Integration of Multiple Active Safety Systems using STPA

2015-01-0277

Published 04/14/2015

Seth Placke, John Thomas, and Dajiang Suo
MIT

CITATION: Placke, S., Thomas, J., and Suo, D., "Integration of Multiple Active Safety Systems using STPA," SAE Technical Paper 2015-01-0277, 2015, doi:10.4271/2015-01-0277.

Abstract

Automobiles are becoming ever more complex as advanced safety features are integrated into the vehicle platform. As the pace of integration and complexity of new features rises, it is becoming increasingly difficult for system engineers to assess the impact of new additions on vehicle safety and performance. In response to this challenge, a new approach for analyzing multiple control systems as an extension to the Systems Theoretic Process Analysis (STPA) framework has been developed. The new approach meets the growing need of system engineers to analyze integrated control systems, that may or may not have been developed in a coordinated manner, and assess them for safety and performance.

The new approach identifies unsafe combinations of control actions, from one or more control systems, that could lead to an accident. For example, independent controllers for Auto Hold, Engine Idle Stop, and Adaptive Cruise Control may interfere with each other in certain situations. This paper demonstrates a method to efficiently identify potential unsafe scenarios without requiring a complete enumeration or individual analysis of all possible scenarios. As a result, the approach is scalable to large systems with many controllers. In this paper, the method is demonstrated through a case study involving several driver assistance systems including advanced brake controls, advanced engine control, and advanced adaptive cruise control. Potential conflicts that would prohibit safe and successful operation are also efficiently identified, allowing engineers to develop suitable controls that prevent these conflicts.

Introduction

Commercial vehicles have recently seen a rapid introduction of new software-controlled features, from parallel parking and lane keeping to Engine Idle Stop and advanced Adaptive Cruise Control systems. As new features are added and integrated with previous systems, the overall complexity increases substantially, especially in terms of new potential interactions that weren't possible previously. This, in turn, has made it much more difficult to analyze all potential interactions or to ensure the combined systems will not behave in potentially hazardous ways. Some have turned to Use Cases as a way to examine expected situations, define the appropriate system responses, and consider potential side-effects on other vehicle systems [1]. However, Use Cases are inadequate for developing complete requirements in complex systems. Use Cases are rarely complete in the sense of capturing every potential scenario, and they are also inefficient and require individual analysis of many different scenarios to ensure that certain behaviors will never happen. In fact, accidents usually occur in off-nominal cases, i.e., conditions where the assumptions made in the Use Case are incorrect.

Traditional analysis methods, such as Failure Modes and Effects Analysis (FMEA) and Fault Tree Analysis (FTA), focus on predicted component failures. However, the increased complexity of modern systems has changed the types of accidents we see today. More and more accidents are being caused not by individual components that have failed but by software that operated exactly as required and components that have not failed [2]. In fact, most software-related accidents are caused by flawed requirements rather than coding errors or component failures [3]. To address these problems, new analysis methods are needed to identify when the required behavior is flawed and to identify potentially unsafe interactions that may result when systems are integrated. Furthermore, if the solution is to be practical for modern complex systems, it must be efficient and cannot rely on manually examining every potential interaction individually.

One way to capture feature interactions is by defining a formal executable model and assessing preconditions and postconditions for various operations [4, 5, 6, 7]. These approaches can be automated and can be much more efficient than manually enumerating each individual interaction. However, they require that a formal model of the system exists and they assume that the necessary preconditions and postconditions are already known and correct. In practice this may not be the case, especially during early development phases when most safety decisions are made [8]. In addition, interactions with other system elements can be challenging because accurate formal models of non-software components, such as humans, may never exist.

Systems Theoretic Process Analysis (STPA)

Systems Theoretic Process Analysis (STPA) [9] is a hazard analysis technique that can identify a broad array of accident scenarios including those due to component failures, dysfunctional interactions, flawed requirements, and other causes. One of the strengths of STPA is the applicability to early development phases and ability to capture interactions between many different types of components including hardware, software, human operators, human managers, and other components. More detailed comparisons and evaluations of STPA can be found in [10, 11, 12, 13]. However, very little guidance has been developed to systematically identify undesirable interactions between multiple features [14,15]. In this paper, a new process is demonstrated to systematically identify undesirable interactions between multiple features during an STPA analysis.

STPA begins by defining the system of interest and the system accidents, which are any unacceptable losses that the system must not experience. The system hazards that may lead to an accident are then identified and prioritized. A system hazard is a system state that will lead to an accident in a worst-case environment. The system being in a hazardous state does not guarantee that an accident will always occur, but the hazards should still be prevented and mitigated through the system design. The defined system accidents and system hazards provide a foundation for traceability throughout the analysis effort, so that findings and results from lower levels of analysis may be traced backed to their system-level impact.

The system is modeled in terms of a functional control structure comprised of hierarchical control loops which may have both social and technical components and can be represented at varying levels of abstraction. Figure 1 is an example of a simple, generic system control structure in which a process is controlled by automated and human controllers who act on the process through actuators and receive feedback information through sensors.

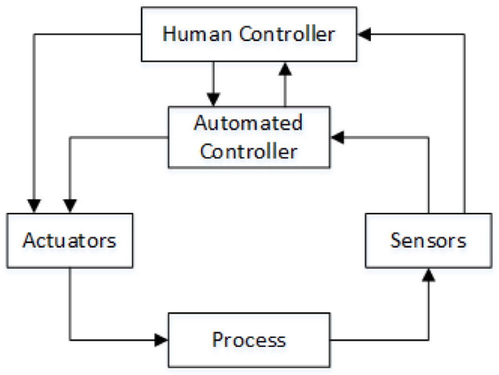

Figure 1. Generic Control Structure Example

After the functional control structure has been modeled, the first step of STPA is to identify how the controllers may issue control actions in a manner that is potentially unsafe or otherwise inappropriate. There are four ways in which a control action may be hazardous:

1). A control action required for safety is not issued.
2). A control action is issued when it is unsafe to do so.
3). A control action is issued too soon or too late.
4). A continuous control action is applied too long or stopped too soon.

Instances when these four types of Unsafe Control Action (UCA) may occur are often recorded in a table for easy reference. Once they have been identified, UCAs may be translated into safety constraints which are high level constraints on system behavior that must be enforced to ensure an accident does not occur. STPA then moves to a second step that identifies reasons that a UCA may be issued and ways in which a correct control action may be issued but not executed properly. For example, a controller may have an incorrect belief about the state of the system (i.e. an incorrect process model) that causes the controller to issue an unsafe control action. A control loop with guidewords to prompt the analysts, such as that shown in Figure 2, is used to ensure a thorough analysis [3].

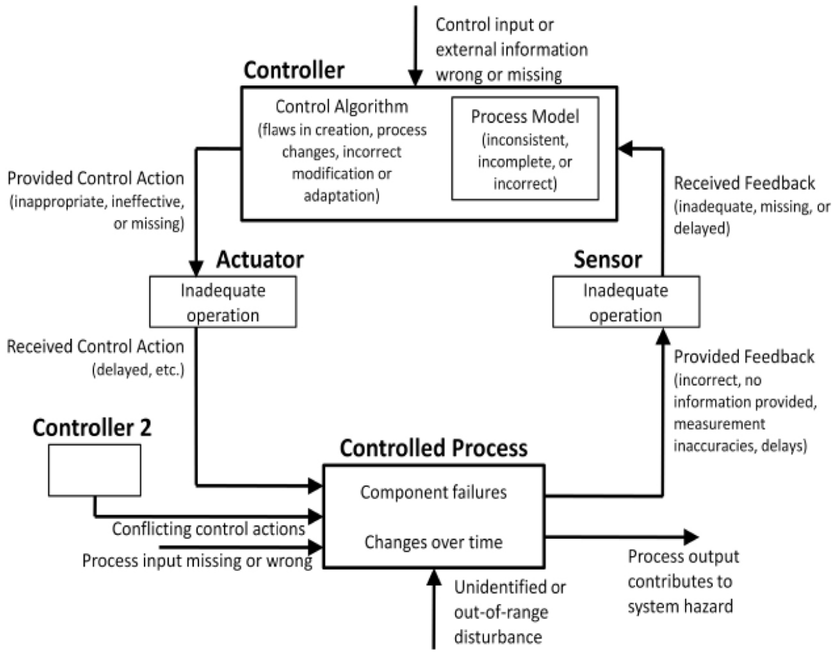
Control input or external information wrong or missing
Controller
Control Algorithm (flaws in creation, process changes, incorrect modification or adaptation)
Process Model (inconsistent, incomplete, or incorrect)
Provided Control Action (inappropriate, ineffective, or missing)
Received Feedback (inadequate, missing, or delayed)
Actuator
Inadequate operation
Sensor
Inadequate operation
Received Control Action (delayed, etc.)
Provided Feedback (incorrect, no information provided, measurement inaccuracies, delays)
Controller 2
Controlled Process
Component failures
Changes over time
Conflicting control actions
Process input missing or wrong
Unidentified or out-of-range disturbance
Process output contributes to system hazard

Figure 2. STPA Step 2 Guide Loop [3]

Once identified, the causal factors leading to accidents may be used to write requirements for the system components. The results of the causal factor analysis may be used to eliminate hazardous scenarios or, if elimination is not possible, reduce their occurrence or their impact.

STPA is a top-down methodology capable of analyzing complex systems and capturing interactive effects between subsystems. Significant guidance is provided throughout the process to ensure the system in question is analyzed in a methodical manner. Because the method can be used at many levels of abstraction, STPA is useful beginning at the concept development stage and through the entire design process.

In this paper, additional guidance is developed to efficiently identify interactions between multiple controllers during STPA Step 1. Note that STPA Step 2 must still be performed to identify causes of unsafe control. The new process is demonstrated and evaluated using a case study that integrates three automotive features.

Case Study

The case study for this research involves three automotive features. Ideally, one would start with the highest level of system goals that, through decomposition, would eventually branch into features that can be developed by different engineering teams. However, it is often the case in industry that an engineer (or engineering team) is responsible for integrating legacy systems that were designed separately and it may not always be possible to start with a blank sheet of paper or develop the integrated system from scratch. The approach described in this paper can be applied in either case, however the case study below describes the more challenging case of three independently developed legacy systems that must be integrated safely.

Feature Descriptions

The three features in this case study are Auto-Hold, Engine Stop-Start, and Adaptive Cruise Control with Stop-Go. These features share a similar concept; an embedded controller is integrated into the vehicle to control existing hardware in the braking and propulsion systems in order to provide new and/or enhanced functionality. Due to proprietary considerations, the descriptions used in this paper are fictional and do not necessarily reflect the production intent of any manufacturers. However, these examples were developed in collaboration with professional automotive engineers to ensure that they are representative of what three independent features might look like before integration efforts and realistic in terms of the potential for undesirable interactions. Publicly available information from a number of manufacturers was also surveyed to ensure the examples are representative in terms of the types of features being introduced throughout the industry and their basic functionality.

Auto-Hold

Auto-Hold (AH) is an automatic braking feature that holds the vehicle and prevents rollback, allowing the driver to remove his foot from the brake pedal without vehicle movement whether it is on an incline or not. When the vehicle is brought to rest using the brakes, the AH feature will maintain the necessary brake pressure to keep the vehicle from moving by capturing the pressure already in the system. Once sufficient wheel torque (other than idle torque) is supplied to move the vehicle forward, the feature will disengage and release the braking system back to its normal state. The AH feature uses existing hardware components including Anti-Lock Braking System (ABS) and Electronic Parking Brake (EPB) components. Auto-Hold may issue the following four high-level commands:

- **HOLD:** When AH is ENABLED and the vehicle is brought to rest using the brake pedal, the HOLD command is issued to capture the existing brake pressure and place the feature in HOLD-MODE. AH identifies this situation by monitoring brake-pressure and wheel speed as provided by the braking system.
- **ADDITIONAL PRESSURE:** When the system is in HOLD-MODE and the wheels begin to rotate, the ADDITIONAL-PRESSURE command is issued to increase brake pressure (using the ABS pump) until the vehicle comes to rest.
- **RELEASE:** When the system is in HOLD-MODE and one of two conditions is met, RELEASE is issued to release the valve and return the brake system to normal operation. These two conditions are: 1) The propulsion torque is sufficient to move the vehicle. 2) Another system takes responsibility for holding the vehicle.
- **APPLY EPB:** When the AH system is in HOLD-MODE, it may engage the EPB. This control action may be issued if the hydraulic brakes are not effective.

Engine Stop-Start

Engine Stop-Start (SS) is a feature designed to reduce fuel consumption and emissions by turning off an engine that would otherwise be idling while the vehicle is stopped. When the vehicle comes to a complete stop, the engine is automatically turned off and then restarted before motion resumes. The two high-level commands are:

- **STOP:** Once the vehicle is brought to rest using the brake pedal, *Auto-Stop* is issued which shuts down the engine.
- **RESTART:** When the system is in AUTO-STOPPED and power needs arise, RESTART is issued to restart the engine.

Adaptive Cruise Control with Stop-Go

Adaptive Cruise Control with StopGo (SG) is an enhanced version of the legacy cruise control feature. With traditional cruise control, a driver may pre-set a speed for the vehicle to maintain, allowing him to remove his foot from the accelerator pedal. Traditional cruise-control systems allow the driver to set the speed, increment the set speed up or down, temporarily increase speed using the accelerator pedal (such as when passing), and disengage the feature using the driver controls or the brake pedal.

Adaptive Cruise Control (ACC) builds upon this traditional architecture in that it intelligently considers the distance to vehicles and other objects ahead of the primary vehicle in the same lane. Radar is mounted in the front of the vehicle that reports the range and range-rate to a controller that may utilize the braking and propulsion systems to maintain a safe trailing distance.[1] As with speed, the driver can change the desired trailing gap through the feature controls.

1. Trailing gap may be implemented as a time-gap to the leading vehicle so that it can vary dynamically with speed.

The Stop-Go capability further builds upon this architecture, enabling the ACC system to bring the vehicle to a full stop and resume motion when following a target vehicle. This is intended to be used in stop-and-go traffic when the vehicle may momentarily come to a stop before moving forward in concert with the surrounding vehicles.

- **ACCELERATE:** When the system is enabled, it may provide the accelerate command to accelerate the vehicle using the propulsion system.
- **DECELERATE:** When the system is enabled, the decelerate command will decelerate the vehicle using the braking system.

Potential for Unsafe Interaction

As mentioned previously, all three of these features have some control authority over the brake and/or propulsion subsystems and thus have the potential to interact through the vehicle dynamics. It is possible that the features may interact positively, for example if Auto-Hold maintains the vehicle's rest position thereby enabling the Engine Stop Start feature to halt fuel consumption while the vehicle is held stationary. However, it is also possible for the features to interact negatively, for example if the Auto-Hold feature prevents the ACC w/Stop-Go feature from taking off after a brief pause in traffic or if the Engine Stop-Start feature shuts off the engine while ACC w/ Stop-Go is attempting to accelerate or resume.

Identifying potential instances of these types of negative interactions, both inconveniences and potential hazards, is the goal of this case study and the techniques demonstrated. We will show that STPA and the new technique can identify instances of potentially dysfunctional and/or hazardous interaction at the beginning of the engineering design process, before significant development work has occurred, and prevent significant rework during verification and validation testing.

Analysis

System Accidents and System Hazards

Tables 1 and 2 present the sets of vehicle-level system accidents and system hazards that were defined at the beginning of the analysis. These system accidents represent the losses that should be avoided during operation and the system hazards represent the system states that might lead to such losses.

Table 1. System Accidents

Number	System Accident Description
A-1	Two or more vehicles collide
A-2	Vehicle collides with other obstacle or terrain
A-3	Vehicle occupants injured without vehicle collision

For example, A-1 could occur if a trailing vehicle rear-ends a leading vehicle in city traffic. A-2 could occur if a vehicle collides with a pedestrian or a guardrail. A-3 could occur if there is excessive deceleration resulting in whiplash.

Table 2. System Hazards

Number	System Hazard Description	Accident
H-1	Vehicle does not maintain safe distance from nearby vehicles	A-1
H-2	Vehicle does not maintain safe distance from terrain and other obstacles	A-2
H-3	Vehicle enters uncontrollable/unrecoverable state	A-1, A-2, A-3
H-4	Vehicle occupants exposed to harmful effects and/or health hazards	A-3

For example, H-1 could occur if a vehicle accelerates into another vehicle ahead. H-2 could occur if a vehicle experiences a near miss with a pedestrian or other object. H-3 could occur if a vehicle accelerates too fast for weather conditions. H-4 could occur if there is excessive temperature such as occupant heat exhaustion or burns from hot surfaces.

System Control Structure

Following the definition of system accidents and system hazards, the system control structure was developed as shown in Figure 3.

The system control structure is a functional representation of the system as hierarchical control loops. The control structure consists of functional blocks connected by arrows that represent control actions and feedback. When drawing the control structure, the analyst assigns responsibilities to each functional block and connects them in a hierarchy. Defining a global control structure helps ensure that all system stakeholders have a common understanding of the system's design and operation.

The blocks in Figure 3 represent functional entities in the system. The convention used here is that entities higher on the structure have authority over entities lower on the structure and that control actions flow down while feedback flows up. For visual simplicity, only one arrow in a given direction is shown between two blocks-so an arrow represents a path rather than an individual command or piece of feedback. Multiple terms labeling an arrow represents the set of control actions or feedback variables (depending on direction) that may exist on the path indicated by the arrow.

The control structure shows that the three feature systems and the driver have overlapping control authority and receive feedback from common processes. This overlap and commonality creates the potential for interaction that may lead to unsafe system behavior.

Identifying Unsafe Control Actions

Figure 3 shows that each of the controllers in the system may issue commands, also referred to as control actions. As discussed previously, there are four ways in which the issuing of a control actions may be unsafe. These four manners of unsafe control must be considered for each control action issued in the system. In this system, the automated controllers issue a total of nine high-level control actions. For space considerations, analysis of only one of these control actions will be shown to demonstrate the process applied to the entire system.

When engaged, the Auto-Hold feature may issue the RELEASE command to return the braking system to normal operation. The circumstances in which it may be unsafe to issue the RELEASE command are listed in Table 3.

The UCA's listed in Table 3 may be further refined and formalized as shown in the Context Table presented as Table 4. This method of formalizing UCA's was developed and first presented by Thomas in [16]. The Thomas method rests on the premise that UCA can be formally defined by a set of Process Model Variables that define the state of the system as viewed by the controller. In other words, the Process Model Variables are used by controllers to determine the appropriate control actions. Each row in the table corresponds to a UCA and asterisks are used to denote which variables do not matter for a given row. In addition to providing increased formalism and clarity, the Thomas method enables automated identification of conflicts and the generation of executable requirements.

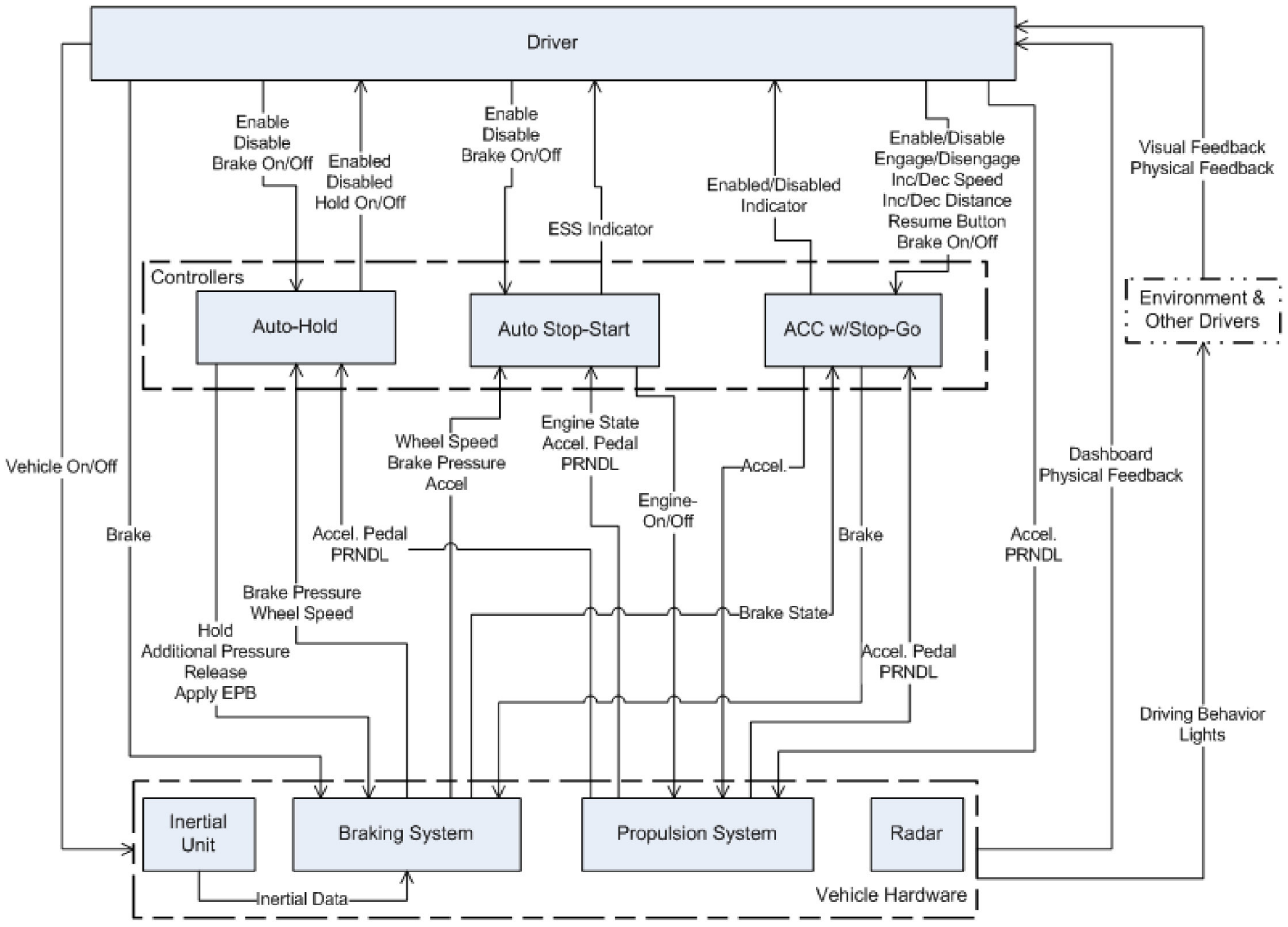

Figure 3. Combined System Control Structure

Table 3. UCA Table for the Auto-Hold Command: RELEASE

Controller: Auto-Hold Module				
Control Action	**Not Provided**	**Provided**	**Too Soon/Late**	**Too Long/Short**
RELEASE	UCA-AH-11: Not providing RELEASE is hazardous if driver has commanded sufficient wheel torque via the accelerator pedal [H-1,2,3]	UCA-AH-12: Providing RELEASE is hazardous if AH is in HOLD-MODE and driver has not commanded sufficient wheel torque [H-1,3]	UCA-AH-13: Providing RELEASE before the there is sufficient wheel torque is hazardous [H-1,3]	
	UCA-AH-14: Not providing RELEASE is hazardous if the driver DISABLES AH [H-1,3]			

Table 4. Context Table for the Auto-Hold Command: RELEASE

Context ID	AH Enabled	Hold-Mode	Driver Present	PRNDL	Gas Pedal	Propulsion Torque Sufficient	Brake Pedal	EPB On	Providing Causes Hazard	Not Providing Causes Hazard	Providing Required for Function	Not Providing Required for Function
AH-R-1	Yes	Yes	Yes	P	*	*	*	No		x		
AH-R-2	Yes	Yes	Yes	N	*	*	Not Pressed	No	x			
AH-R-3	Yes	Yes	Yes	N	*	*	Pressed	No	x			
AH-R-4	Yes	Yes	Yes	R,D,L	Pressed	Yes	*	No		x		x
AH-R-5	Yes	Yes	Yes	R,D,L	Pressed	No	*	No	x			
AH-R-6	Yes	Yes	Yes	R,D,L	Not Pressed	*	*	No	x			
AH-R-7	Yes	Yes	No	*	*	*	*	No	x			
AH-R-8	No	Yes	*	*	*	*	*	No	x			
AH-R-9	*	No	*	*	*	*	*	No				
AH-R-10	*	*	*	*	*	*	*	Yes				

Identifying Conflicts between Controllers

One way that conflicts occur is when the effects of one command violate the assumptions and conditions of another. As a consequence, one or more controllers may be prevented from realizing their intended functions resulting in potential performance and safety issues.

Another way that conflicts occur is when the effects of one command satisfy the assumptions and conditions of another, triggering the second command to be issued at an unanticipated time or in an uncoordinated manner. This type of conflict is more subtle than the first that arises from commands effectively prohibiting each other. In this case, the requirements for an individual command to be safe and functional are still met; however, from a system perspective, the control amongst controllers is uncoordinated such that they may be competing against each other.

Creating a Conditions Table

To identify potentially unsafe interactions, a conditions table like the one in Table 5 is used. The conditions table is a table that keeps a record of the design assumptions, required conditions, and effect on the system associated with each command.

Table 5. Conditions Table Format

	Controller 1		Controller 2	Controller 3	
	Command A	Command B	Command C	Command D	Command E
Design Assumptions & Required Conditions					
Effect on the System					

The fields in the conditions table can be partially populated with the results of the UCA analysis presented earlier. The formal analysis of UCA's using Context Tables produces the "Required Conditions" input for the conditions table. The "Design Assumptions" may also arise during the UCA analysis or can be elicited from the engineering design team. The "Effect on the System" can be populated by considering the process model variables identified previously in STPA.

As an example, the combination of rows in Table 4 that are not marked as 'Providing Causes Hazard' but are marked as 'Not Providing Causes Hazard' or 'Not Providing Required for Function' can be used to help define the required conditions for the AH RELEASE control action. These are further refined by the design assumption that RELEASE will be issued when AH is holding the brakes (as defined in the context table) to produce the results in Table 6.

Table 6. Auto-Hold portion of the Conditions Table

		Design Assumption & Conditions Required	**Effect on the System**
Auto-Hold	**HOLD (HOLD-MODE)**	**AH Enabled:** Yes **Wheels Rotating:** No **Brakes:** On **Driver Present:** Yes **Gas Pedal:** No **Range:** D | L	**Brakes:** Applied by AH
	RELEASE	**Brakes:** Applied by AH **Propulsion Torque:** Yes **AND Driver Present:** Yes **OR Vehicle Held:** Yes (i.e. **Range:** Park | **EPB:** Yes) **Range:** D | P	**Brakes:** Not Applied by AH
	ADDITIONAL-PRESSURE	**Brakes:** Applied by AH **Wheels Rotating:** Yes **Electrical Power:** Available (i.e. **Engine:** On | **Battery:** High) **Brake Pressure:** <Max	**Battery:** Reduce Charge **Brake Force:** Increased

Table 7. Engine Stop-Start portion of the Conditions Table

		Design Assumption & Conditions Required	Effect on the System
Engine Stop-Start	**STOP (AUTO-STOPPED)**	**ESS Enabled:** Yes **AUTO-STOPPED:** Yes **Vehicle Held:** Yes (i.e. **Brake:** On | **EPB:** Yes) **Restart Possible:** Yes (i.e. **Battery Charge:** High) **Driver Present:** Yes **Gas Pedal:** No **Auxiliary Power Needs:** Low **Range:** !=P,R,N	**Propulsion:** Off **Idle Torque:** No **Electrical Power**: Off **AUTO-STOPPED:** Yes
	RESTART	**Vehicle Held:** Yes (i.e. **Brakes:** On | **Range:** Park | **EPB:** Yes) **Wheels Rotating:** No **Restart Possible:** Yes (i.e. **Battery Charge:** High) **Driver Present:** Yes **Range:**!=P,R,N	**Propulsion:** On **Idle Torque:** Yes **Electrical Power:** On - power reduced ~2s **AUTO-STOPPED:** No

Table 8. Adaptive Cruise Control with Stop & Go portion of the Conditions Table

		Design Assumption & Conditions Required	Effect on the System
Adaptive Cruise Control with Stop-Go	**Accelerate**	**ACC Enabled & Engaged:** Yes **Vehicle Held:** No (i.e. **Brakes:** Not Applied **& EPB:** No) **Driver Present:** Yes **Driver Gas Pedal:** No **Range:** D *If* **Target Locked:** Yes *Then* **Distance >= Threshold:** Yes **Speed <= Threshold:** Yes **Propulsion Power:** Available (i.e. **Engine:** On)	**Gas:** Applied by ACC **Speed:** Increased **Wheels Rotating:** Yes
	Brake	**ACC Enabled & Engaged:** Yes **Driver Pedal Input:** No **Gas:** Off **Driver Present:** Yes **Range:** D **Hydraulic Power:** Available (i.e. **Engine:** On | **Battery:** High)	**Brakes:** Applied by ACC **Speed:** Decreased

The Condition Tables for Auto-Hold, Engine Stop-Start, and Adaptive Cruise Control with Stop & Go are presented below. The generic table presented as Table 5 has been inverted and broken into three sections (one for each feature controller) for readability. Appending tables 6, 7, 8 and inverting the axes will put the information in the same format as presented in Table 5.

Searching for Conflicts

The conditions table can be reviewed and searched for instances when the effects of one control action conflict with the assumptions and required conditions of another. The conditions table grows on the order $O(n)$ where n is the number of control actions that must be analyzed for interactions and conflicts. In other words the table grows linearly for each new control action that must be analyzed. A brute-force approach such as enumerating Use Cases can be thought of as analyzing and populating an n × n matrix (where n is the number of control actions), which exponentially grows on the order $O(n^2)$ by contrast and only captures pairwise interactions. In contrast, a conditions table like Table 5 is much more practical for large systems. Although the number of conflicts may inevitably grow as more control actions are included, a tool can also be used to automatically search a conditions table like Table 5 thereby ensuring that the manual effort by the user remains on the order $O(n)$. STPA Step 2 can then be performed on the resulting list of conflicting control actions to identify potential causes.

Results

By following this method, more than 60 conflicts were identified allowing potential controls to be developed. These conflicts are the result of design or requirements flaws, and many of them are not immediately obvious without a systematic technique to identify them. Many also involve assumptions that may have seemed reasonable for individual features but are not appropriate for an integrated system. Although only two scenarios are discussed here due to space limitations, a more complete demonstration is available in [13].

Conflict Scenario 1

The first example result is a conflict between Auto-Hold and Adaptive Cruise Control w/ Stop-Go. When both features are installed on a vehicle, they can conflict due to operation of the EPB. The conflict was identified as:

Conflict 16: AH APPLY EPB prior to ACC w/SG ACCELERATE violates the constraint *Vehicle Held: No*

When both AH and SG are installed and enabled in a vehicle, the following scenario is possible that would lead to a dysfunctional and potentially hazardous situation:

- Driver engages ACC w/SG to maintain a selected speed and safe following distance.
- Traffic slows to a stop; SG slows the vehicle and holds it at rest.
- Once at a stop, AH engages (because the brakes were applied) and captures the existing pressure in the brake lines.
- Some stimulus (see below) triggers AH to engage the EPB.

Outcome: ACC w/SG cannot ACCELERATE because the vehicle is held by the EPB…

In this scenario, EPB is not released. Although AH can apply EPB, it does not have the ability to release EPB as a safety precaution (see Figure 3). AH issuing the APPLY EPB control action will change the 'Vehicle Held' state to 'Yes.' This violates the ACC w/SG requirement that 'Vehicle Held' be 'No.' If the vehicle is being held, the engine torque from the ACCELERATE command (e.g. when traffic clears) will oppose the force holding the vehicle which may prevent or delay forward motion or potentially damage systems like the parking brake. Recall that as a safety precaution the SG system does not have authority to disengage the EPB (see Figure 3).

There are several reasons that the AH feature may engage the EPB, one being the driver disabling the AH feature. Other potential reasons can be identified in STPA Step 2.

At first glance, this conflict appears to lie outside the safety domain; however, it may have safety implications when the driver needs to quickly move the vehicle (i.e. it is stopped in an intersection) but is unable to do so because he or she must first recognize the EPB is applied, and then disengage and take control from SG.

This conflict may be resolved; three potential strategies are described below:

1). Require that ACC w/SG monitor the state of the EPB and allow it to disengage when appropriate. Allowing ACC w/SG to disengage the EPB resolves the conflict, but adds complexity in that it must now be decided when it is appropriate for an automated system (ACC w/SG) to disengage the EPB.

2). Require that an issuing of the EPB turns other features 'off' and requires Driver intervention to disengage the EPB. This change is simpler than the first potential resolution and does not require giving automation the authority to disengage the EPB. However, not giving automated systems the ability to disengage the EPB means that each time it is issued, the Driver will be required to intervene. If the driver is required to intervene often, this strategy may reduce the transparency with which some features operate and thus reduce their value.

3). Require that AH does not engage when ACC w/SG is engaged because AH becomes redundant when ACC w/SG is holding the vehicle still. Note, this change does not prevent AH and ACC w/SG from using the same strategy and physical hardware to control the vehicle's brakes. A design solution is that ACC w/SG effectively 'implements' AH when it brings the vehicle to a stop; however, this should be done within the ACC w/SG logic and the standalone AH system should remain disengaged.

As this conflict exists at a fairly high-level of abstraction and concerns the overlap of authority between Auto-Hold and Adaptive Cruise Control w/Stop-Go, the third potential resolution is perhaps the best choice.

Conflict Scenario 2

The first conflict involved a pair of controllers. This second conflict scenario is an example of conflict between three controllers: Auto-Hold, Engine Stop-Start, and the Driver. The following conflicts were detected using the new approach:

Conflict 21: AH RELEASE while engine AUTO-STOPPED violates the AH required condition *Propulsion Torque: Yes*

Conflict 51: Driver SHIFT prior to ESS RESTART may violate the constraint that the Range is in a forward gear

Again consider a vehicle that has both AH and ESS enabled:

- Vehicle comes to a stop, both the AH and ESS features engage successfully.
- Driver attempts to move the vehicle backward:
  - Driver shifts to Reverse
  - Driver applies the Accelerator Pedal

Outcome: The vehicle is effectively stuck, because:

- ESS is prevented from restarting the engine by FMVSS 102 [17].
- AH cannot RELEASE because there is insufficient wheel torque.

The stopping and starting of a vehicle engine is partially regulated by Federal Motor Vehicle Safety Standard (FMVSS) 102. The older version of this standard prohibited the engine starter from operating while the transmission shift lever is in either the forward or reverse drive position. This proves to be a barrier for new technologies, including hybrid-electric vehicles and idle-stop systems. In response to developments in those technologies, an updated version of FMVSS 102 prohibits the engine from automatically stopping in reverse gear but allows it to restart if automatically stopped while in a forward gear (i.e. followed the driver shifting to reverse) [17]. The engine may not automatically restart in reverse when the service brake pedal is not applied and must restart automatically when it is applied. This means that in the scenario described above, the engine may not automatically restart until the driver applies the service brake: AH maintaining brake pressure is not sufficient.

With the engine off, the driver pressing the gas pedal does not produce any propulsion torque and AH may not issue the RELEASE command. Thus, the vehicle is effectively stuck until the driver moves his foot to the service brake and the engine restarts. Once this happens, restored engine power will produce engine torque and if the driver presses the gas pedal, AH will issue RELEASE. However, this sequence may take several seconds during which the vehicle is effectively stuck and the sequence of driver actions required for recovery may not be obvious.

Resolving this conflict is not as straightforward as the first example as it involves several layers of control logic and regulation. Design engineers must prioritize the hazards associated with various solutions and choose one that is acceptable to all stakeholders.

Summary/Conclusions

This paper introduced a new method to identify potentially hazardous interactions among software-intensive features during STPA Step 1. The method was demonstrated by applying it to a case study with three independently developed features that were to be integrated. A number of potentially hazardous interactions were systematically identified, including interactions caused by potentially flawed requirements. More important, although a large number of potential interactions were possible, the method was found to be scalable and did not require enumeration of all possible interactions. Instead, hazardous interactions and conflicts were efficiently identified using much smaller condition tables.

Although a number of formal methods utilize pre-and post-conditions to search for undesirable feature interactions, these methods require a formal model of the system and assume that the required conditions are already known or given. The proposed method was able to derive the necessary conditions systematically from the context tables of an STPA analysis. In addition, the method did not require a formal model of the system to begin the analysis. This is important because formal models typically do not exist during early system development and may never exist for non-software components such as humans. As a consequence, this approach was successfully applied beyond software components to identify dangerous human interactions with automated systems such as driver shift commands before an ESS restart.

Future work includes a case study of greater complexity with a set of real vehicle systems. An example with controllers at multiple levels in the vehicle control hierarchy should also be considered to accurately represent modern production vehicles.

The method in this paper has the potential to be partially automated and tools can be developed to search for these types of conflicts. An open-source software tool is currently being developed to partially automate the STPA process, especially the identification of unsafe control actions in STPA Step 1 [18]. Once potential unsafe actions are identified, safety requirements can be generated or existing requirements can be checked to verify that unsafe behaviors are prevented.

References

1. Alladi, V., Wei, J., and Ganesan, S., “Writing Better Real-Time System Requirements with Use Cases and Services,” SAE Technical Paper 2005-01-1315, 2005, doi:10.4271/2005-01-1315.
2. Leveson N., Applying Systems Thinking to Analyze and Learn from Events, *Safety Science* 49(1):55-64, 2010.
3. Leveson N., “A New Accident Model for Engineering Safer Systems,” *Safety Science* 42(4):237-270, 2004.
4. Tsui, F., Karam, O., Bernal, B., “Essentials of Software Engineering,” (Jones & Bartlett Learning LLC, 2014).
5. Hayes, I. J.. “VDM and Z: a comparative case study.” *Formal Aspects of Computing* 4(1):76-99, 1992.
6. Zave P.. “A practical comparison of Alloy and Spin.” *Formal Aspects of Computing* 2014. Available at Springer via http://dx.doi.org/10.1007/s00165-014-0302-2
7. Leavens Gary T. and Baker Albert L., “Enhancing the pre-and postcondition technique for more expressive specifications.” FM'99 - Formal Methods. Springer Berlin Heidelber, 1999. 1087-1106.
8. Frola, F. R., Miller, C. O., “System safety in aircraft acquisition.” Logistics Management Institute. Bethesda, MD. 1984.
9. Leveson, N., “Engineering a Safer World,” MIT Press, 2012.
10. Balgos V. H., “A systems theoretic application to design for the safety of medical diagnostic devices,” Master's thesis, MIT, Cambridge, 2012.
11. Torok, R., Geddes, B., “Systems Theoretic Process Analysis(STPA) Applied to a Nuclear Power Plant Control System,” Presentation at MIT STAMP Workshop, March 2013.
12. Leveson, N., Wilkinson, C., Fleming, C., Thomas, J., Tracy, I., “A Comparison of STPA and the ARP 4761 Safety Assessment Process,” MIT PSAS Technical Report, 2014.
13. Placke, S., “Application of STPA to the Integration of Multiple Control Systems: A Case Study and New Approach,” Master's thesis, Engineering Systems Division, Massachusetts Institute of Technology, 2014.
14. Ishimatsu, T., Leveson, N., Fleming, C., Katahira, M., Miyamoto, Y., and Nakao, H., “Multiple Controller Contributions to Hazards,” presented at the 5th IAASS Conference, Versailles, France, October 2011.
15. Ishimatsu T., Leveson N., Thomas J., Fleming C., Katahira M., Miyamoto Y., Ujiie R., Nakao H. and Hoshino N., “Hazard Analysis of Complex Spacecraft using Systems-Theoretic Process Analysis,” *Journal of Spacecraft and Rockets* (51)2:509-522, 2014.

16. Thomas J., "Extending and Automating a Systems-Theoretic Hazard Analysis for Requirements Generation and Analysis," Ph.D. Dissertation, Engineering Systems Division, Massachusetts Institute of Technology, 2013.
17. "Federal Motor Vehicle Safety Standards; Transmission Shift Position Sequence, Starter Interlock, and Transmission Braking Effect," 49 CFR Part 571, 2005.
18. Thomas, J. and Suo, D. "An STPA Tool." Presented at 3rd STAMP/STPA Conference., MIT, Cambridge, MA, 2014.

Definitions/Abbreviations

ACC - Adaptive Cruise Control

AH - Auto Hold

EPB - Electronic Parking Brake

ESS - Engine Stop Start

FMEA - Failure Modes and Effect Analysis

FTA - Fault Tree Analysis

SG - Stop-Go

STPA - System Theoretic Process Analysis

ISSN 0148-7191

http://papers.sae.org/2015-01-0277