA-G4: The EBA should initiate emergency brake to prevent front-end collisions under emergencies

EBA-G4.1: The EBA should initiate full-powered brake if the brake pedal is pressed and there is a collision risk

EBA-G4.2: The EBA should initiate full-powered brake if the driver does not react to warning and a collision is imminent

Emergency Braking Assist Assumptions:

EBA-A1: The driver is not able to disable the EBA

EBA-A2: The EBA will alert the driver by audio/visual signals after it detects a collision risk

EBA-A3: The EBA is able to initiate full-powered brake under two situations:

- Driver presses the brake pedal and there is a collision risk
- **OR** Collision is imminent (without inputs from the driver)

EBA-A4: The Emergency Braking Assist will not release the brake during emergency until the vehicle comes to a full stop

EBA-A5: The EBA must continue to apply the brake after the vehicle comes to a stop after a emergency brake if no other controllers are holding the vehicle.

Emergency Braking Assist Design Constraints:

EBA-R1: The EBA must not take the control authority over the brake if there is no emergency

EBA-R2: After the EBA detects collisions risks and initiates the warning signals, it must give the driver enough time to respond (before providing full brake) as long as collisions are not imminent

EBA-R3: The EBA must not compromise the driver's ability to control the vehicle under normal condition

Le l 3: Formalized UCAs and safety requirements (SpecTRM-RL)

Key : ECU – Engine Start (Driver-initiated)

	Controller: Electronic Control Unit							
Control Action	Context ID	Driver Selection	Key Present	Trans Range	Brake Pedal	Engine Status	Not Providing causes Hazards	Providing causes Hazards
Engine Start	ECU-Start-1	No	*	*	*	*		Yes
	ECU-Start-2	Yes	No	*	*	*		Yes
	ECU-Start-3	Yes	Yes	*	*	On		Yes
	ECU-Start-4	Yes	Yes	*	No	Off		Yes
	ECU-Start-5	Yes	Yes	Not P/N	Yes	Off		Yes
	ECU-Start-6	Yes	Yes	P/N	Yes	Off	Yes	

EC tart-1: Start the engine without the driver's selection can lead to driver's confusion or panic. (↑ A4, H3)

EC tart-2: This is required by FMVSS 114 – i.e. "Each vehicle must have a starting system which, whenever the key is removed from star system prevents the normal activation of the vehicle's engine". (↑ A1, A2, R1)

EC tart-3: It could lead to loss of function only it happens multiple times. ((↑ DC1)

EC tart-4: Engine must not be activated unless the brake pedal is pressed. This UCA involves safety concerns that the engine may be unii tionally started by others (e.g. kids or animals) rather than the driver. (↑ A1, A2, H1, H2)

EC tart-5: The ECU starting the engine with transmission not being in P/N can lead to the violation of regulations ((↑ R1).

EC tart-6: The controller should follow the driver's commands to start the engine when pre-conditions are met, it is also required for ach ıg functional goals of the starting system. (↑ DC1)

Control algorithm for 'Engine Start' cmd		
Driver Selection=	Yes	T
	No	
Key Present=	Yes	T
	No	
Trans Range=	P/N	T
	!=P/N	
Engine Status=	On	
	Off	T

Keyless ECU – Engine Stop

Controller: Electronic Control Unit							
Control Action	Context ID	Driver Turns Off Engine	Vehicle Motion	Trans Range	Engine Status	Not Providing causes Hazards	Providing causes Hazards
Engine Stop	ECU-Stop-1	No	*	*	*		Yes
	ECU-Stop-2	Yes	Moving	*	*		
	ECU-Stop-3	Yes	Stopped	*	Off		
	ECU-Stop-4	Yes	Stopped	Not P	On		Yes
	ECU-Stop-5	Yes	Stopped	P	On	Yes	

ECU-Stop-1: The ECU providing *Engine Stop* without driver selection can lead to hazards (E.g. If vehicle is moving on a highway, engine being shut down automatically can lead to loss of propulsion and drivers' confusion). (↑ A1, H1)

ECU-Stop-2: Although shutting down the engine when moving at a high-speed can lead to the loss of propulsion, the vehicle should follow the driver's selection unless doing this will cause hazards or the violation of regulations. (↑ DC1)

ECU-Stop-3: It is assumed that engine has already been turned off.

ECU-Stop-4: Turn off the engine with the transmission not being P position can lead to violation of regulations if turning off engine also leads to removal of the "key code", thus violating FMVSS 114 – "The starting system must prevent key removal unless the transmission or gear selection control is locked in "park" or becomes locked in 'park' as the direct result of key removal". (↑ R1)

ECU-Stop-5: Engine control not turning off the engine after receiving driver's commands can lead to loss of fuel, potential vehicle theft, and potential exhaust and CO accumulation if parked indoors. (↑ A4, H3)

Control algorithm for 'Engine Stop' cmd		
Driver Engine Off=	Yes	T
	No	
Vehicle Motion=	Moving	
	Stopped	T
Trans Range=	!=P	
	P	T
Engine Status=	On	T
	Off	

EBA – Full Brake

Controller: Emergency Braking Assist								
Control Action	Context ID	Collision Risk	Collision imminent	Full Brake Applied	Gas Pedal	Brake Pedal	Not Providing causes Hazards	Providing causes Hazards
Full Brake	EBA-FB-1	No	*	*	*	*		Yes
	EBA-FB-2	Yes	*	*	Yes	*		
	EBA-FB-3	Yes	*	Yes	No	*		
	EBA-FB-4	Yes	No	No	No	Yes	Yes	
	EBA-FB-5	Yes	No	No	No	No		Yes
	EBA-FB-6	Yes	Yes	No	No	*	Yes	

EBA-FB-1: Full Brake should not be initiated automatically unless there is a collision risk. Doing this can surprise and confuse the driver (↑ DC1, EBA-A3, EBA-R1)

EBA-FB-2: Providing Full Brake when the driver steps on the gas pedal can make the vehicle difficult to control ((↑ DC1, ←EBA-R1)

EBA-FB-3: The Full Brake has already been applied (but not necessarily initiated by EBA).

EBA-FB-4: Whenever there is a collision risk, the EBA should alert the driver and monitor the brake pedal. If the driver press it, the EBA must initiate the Full Brake (↑ H1, H2, ←EBA-A2, EBA-A3).

EBA-FB-5: Before initiating the Full brake automatically, the vehicle should give the driver some time to respond as long as the collision is not imminent (← EBA-R2)

EBA-FB-6: If the driver does not react to the warning, the Full Brake must be applied automatically when the collision is imminent (↑ H1, H2, ←EBA-A3)

EBA – Release

Controller: EBA								
Control Action	Context ID	Vehicle Motion	Driver Present	Full Brake Applied	Vehicle Held	Gas Pedal Applied	Not Providing causes Hazards	Providing causes Hazards
Release	EBA-R-1	Moving	*	No	*	*		
	EBA-R-2	Moving	*	Yes	*	*		Yes
	EBA-R-3	Stopped	No	Yes	No	*		Yes
	EBA-R-4	Stopped	No	Yes	Yes	*		
	EBA-R-5	Stopped	No	No	*	*		
	EBA-R-6	Stopped	Yes	Yes	Yes	*	Yes	
	EBA-R-7	Stopped	Yes	Yes	No	No		Yes
	EBA-R-8	Stopped	Yes	Yes	No	Yes	Yes	
	EBA-R-9	Stopped	Yes	No	*	*		

EBA-R-1: There is no need to provide Release command if the vehicle is moving without full-powered brake applied

EBA-R-2: This context involves the condition in which emergency brake is initiated but has not been finished. Releasing the brake can cause loss of braking power. (↑ H1, H2, ←EBA-A4)

EBA-R-3: It is possible that that driver opens the door and leaves the vehicle after an emergency brake. If the EBA release the brake without the vehicle being held, the vehicle can start to rollaway. (↑ H1, H2, ←EBA-A5)

EBA-R-4: EBA releasing the brake with vehicle being held will not cause hazards (←EBA-A5)

EBA-R-5: EBA does not have the control authority to release the brake unless a full-powered brake is initiated by itself. (↑ DC1, ←EBA-R1)

EBA-R-6: After the emergency brake, the EBA should give the control authority of the brake back to the driver if the vehicle has been held (↑ DC1, ←EBA-R1)

EBA-R-7: Release the brake when the vehicle is not held can lead to unintended rollaway. (↑ H1, H2, ←EBA-A1))

EBA-R-8: This context assumes that the vehicle came to a stop due to the full-powered brake and receives inputs from the gas pedal. If the driver wants to drive, EBA should not reduce the driver's ability to control the vehicle (↑ DC1, ←EBA-R1, R3)

EBA-R-9: This is a normal condition without collision risks – i.e. the vehicle comes to a stop after driver presses the brake

Appendix C: Conflicts detected by STPA Tool

ACC providing *Accelerate* command before EBA provides *Release* command will violate required condition by EBA *Vehicle Motion: Stopped*

ACC providing *Accelerate* command before EBA provides *Full Brake* command will violate required condition by EBA *Gas Pedal: No*

ACC providing *Brake* command before EBA provides *Release* command will violate required condition by EBA *Full Brake Applied: No*

EBA providing *Full Brake* command before Auto-Hold provides *Release* command will violate required condition by Auto-Hold *Brakes: Applied by AH*

EBA providing *Full Brake* command before Auto-Hold provides *Additional Pressure* command will violate required condition by Auto-Hold *Brakes: Applied by AH*

EBA providing *Full Brake* command before Auto-Hold provides *EPB* command will violate required condition by Auto-Hold *Brakes: Applied by AH*

EBA providing *Release* command before Auto-Hold provides *EPB* command will violate required condition by Auto-Hold *Brakes: Applied by AH*

ESS providing *STOP* command before Keyless Ignition provides *STOP* command will violate required condition by Keyless Ignition *Engine: on*

ESS providing *Restart* command before Keyless Ignition provides *Start* command will violate required condition by Keyless Ignition *Engine: Off*

ACC providing *Accelerate* command before ECU provides *Engine Stop* command will violate required condition by ECU *Wheels Rotating: No*