Deriving Functional Safety Requirements using Undesired Combination State Templates

Daniel Aceituna
DISTek Integration
Fargo, North Dakota, USA
Daniel.aceituna@distek.com

Kaushik Madala, Hyunsook Do
Department of Computer Science and Engineering
University of North Texas
Denton, Texas, USA
kaushikmadala@my.unt.edu, hyunsook.do@unt.edu

***Abstract*— Functional Safety (FS) has become a major consideration during the development of embedded and reactive systems. When incorporating functional safety into a system under development, the engineer must be cognitive of not just system requirements, but safety requirements as well. These safety requirements are typically derived from an initial set of hazard and risk assessments. While there are various means of performing these assessments, there is a particular classification of hazards, based on undesired combination states that may be overlooked by other techniques. In this paper, we propose a technique that addresses these hazards, using Undesired Combination States Templates (UDCST). The technique has been conceived to accommodate the constraints on time and cost typically encountered in industry. We demonstrate the technique with two real world applications.**

***Index Terms*—Functional Safety, safety requirements, hazard assessment.**

I. Introduction

With the ever-increasing presence of reactive systems, both embedded and non-embedded, the degree of human interaction with decision-making systems has prompted a greater need for safety considerations. In particular, systems must be designed to detect, prevent, and/or recover from a potentially hazardous state. Unfortunately, more often than not, the stakeholders specifying a reactive system tend to focus on the expected and intended behaviors of the system, and overlook contingencies that would address undesired scenarios. Consequently, additional time and expense is often spent addressing these safety issues during and after the system's implementation phase. To help raise safety awareness and guide stakeholders and engineers in addressing safety considerations, various industries have incorporated the practice known as Functional Safety (FS). For our purposes, we will view functional safety as the practice of assessing a system's potential hazards and reducing those hazards to an acceptable risk by developing safety functions that would detect the presence of those assessed hazards and place the system into a safe state [1]. Various ISO standards and other documents define how the practice of functional safety should be performed such as ISO 26262 [1] for the auto industry, ARP 4761 [2] for aerospace, and the IEC 61508 [3] for electronics industry in general. Whichever the industry, there are some common FS practices that are applied to the system being safe-guarded, in particular, the practice of assessing potential hazardous situations or scenarios, sometimes known as functional hazard assessment (FHA). These hazard scenarios are then used to determine the safety requirements/functions needed to reduce the hazards to acceptable levels. Various FHA techniques are practiced in industry to assess hazard scenarios, more commonly fault tree analysis (FTA) [4], and failure modes and effects analysis (FMEA) [5].

Both approaches focus on hazards that be potentially caused by single-events that are often tied to the state of one component (FMEA) or one event (FTA). However, there is a certain class of potential hazards that are not caused by a single (state) event, but rather they are caused by a combination of system component states, as illustrated by the Swiss cheese model [6], which describes the hazardous cumulative effect of more than one event state.

In this paper, we propose a means of discovering hazards that can occur via a combination of system component (and environmental) states, while minimizing the number of state permutations that can arise from a combinatorial approach. The minimizing of permutations is meant to make the approach less time consuming, easier to enact, and thus, more industry friendly. In our approach, we use the concept of the Un-Desired Combination State (UDCS). UDCSs are then expressed within a set of UDCS Templates (UDCST) that translates the UDCSs into safety rules that can in turn be used throughout the functional safety life cycle. UDCSs are defined using a notation scheme based on two concepts: sets and "component(state)" labeling.

A UDCS uses a pair-wise combination of states. We focus on the combination of two systems component states at a time because it requires at least two component states to constitute a combinational-induced hazard. Once a two combination hazard is identified, it is ultimately up to the engineer and stakeholders to identify if one or more additional concurrent component states will indeed nullify the hazard arising from only two. Furthermore, in theory, there is nothing restricting a UDCST user from extending the combination to three or more.

978-1-5386-8410-8/18/$31.00 ©2018 IEEE
DOI 10.1109/RESACS.2018.00006

There are times where the combination of two states does not result in an immediate hazard, but rather the hazard could arise after the pair of states has been concurrent for a specific amount time. Thus, we allow for an optional third state of "elapsed time" and allow for temporal considerations that may be a decision factor on when a hazard can eventually occur. We also manage a potential combinatorial state explosion by focusing on the system component states that are highly hazard prone.

Once a set of UDCSs are derived, they are each formatted into safety rules using UDCS templates, which in turn can express a given UDCS in terms of its reachability (in order to prevent the UDCS from occurring) or its recovery (in order to recover from the UDCS). Figure 1 shows how the UDCS templates are used within the specification, design, and implementation of a reactive system. Undesired states are assessed by engineers and stakeholders [A], and the assessed states are translated into safety rules via the templates [B], from which safety requirements [C], can be elicited for the safety protection system [D]. Note that our focus is in identifying safety requirements [B], not on the implementation of a safety function. Also note that the UDCS templates [B] are used as a supplement to the hazard assessment process. Our approach is not meant to substitute techniques such as FTA or FMEA [E], but rather the UDCS templates are meant to address hazards that can arise from state combinations. Our technique can also be implemented fairly quickly, allowing for its acceptance into an industry setting.

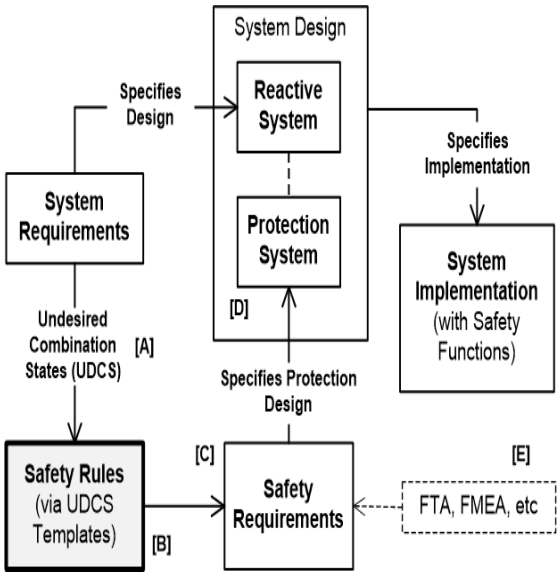

Fig.1. Overview of the proposed approach to deriving safety requirements.

The rest of the paper will address the conceptual background of UDCS Templates (Section II), steps in proposed approach (Section III), two small real world applications (Section IV), a discussion on the applications, and observed limitations (Section V), related work (Section VI), and the conclusion (Section VII), where we summarize the contributions and future direction of this research.

II. Preliminary Background of UDCST

We mentioned earlier the two concepts of preventing a UDCS from being reached and the recovery from a UDCS that has occurred. We now give the theoretical basis for these two concepts beginning with the idea that a hazardous situation can be described as a combination of two system component/environmental states that each by itself may not constitute a hazard. Figure 2 is a conceptual representation of any system consisting of N number of system states. In this case N is 4 system states, where every system state can potentially transition to every other system state (here each system state consists of two concurrent component states). Figure 2 is a worst case example of bidirectional adjacency between N system states, where the number (#) of transitions is a function of the number of states N, as defined by #MaxTransitions = 2(N(N – 1)/2). In reality, the number of transitions is typically less than #MaxTransitions. The transitions to our conceptual system have been labelled E/T to signify when a transition is caused by an event in the environment (E) or the transition is caused automatically, by the system, after a certain time interval (T). We define a system behaviour as a sequence of transitions between n number of system states, where n is a subset of or equal to N.

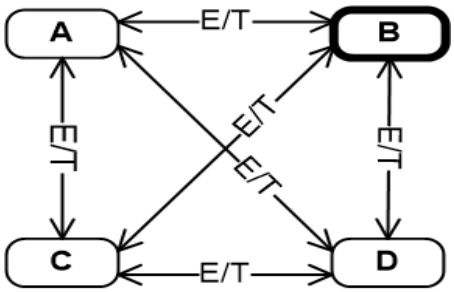

Fig. 2. Representation of a system with n-states and n-combinations of transitions.

From a system perspective, we define a hazardous situation as potentially occurring when a system's behavior results in an undesired system state, consisting of either an unintended combination of two component states that can potentially be detrimental to a human or personal property, and/or a detrimental combination of a component state and environmental state (i.e. a UDCS). Termination in a UDCS does not account for every possible hazardous situation, but it is an easy way to identify a type of hazard that can occur from combination of component states. For example, the undesired combination of a food processor blade turning while the cover is removed. Each state by itself would not be considered as a hazard, whereas combined states could cause harm.

In a hazardous behavior (HB), there is typically one final transition before the UDCS is reached. Stated another way, a behavior is not considered hazardous until the UDCS is reached. For example, in the longest behavior in Figure 3, the

sub-path {D A C}, is not considered a hazardous behavior because the UDCS (B) has not been reached.

Thus, we can say that every hazardous behavior (HB) has a nominal behavior (NB) as a subset. If we view that subset as always being one transition prior to the UDCS, we can define an HB as a NB plus a UDCS. We can then argue that to eliminate an HB, we determine the UDCS and subtract it from the HB. This results in a nominal behavior, NB. In practice, there are two ways to eliminate UDCS. One way is to prevent the UDCS from being reached, and the second way is to recover from UDCS as soon as it is reached. This in effect keeps the aforementioned behavior from terminating (dwelling) in UDCS. Reachability of UDCS can be prevented if we eliminate the UDCS's in-degrees. Recovery can be achieved if we add at least one out-degree to the UDCS that transitions the system into a safe state. Reachability and recovery will become two template categories as we explain the use of UDCS templates in the next section.

A. UDCS Templates

Recall that a UDCS is an undesired state consisting of two concurrent component states, where a component is part of the overall system being protected. UDCSs can also consist of concurrent system environmental states. Component states can be at any level of abstraction, in accordance to however someone decides to partition a system. Some example sets of components as they apply to their system's domain can be:

- {engine, tire} as they apply to a system called "Automobile"
- {automobile, traffic light} as they apply to a system called "traffic flow".
- {thermostat, compressor} as they apply to a system called "Air Conditioner"

Two further examples involving a system's operating environment would be:

- {Car, Pedestrian} as they apply to a system called "traffic flow".
- {gas valve, flame} as they to a system called "Gas fire place"

Each component in turn can consist of a set of possible states, expressed using an attribute(value) notation where the attribute is a given component, and the value is the component's state. Thus, expanding on our prior examples, we can express the following set of component states.

- {engine(idling), tire(worn)} as they apply to a system called "Automobile"
- {automobile(not stopping), traffic light(red)} as they apply to a system called "traffic flow".
- {Thermostat(80), Compressor(on)} as they apply to a system called "Air Conditioner"
- {Car(moving), Pedestrian(crossing street} as they apply to a system called "traffic flow".
- {gas valve(on), flame(off)} as they to a system called "Gas fire place"

The number of component states expressed in a UDCS is two, with the exception of temporal attributes that need be defined in certain situations. Thus, examples of temporal UDCSs include:

- {car(locked), headlights(on), time(>1 hour)}. This UDCS would tend to drain the car's battery, thus should be prevented.
- {Microwave Oven(running), microwave door(open), time(>1 second)}. In this case, the microwave should have stopped running instantly, recovering the UDCS to a safe state.

In general form, we can express a UDCS as: [FORM 1] UDCS = {Cn(Sn), Cm(Su)}, where Cn and Cm are two independent system components states (e.g. {Inlet valve(on), Outlet valve(off)}), or one system component state paired with an environmental "component" state from the system's operating environment (e.g. {Fan(on), Human bystander(near)}). Sn is any given state within the domain of Cn, and Su is any given state within Cm's domain. FORM 1 becomes the basis for a UDCS recovery template:

> IF <Cn> is <Sn> and <Cm> is <Su> THEN <Cn> cannot remain <Sn>

FORM 1 also becomes the basis for a UDCS prevention template:

> WHEN <Cm> is <Su> THEN <Cn> cannot remain <Sn>

Note that prevention is something that is not always preferred. This is because recovery denotes a UDCS that has been reached, which in many cases may be the indicator that a safety function must be enacted. For example, a UDCS that should not occur is that of a microwave oven cooking its food while the microwave door is open {Microwave Oven(cooking), Microwave Door(open)}. However, in practice, a microwave oven door can be readily opened at any time, including while cooking food. To place the microwave oven in a safe state {Microwave Oven(off), Microwave Door(open)}, the opening of the door must be first detected before the system can react by turning off the microwave oven. Thus, the UDCS must be reached and sensed before it is recovered.

When applicable, we have the second possible form for UDCS, which accounts for temporal constraints on when the UDCS is actually considered as a hazardous situation. That second form is expressed as: [FORM 2] UDCS = {Cn(Sn), Cm(Su), Time(value : unit)}, where Cn(Sn) and Cm(Su) are defined in FORM 1, and *Time* is the time period in which the UDCS becomes a hazard situation, and a value is the actual

time expressed as seconds, minutes, or hours, according to the unit under consideration. FORM 2 becomes the basis for the temporal recovery template:

> IF <Cn> is <Sn> and <Cm> is <Su> THEN <Cn> cannot remain <Sn> after <value> <unit>

These templates formally translate a UDCS into a safety rule, which in turn can be used to derive safety requirements and subsequently safety functions. With an understanding of UDCS notation and its Templates, we now look at the procedure used to create the safety rules.

III. The UDCST Approach

In this section, we describe the proposed approach for creating safety rules. The overview of the approach is show in Figure 1. These are the steps to be followed for creating safety rules:

1. We identify components in a system, and their states based on the anticipated functionality of the system detailed in system requirements. In the case of an implemented product we can identify the system components that contribute to the system's functionality. Component states could also be deduced based on the observable functionality.
2. Among the components states of the system, we identify hazard prone components states based on the degree to which a component state contributes to a hazard. For example, a microwave oven that is in a cooking state (Oven(cooking)) is the greater contributor to a hazard than the microwave oven door being open. The combination of {Oven(cooking), Door(open)} is a hazardous situation, but it is a hazard because of Oven(cooking) more so than because of Door(open).
3. Once the hazard prone components states are identified, we generate the UDCSs (detailed in Section II) by using the following two procedures:
    a. We always list the first component(state) in the UDCS pair as the component(state) that is the most hazard prone. We start with a hazard prone component(state) in order to avoid having to generate every permutation of UDCS possible, from the system's total set of component(state)s. The potential for a hazard can be considered as a very low risk when combining two component (state)s, if neither one is considered hazard prone because typically, the components that are not hazard prone do not contribute to a hazard.
4. Having chosen the hazard prone component state, we then combine every other component state with the hazard prone state to create pairwise combination of component states, examining the combination as to whether it presents a potential hazard. Those combinations that do not pose a hazard are ignored.
5. Once the UDCSs are identified, based on the nature of hazardous behavior, we choose one of UDCS templates defined in Section II and write safety rules.

The safety rules identified can be used to create safety functions or test cases. We applied the approach to two real world applications, which are detailed in the next section.

IV. Two Real World Applications

The following two applications demonstrate how the UDCS and templates are used, as well as the usefulness of the proposed approach. In an effort to illustrate the approach under different circumstances, the two applications revolve around two systems that are more commonly known. The two applications include a residential gas furnace and an automobile with a push button start which also demonstrates that our approach can be applied to existing products, and therefore can be used to develop safety rules that can be used in the further verification of the product. Note that the approach can also be used at the early stages of a product's development, as shown by the fact that the components used in the two applications could have been arguably been conceptualized during the products requirements phase. Furthermore, we do not want to imply that the UDCST approach can only be used on a fully developed product. Thus, as we examine each product, it helps to imagine that the needed safety features are being discovered for the first time. We begin with the residential gas furnace.

A. Application One: Residential Gas Furnace

In this application, we examine the start sequence of a typical home gas furnace. The system components associated with the start sequence are shown in Figure 3.

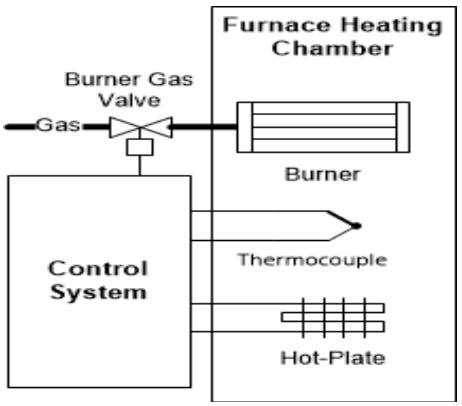

Fig. 3. Components involved in a typical gas furnace.

The sequence starts by activating a ceramic hot plate ignitor, and waiting a pre-set amount of time for the hot plate to get white-hot. After said time, a gas valve opens, allowing natural gas to enter the burner chamber. When the gas comes in contact with the hot plate, the burner should ignite, creating a series of flames that will provide heat. A thermocouple (a device that generates a voltage when it senses a flame) then tells the system controller that a flame is present, and turns off the hot plate ignitor. Note that since the sequence has temporal constraints, we will include Time as the third state.

Many furnaces also have a pre-ignite sequence in which an extractor (in the form of a blower fan) removes any gases that may be in the chamber, before activating the hot-plate. We will ignore the state of the extractor for now, and focus on the ignition sequence. This ability to divide system operations into subsequences also allows us to manage the potential UDCS permutations. We will use a high level representation of the component states involved in the start-up sequence. The states are:

Burner gas(off), Burner Gas(on), Hot-Plate(off), Hot-Plate(on), Thermocouple voltage(0), Thermocouple voltage(>0), Time(xS), where x is defined for derived UDCS.

To keep the number of potential UDCS permutations manageable, we will focus on only those states that are hazard prone. In this case that includes the Burner gas(on) and the Hot-Plate(on), both of which can result in an unwanted fire. In order to select hazard prone component states, we consider system actuators over system sensors. This is because actuators tend to cause actions that are in closer contact with humans interacting with the system. Having determined the potential hazard prone states, we create the following list of UDCSs, by combining every other component state with either Burn gas (on) or Hot-Plate(on), the two hazard prone states. We look for potentially hazardous combinations of states, and compile them as follows:

- UDCS1: {Burner gas(on), Hot-Plate(off), Time(0s)}
- UDCS2: {Burner gas(on), Thermocouple voltage(0), Time(>3s)}
- UDCS3:{Hot-Plate(on), Burner gas(off), Time(>20s)}
- UDCS4: {Hot-Plate(on), Thermocouple voltage(>0), Time(>5s)}
- UDCS5: {Hot-Plate(on), Thermocouple voltage(0), Time(>10s)}

For clarity, we provide the following justification for the derived list of UDCSs:

UDCS1: In this scenario, the gas has been allowed to flow, with no chance of the gas igniting because the Hot-Plate is off. This can result in a dangerous build-up of natural gas in the furnace's chamber. Because the gas should never be turned on without the Hot-Plate being on first, we combine the scenario with a Time state of 0 seconds. In fact, we would consider this a UDCS that uses the prevention template.

UDCS2: In this scenario, the gas has been turned on, but, for some reason, it never ignited after more than 3 seconds. This also factors in the possibility of the thermocouple being defective and thus unable to detect the presence of a flame.

UDCS3: The Hot-Plate is on for over 20 seconds, but no gas flow has been turned on. While it may not pose as severe a hazard as a burning flame, there is still a source of intense heat from the hot-plate. Further, having the Hot-Plate on indefinably could damage it. In this case, we would note that this is a potential hazard, albeit with a low severity level associated with its risk assessment.

UDCS4: The Hot-Plate is still on for over 5 seconds while a flame has been sensed by the thermocouple producing a voltage greater than 0. This again can result in Hot-Plate damage, while having a low severity.

UDCS5: The Hot-Plate is on for over 5 seconds, but the thermocouple is not producing a voltage greater than 0. This indicates that a flame has not ignited, or the thermocouple is damaged. This could be a higher severity level than UDCS3, because a damaged Thermocouple means a flame could be on without the system being notified of its presence.

The first safety rule, resulting from UDCS1, uses a prevention template because we prefer that the gas never be allowed to flow if the hot-plate is off.

UDCS1: ***WHEN <Hot-Plate> is <off> THEN <Burner gas> cannot become <on>.***

The safety rules resulting from the remaining UDCSs are all using the temporal recovery template (Figure 4) because the UDCS must occur first before the system can sense and react to it within a certain amount of time. The safety rules are:

UDCS2: ***IF <Burner gas> is <on> and <Thermocouple voltage> is <0> THEN <Burner gas> cannot remain <on> after <more than 3> <seconds>.***

UDCS3: ***IF <Hot-Plate> is <on> and <Burner gas> is <off> THEN <Hot-Plate> cannot remain <on> after <more than 20> <seconds>.***

UDCS4: ***IF <Hot-Plate> is <on> and <Thermocouple voltage> is <greater than 0> THEN <Hot-Plate> cannot remain <on> after <more than 5> <seconds>.***

UDCS5: ***IF <Hot-Plate> is <on> and <Thermocouple voltage> is <0> THEN <Hot-Plate> cannot remain <on> after <more than 10> <seconds>.***

These safety rules would then be used to create either a more detailed set of safety requirements, or they can be used as the safety requirements themselves. The rule can also be used as test cases, but we do not discuss this further because it is beyond the scope of this paper.

B. Application Two: Button Start Automobile

Our next application involves the safe activation of an automobile engine. In this case, the automobile engine is started by the simple push of a button. Because anyone can push a button (including a child, or someone other than the car's owner), there are certain component state combinations that need to be considered in order to prevent potential harm from the automobile moving during the engine start-up. The system component states we consider are: Engine(start), Engine(off), Transmission(in park), Transmission(not in park), Door(closed), Door(open), Brake(off), Brake(on), Start Button(off), Start Button(on), Key Fob(near), Key Fob(far).

For clarification, a *Key Fob* is often used with automobiles to open and close doors and trunk. In a push button vehicle, the Key Fob also serves as the means of electronically notifying the automobile that the carrier of the Key Fob (i.e. driver) is inside the vehicle. Referring back to the list of component states, to manage the number of permutations we first determine which states can contribute to the greater hazard potential (the hazard prone states). In this case, that would be the En-

gine(start), and the Transmission(not in park). Using these two states, we consider the combination of Engine(start) and all the other states, and repeat the process using Transmission(Not in park). In the end, we derive the following UDCSs.

- UDCS1: {Engine(start), Key Fob(far)}
- UDCS2: {Engine(start),Transmission(not in park)}
- UDCS3: {Engine(start), Door(open)}
- UDCS4: {Transmission(not in park)}, Engine(start)}
- UDCS5: {Transmission(not in park)}, Brake(off)}

UDCS1: This UDCS can be a major source of problems, because not having the Key Fob near is the closest analogy to not having a car key. We definitely do not want the possibility of starting the engine without a Key Fob. This represents a security issue as well.

UDCS2: This is a potential hazard situation with the highest severity. A car jumping forward (or backward) during start-up can cause harmful damage.

UDCS3: The Engine starting with the Door open does not seem to be the source of a hazard on the assumption that the transmission is in park, and/or the Brake is on. However, to err on the side of caution, we should not assume that a third state (Transmission (in park)) would be in place. Instead, we simply consider that in a worst case scenario, the Engine starting with the door open could pose a hazard to someone standing by the open door. The safest position to take is to simply not allow the engine to start with the door open.

UDCS4: This is a repeat of UDCS2. However, it does bring to mind that a safety function that senses the transmission's state can be utilized along with a safety function that monitors the engine state, as a redundant safety feature. In this case, it could even be considered more than a redundancy because monitoring both the engine and transmission states could eliminate the possibility of missing a UDCS that should have been addressed.

UDCS5: This carries the same rationale as UDCS4; we cannot assume that the engine could never be started with both the transmission out of park and the brake not on. However, upon further consideration, we note that it is possible for the vehicle to be on a hill, while in neutral, and does start rolling on its own. UDCS5 should at least raise the question of either a warning (as opposed to direct intervention that overrides the driver's actions) or the addition of a tilt sensor that senses the car on a hill then requires the opposite of UDCS5, namely the driver is prevented from placing the vehicle out of park, unless the brake is on.

The safety rules resulting from the derived UDCSs are all preventative because in this case we are trying to avoid a potential, rather than react to the hazard by placing the system in a safe state. We will also ignore UDCS4 because it is very similar to UDCS2. The four safety rules are:

UDCST1: ***WHEN <Key Fob> is <far> THEN <Engine> cannot become <start>.***
UDCST2: ***WHEN <Transmission> is <not in park> THEN Engine cannot become <start>.***
UDCST3: ***WHEN Door is <open> THEN Engine cannot become <start>.***
UDCST5: ***WHEN <Brake> is <off> THEN <Transmission> cannot become <not in park>.***

As in the first application, the four safety rules above become the basis for incorporating safety functions into the automobile's design.

C. Potential Automation of UDCSTs

The automation of UDCSTs, via a tool, is illustrated in the workflow depicted in Figure 4 (using application two).

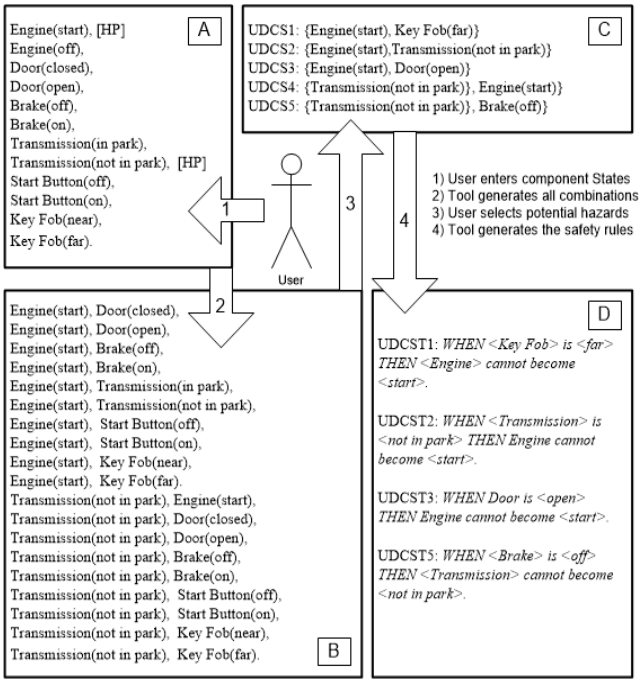

Figure 4: Workflow of a potential UDCST tool.

The tool would involve allowing the user to enter Figure 4, frame A) all the component states with the ability to label the component states that are hazard prone. The tool would then create all the combinations of hazard prone states with other component states (frame B). This would produce a list from which a user can select from, and create a list of USCSs (frame c), whose combinations that are deem hazardous (frame C) That list of UDCSs is then automatically translated into a list of templates (frame D), from which the user performs another round of manual screening. While there would still be some degree of manual intervention, the tool would address the tasks that involve the derivation of permutations. Overtime, some of the manual intervention could perhaps be reduced, and automated. The use of a tool would also alleviate any potential scalability problem that may arise with a very large number of component states. The use of a tool could also allow for the practical analysis of hazard that could arise from two (or more) non-hazard prone component states combining.

V. Discussion and Limitations

We have shown that the UDCS Template (UDCST) approach can facilitate the derivation of a safety rule by allowing a stakeholder to start from the undesired combination states of the system and its operating environment. This approach also allows non-technical stakeholders to view the system at a high enough level to make a safety related contribution during requirements elicitation. While the UDCST approach can be considered simple in concept, the value comes from starting with undesired states, which are readily easier to conceive than an entire hazard scenario.

Also note that we are not claiming that the UDCST should be used at the exclusion of any other approach that a user may already be comfortable using, and this includes FTA and FEMA. Rather, the goal here is to provide a supplemental means of performing hazard assessments, which focuses on the specific class of hazards that come from state combinations (as opposed to single-event hazards). Considering component state combinations can also expose potential hazards arising from component failures. For example, referring back to application one, the UDCS {Burner gas(on), Thermocouple voltage(0)} can be due to the thermocouple being defective, or a plugged-up gas feed to the burner, or the burner holes themselves being plugged-up. While the exact defect cannot be deduced, it is valuable that the question of possible component failure was raised. In addition, the UDCST approach can supplement FTA or FEMA, when finding certain non-single-event hazards with a simpler approach.

Although our approach and results are promising, our approach has some limitations. There are certain types of hazards that cannot be addressed with this approach, such as when a hazard is not associated with a system component or hazards that can occur from an unintended sequence of behaviors. There may also be situations where defining a hazard as an undesired state may not be obvious or easy, such as when states are not clearly defined, or states are not discrete (as in the case of a continuous system). Our approach does not guarantee finding hazards that are possible only because three or more component states must occur concurrently. For example, as in the case where the snow blower engine is on, while the auger is on, and a person is standing in front. There is also the task of having to flesh out the safety rule template into a safety requirement, which may be perceived as an extra step during the requirement phase. However, the fleshing-out can occur during subsequent phases when the safety rules are turned into safety requirements, safety functions, or even test cases. There is also the consideration that while the templates may be viewed as too concise, the abstract level of information it provides can facilitate the elicitation of safety requirements by non-technical stakeholders. Concerning scalability, because the approach depends on identifying undesired states, and not on the system's complexity, there is no inherent hindrance to scalability. As demonstrated, the permutation concerns are addressed by the idea of combining hazard prone states. The relative small size of the applications used in the paper is sufficient to demonstrate the approach, without having use larger example.

VI. Related Work

As we now consider the related research in the derivation of safety requirements, there are three objectives we must consider in relation to our research. They are as follows: 1) begin with an easily identifiable undesired state and work backwards toward a safety rule, 2) the use of concise templates to facilitate the participation of stakeholders, technical and non-technical alike, and 3) the idea of addressing non-single event hazard by examining component state combinations, albeit pair-wise.

With these three objectives we look at the related work in this area. We begin with the common Functional Hazard Analysis (FHA) techniques other than FTA and FMEA, which have already been mentioned as dealing with single-event versus multiple combinational events. Another FHA technique is the use of Dependency Diagrams (DD) [11] that is similar to FTA and describes causal dependencies between components, as opposed to the concurrent dependency between component states, addressed by UDCST. Common Mode Analysis (CMA) [12] is concerned with common characteristics between hardware and software that can lead to single-event failures from similar components, as opposed to Hazards occurring from different components acting concurrently. Zonal Safety Analysis (ZSA) [13] addresses hazards that may occur from interdependent components being in physical proximity, as in the case of redundant aircraft control wires being bundled into the same harness. Thus, ZSA focuses more on hardware interaction than the system state interactions, which UDCST addresses. Particular Risks Analysis (PRA) [14] is also concern with hardware components failures as they physically interact with the environment, as oppose to the hazardous situation that can arise from two components that are perfectly operational. While Common Cause Analysis (CCA) [15] deals the multiple event (rather than single event) hazards, the goal of CCA is to find a common cause, as opposed to the combined effect of two identified components states.

Markov Analysis (MA) [16] deals with the probability of reaching a hazardous situation, whereas UDCST exposes the more certain hazardous situation that will occur based on the combination of two component states. Note that UDCST can complement the prior six techniques by focusing on hazards from component state combinations, and simplifying the creation of safety requirements by using templates.

Beyond the more commonly used techniques, researchers have proposed techniques to address certain problems associated with hazard analysis. Some researchers have combined the elicitation of safety and security requirements on the assumption that they both have a harm element to them, and use UML-based models to capture the safety requirements [8]. Our approach focuses on safety by virtue of the undesired component state. However, an extension of UDCST, based on the unsecure state, is a possibly for future work.

Others have used an input/output constraint meta-automata, to restrict system behavior at the meta level, and/or model interactions between the system, its operating environment and its components [7, 9], however, the required model analysis does not facilitate the creation of requirements as that of the

direct mapping of templates. Menon et al. [10] addresses the elicitation of safety requirements in complex systems, by viewing the system at a higher architectural level. While they take a similar approach to ours in trying to view the system in a way that is conductive to deriving safety requirements, in our approach we try to avoid architectural knowledge of the system, and view any system as a set of interacting component states. While some may argue that hazardous situations from component state combinations could still be assessed without the use of UDCST (e.g. via manual examination), we still see an advantage to making that examination process systematic, thus reducing the chances of overlooking a combination.

Overall, our approach achieves the stated three-fold objective, which is partly achieved by other techniques. This is not to claim that UDCST replaces all other FHA techniques. There are obviously aspects of hazard assessment not covered by UDCST such as probability, propagation, and common causes. However, as a supplemental approach addressing a peculiar class of hazard in an industry friendly manner, we feel that UDCST makes a contribution to the area of hazard assessment.

VII. Conclusion

In this paper, we proposed an approach, using Undesired Combination State Templates (UDCST) that derives safety rules from undesired component state combinations, instead of having to analyze the full complexity of the system. The derived safety rules can then be expanded into safety requirements. The UDCST approach also addresses the class of hazards that are not strictly caused by a single-event. This makes the UDCST approach a useful supplement to the existing approaches such as FTA and FMEA. We used two small, yet real world examples to demonstrate the approach and made the argument of scalability because the approach is dependent on the number of hazard prone undesired states and not the system's level of complexity. Future research that can stem from this work is the use of UDCST to generate safety test cases, and the partial automation of the safety rules from the derived UDCSs.

One lesson that may come from expanding the type of hazards and/or a greater number of applications is the need for more templates. So far, we have been considering a fixed operating environment. As part of future work, we want to expand our technique to find hazards in different environmental contexts, system of systems and adaptive systems. We also plan to address the limitations addressed in Section VI and to offer automated support in finding hazards based on prior knowledge as well as knowledge for similar devices.

Acknowledgment

This work was supported, in part, by NSF CAREER Award CCF-1564238 to University of North Texas.

References

[1] A. George, and J. Nelson, "Managing Functional Safety (ISO26262) in Projects," No. 2017-01-0064, SAE Technical Paper, 2017.

[2] S. ARP, "4761. Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment," December, 1996.

[3] D. J. Smith and K. G. L Simpson, "Functional Safety: A straightforward guide to applying IEC 61508 and related standards," Routledge, 2004.

[4] C. A. Ericson, "Fault tree analysis," In System Safety Conference, Orlando, Florida, pp. 1-9. 1999.

[5] D. H. Stamatis, "Failure mode and effect analysis: FMEA from theory to execution," ASQ Quality Press, 2003.

[6] J. Reason, "Human Error," Cambridge University Press, 1990, ISBN 9781139062367.

[7] D. Aceituna and H. Do, "Exposing the susceptibility of off-nominal behaviors in reactive system requirements," 2015 IEEE 23rd International Requirements Engineering Conference (RE), Ottawa, ON, 2015, pp. 136-145.

[8] C. Raspotnig, P. Karpati, & V. Katta, "A combined process for elicitation and analysis of safety and security requirements," In Enterprise, business-process and information systems modeling, Springer, Berlin, Heidelberg, 2012, pp. 347-361.

[9] Z. Chen and G. Motet, "Modeling System Safety Requirements Using Input/Output Constraint Meta-automata," 2009 Fourth International Conference on Systems, Gosier, Guadeloupe, 2009, pp. 228-233.

[10] C. Menon and T. Kelly, "Eliciting software safety requirements in complex systems," 2010 IEEE International Systems Conference, San Diego, CA, 2010, pp. 616-621.

[11] Guo, H. and Yang, X., 2007. A simple reliability block diagram method for safety integrity verification. Reliability Engineering & System Safety, 92(9), pp.1267-1273

[12] G. Skibinski, R. Tallam, R. Reese, B. Buchholz and R. Lukaszewski, "Common Mode and Differential Mode Analysis of Three Phase Cables for PWM AC Drives," Conference Record of the 2006 IEEE Industry Applications Conference Forty-First IAS Annual Meeting, Tampa, FL, 2006, pp. 880-888.

[13] E. J. P. Cabrera, "System and method for performing a Zonal Safety Analysis in aircraft design," U.S. Patent Application 11/516,029, 2008.

[14] N. V. Kale, F. Ilkay and O. Zysk, (2015) "Particular risk analysis: impact on hybrid aircraft design", International Journal of Structural Integrity, Vol. 6 Issue: 3, pp.402-409

[15] J. L. Rouvroye and E. G. Van den Bliek, "Comparing safety analysis techniques," Reliability Engineering & System Safety, 75(3), 2002, pp.289-294.

[16] J. L. Rouvroye, A. C. Brombacher, "New quantitative safety standards: different techniques different results?", Reliability Engineering & System Safety, vol. 66, pp. 121-125, 1999.