STPA Analysis of Automotive Safety Using Arcadia and Capella

David Hetherington
President Asatte Press, Inc
david_hetherington@ieee.org

Pascal Roques
Independent Author, Trainer, and Consultant at PRFC
pascal.roques@prfc.fr

Abstract

This paper demonstrates the use of the Arcadia methodology and the open source Capella tool to implement a STPA-based analysis technique that augments the conventional HARA, HAZOP. The STPA approach extends the conventional methods to include a holistic perspective considering hardware, software, humans, and control failures in a balanced manner.

Introduction

As embedded software becomes an ever-increasing percentage of the value of an automobile, functional safety and cybersecurity are becoming dominant concerns in the design of both the automotive embedded electronics and the embedded software that runs on that hardware. However, both topics are exceptionally challenging in an automotive embedded software environment.

Current safety methodologies have evolved over the last 30-40 years from a set of practices originally intended for chemical plants.

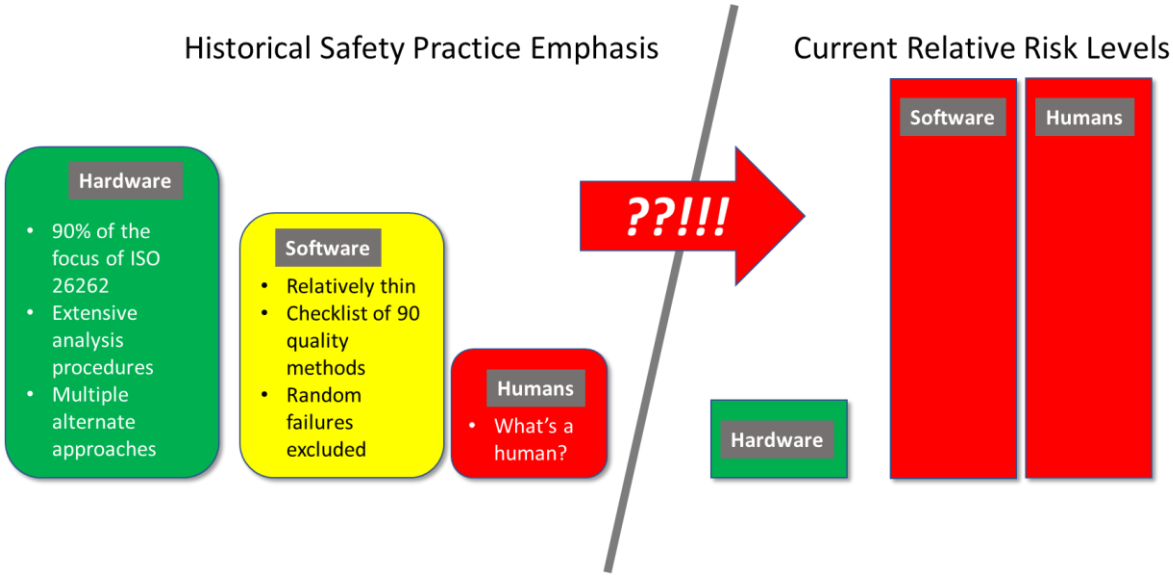

Figure 1 – Historical safety approaches not well-matched to current challenges

IEC 61508 had its origins in industrial plant safety. At the time, the primary concern with cascading hardware failures. This early focus has tended to shape the perspective of other standards such as ISO 26262 that are descended from the original versions of IEC 61508.

The 2018 version of ISO 26262 consists of 12 parts totally 808 pages. Of these, software makes up one part (Part 6) which at 66 pages comprises 8.2% of the standard. For comparison, parts 5 and 11 are completely

about hardware (288 pages). Parts 8, 9, and 10 are nominally applicable to both hardware and software, but a close examination of the detailed recommendations in these parts (For example Part 9, 7.4.4) reveals an almost complete focus on hardware as well. (Another 200 pages).

As for software, part 6 itself consists of 15 tables listing well-established quality techniques in bullet form.

Table 2 — Notations for software architectural design

Notations		ASIL			
		A	B	C	D
1a	Natural language[a]	++	++	++	++
1b	Informal notations	++	++	+	+
1c	Semi-formal notations[b]	+	+	++	++
1d	Formal notations	+	+	+	+

[a] Natural language can complement the use of notations for example where some topics are more readily expressed in natural language or providing explanation and rationale for decisions captured in the notation.

[b] Semi-formal notations can include pseudocode or modelling with UML®, SysML®, Simulink® or Stateflow®.

NOTE UML®, SysML®, Simulink® and Stateflow® are examples of suitable products available commercially. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO of these products.

Figure 2 – Example table from ISO 26262 Part 6 (2018)

Figure 2 above is an example of the thin nature of the standard when it comes to software. The entire topic of model-based engineering is reduced to a single bullet: “semi-formal notations”. Model-based engineering is a huge topic, and the standard provides no depth of thinking about what one might be trying to accomplish with the use of these techniques. Although some work was done between the 2011 and 2018 versions of the standard to improve part 6, the recommendations are still weak when it comes to specific methods for dealing with emergent behavior in software system- of-systems. Interactions with humans are not considered at all. Artificial Intelligence and other types of software with behavior that is not deterministic by design is out of scope in ISO 26262.[1]

The main difficulty with classical automotive safety techniques is that they have become somewhat of a victim of their own success. The hardware reliability of automotive components has improved by orders of magnitude since the 1980s when the industry started thinking in earnest about the problem. On the other hand, the amount and complexity of the software in and around the vehicle has exploded. Even without artificial intelligence, vehicles are already streaming data into the cloud and downloading software updates from the cloud. The complexity of systems like the infotainment system one would find in a current competitive minivan are far beyond the wildest imagination of the 1980s engineers who laid the groundwork for current safety practices. Overwhelmed drivers are a real problem. Software that is so complex that it exhibits what might as well be random failures is also a problem. Our automotive functional safety processes are balanced to fit the challenges of the 1980s, not the challenges of the 2020s.

The “System-Theoretic Process Analysis” or “STPA” hazard analysis technique addresses many of the weaknesses listed above. During the hazard analysis process, STPA looks at control loops within the system. In the STPA technique, hazards are posed by unsafe control actions. This technique is quite helpful in that any of the elements in the loop can be hardware, software, or human. For example, if the controlled process is keeping a car in its lane on the highway, the “process model” that might fail could be the driver’s perception of where the lane is. With the STPA technique, we have an analysis approach that can more evenly and uniformly consider software and humans in the loop. Even better, we do ***not*** have to assume that the software or the humans are “deterministic” for the analysis technique to work.

The first question is what sort of tool, if any, we should use for STPA. Many instructors of STPA apparently discourage the use of modeling tools, perhaps out of fear that the modeling tool will introduce some sort of tunnel vision regarding the system.

[1] The ISO 26262 community has developed the separate SOTIF standard to address this gap. See [3]

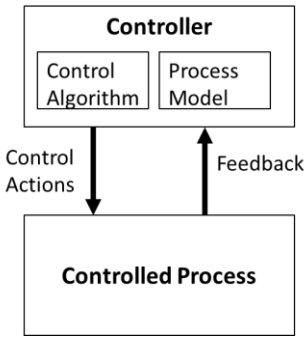

Figure 3 – Generic control loop [2]

While that concern cannot be discounted, an actual safety analysis at the scale needed for a commercial vehicle is simply not feasible without tooling.

By the time a vehicle first rolls off of the assembly line, the vehicle maker and its multiple tiers of suppliers will have worked their way through hundreds of thousands of context elements, system elements, hazards, failures, effects, and other related information. This scale of analysis simply cannot be done with paper and clipboard or even with spreadsheets. Intelligent modeling tools are mandatory. It is not feasible to design a current generation commercial automobile without them.

A number of suppliers make purpose-built safety analysis tools. Many of these purpose-built safety analysis tools include "SysML-like" features that are tailored for safety analysis work. Some tools also claim support for STPA. Unfortunately, neither author has current license for any of these tools. As such, the authors are not in a position to evaluate these sorts of purpose-built tools and they are out of scope for this paper.

That leaves model-based systems engineering (MBSE) tools as candidates for conquering the complexity that would be involved in a full-scale STPA analysis of an entire vehicle or a large subsystem within a vehicle. Within the field of MBSE tools, SysML tools are obvious candidates for performing this sort of analysis and many companies do at least some parts of their safety architecture work using SysML tools. Both authors are quite familiar with SysML tools. In fact, David Hetherington publishes beginner books for SysML tools and uses SysML tools regularly for functional safety modeling.

Recently, however, the authors have begun collaborating on a beginner book for the Capella tool and the Arcadia method. During this work, the authors noticed that Arcadia has some special characteristics that are well-suited for functional safety work and STPA in particular. This rest of this paper will demonstrate an approach to using the open source Capella tool and the Arcadia methodology[3] to perform the STPA analysis. The specific MBSE features of the tool and method that are convenient for STPA analysis will be highlighted.

Getting to the Starting Line

There are a few things we need to do to get to the starting line to use the analysis technique laid out in the STPA Handbook.[4]

System of Interest

Our system of interest is the "Bold Truck" Electric Sport Utility Vehicle.

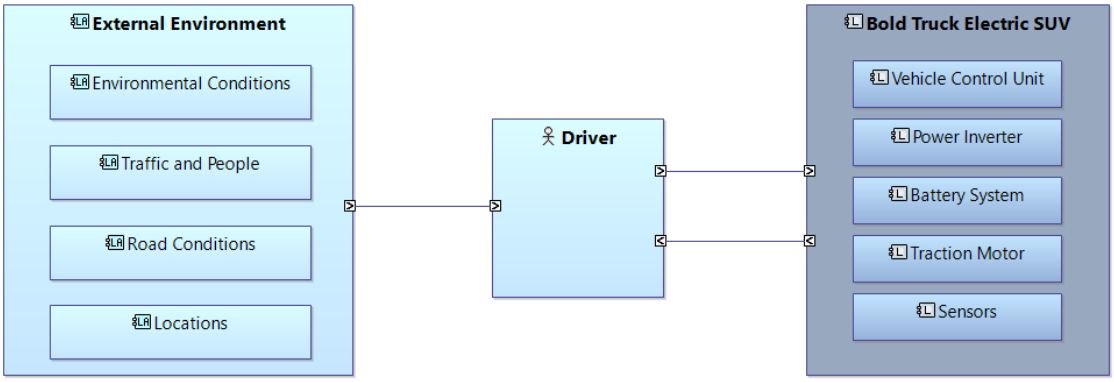

Figure 4 – The Bold Truck electric sport utility vehicle

[2] See [2] Adapted from STPA Handbook Figure 2.6 on page 23
[3] See **Erreur ! Source du renvoi introuvable.**
[4] See [2] to download the STPA Handbook.

Figure 4 shows the truck in its context. Of course, the truck would have a lot of other subsystems. For this paper, however, we are going to focus on a "narrow slice" to explore how STPA would be used to model safety hazards in the control of the electric motor.[5]

Modeling Setup

For this paper, we used Capella 5.1. We also installed the requirements add-on. [6]

Analysis Procedure

For the context analysis, we will follow the definition of "TypicalAutomotiveSituation" from the OMG Risk Analysis Modeling Language (RAML) 1.0 Beta specification[7] with the modification that we will replace the word "Typical" with the word "Valid", the word "Typical" being a little too open-ended for our purposes.

In order to keep the size of this paper manageable, we will focus on just one valid situation.

Name	Vehicle Usage	Traffic and People	Road Condition	Location	Environmental Condition
Freeway	Driving forward at >100 km/hr	Light traffic. Nearest car is 15 seconds away.	Clean, dry, asphalt	Public high-speed highway	Warm, sunny, dry, normal humidity

Table 1 – Valid automotive situation

A real-life analysis of an entire vehicle would start with a large number of such situations, perhaps 100 or more. For example, the control actions and hazards for backing out of a driveway would be quite different from those for driving on a freeway. Likewise, driving in snow or rain would present different control behaviors than driving in nice weather.

The first thing we will model is the Freeway valid automotive situation.

Below is a specific Arcadia diagram called "Contextual System Actors", modeled at "System Analysis" level. The Bold Truck Electric SUV is considered as a "black box", and all external entities are called "Actors" (as in UML and SysML). We used the "constraint" concept, still as in UML and SysML and noted with {c}, to model the qualifying scope constraints of the valid automotive situation.

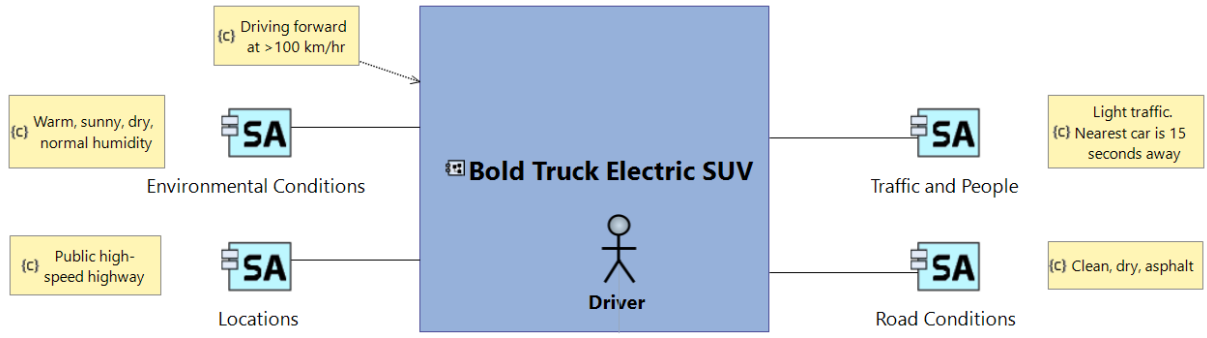

Figure 5 – The Freeway valid automotive situation

With the context defined, we can proceed with the steps laid out in the STPA Handbook.

STPA Step 1: Define the Purpose of the Analysis

Let us start now with the first STPA step: "Define Purpose of the Analysis".

[5] See [5] for an excellent discussion of the functionality and safety concerns for such a power inverter.
[6] See [7] for download of Capella and also the requirements add-on
[7] See [4] Figure 9.124 - TypicalAutomotiveSituation

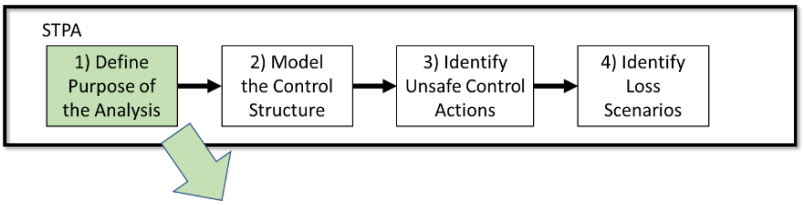

Figure 6 – STPA Step 1: Define the purpose of the analysis[8]

The first step consists of four parts:

1. Identify losses
2. Identify system-level hazards
3. Identify system-level constraints
4. Refine hazards (optional)

Identify Losses

In the first step, we need to identify the potential losses at the system level.[9]

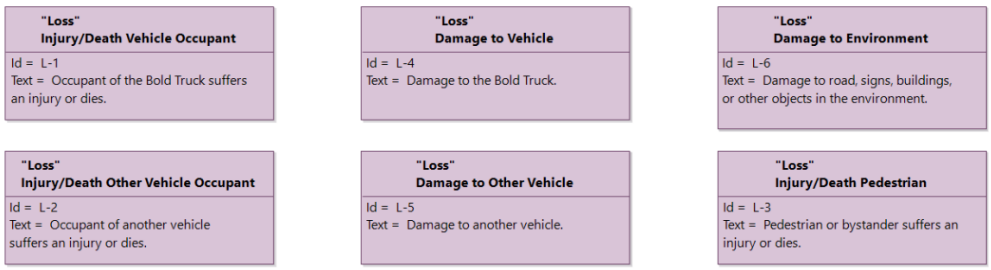

Figure 7 – Freeway: identify losses

In order to capture the losses in the model, we have created a new requirement type, using the Capella "Requirements viewpoint" add-on. For the Ids we follow the convention shown in the STPA Handbook.

Identify System-level Hazards

The next step is to identify the system-level hazards and tie them to losses. A key point here is that the methodology and the tool can help, but it is ultimately the humans who identify the hazards. The methodology and the tool merely provide a framework to stimulate productive thinking and help keep track of the hazards identified by the humans.

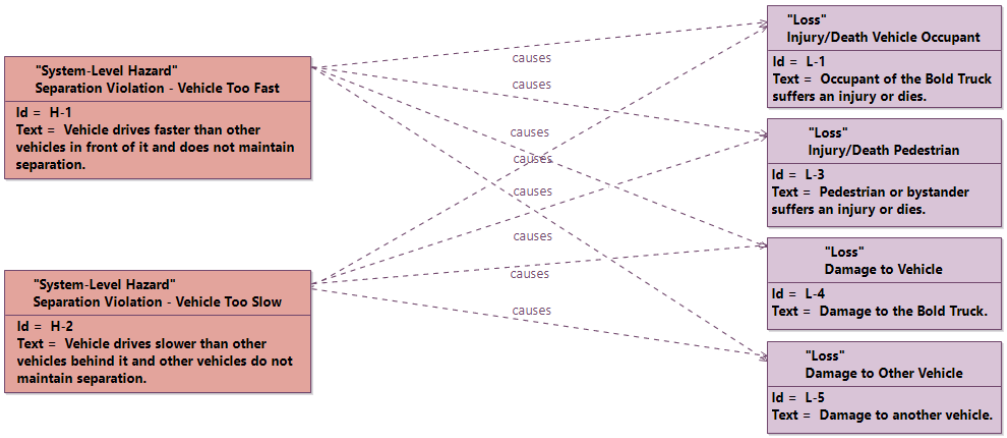

Figure 8 – Freeway: identify system-level hazards

[8] See [2] Adapted from STPA Handbook Figure 2.2 on page 15
[9] See [2] page 16 for the formal definition of a loss.

Here we create a “system-level hazard” type and the “causes” relationship. As it turns out, all of the hazards that we have identified can cause all of the losses. That is a coincidence and would not be the case in general.

Identify System-level Constraints

Here the goal is to identify constraints that will prevent or at least mitigate the hazards identified in the previous step and thereby prevent the losses from occurring.

Figure 9 – Freeway: System-level constraints

In the interest of brevity, we have shown only one system-level constraint here.

STPA Step 2: Model the Control Structure

Now we are ready to proceed to the second STPA step: “Model the Control Structure”.

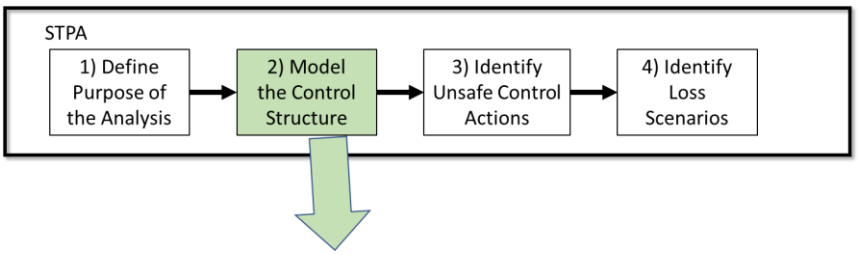

Figure 10 – STPA Step 2: Model the control structure[10]

The first level of control loop is between the driver and the vehicle with input from the weather and scenery.

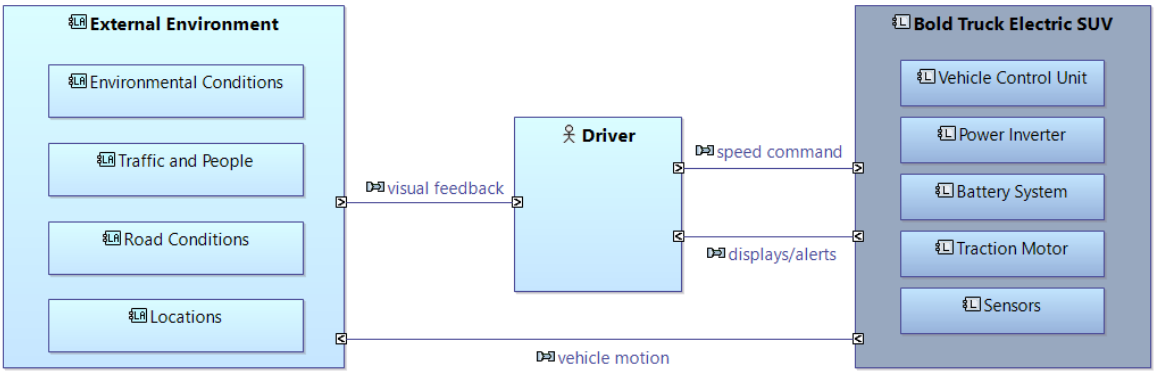

Figure 11 – Control loop: driver, vehicle, and weather/scenery

This is our first option for an Arcadia diagram to show a control loop. We used a “System Architecture Blank” diagram in Capella, which is similar to a SysML internal block diagram (*ibd*). The light blue rectangles represent external actors, as in Figure 4. Notice that small arrow icons inside the ports indicate the direction of flow between the structural blocks (system / actors). As the vehicle moves, the changing position of the vehicle in the environment causes changing visual feedback to the driver. The visual feedback from the environment as well as potential alerts from the vehicle itself cause the driver to take action to increase, decrease, or maintain speed as needed.

[10] See [2] Adapted from STPA Handbook Figure 2.5 on page 22

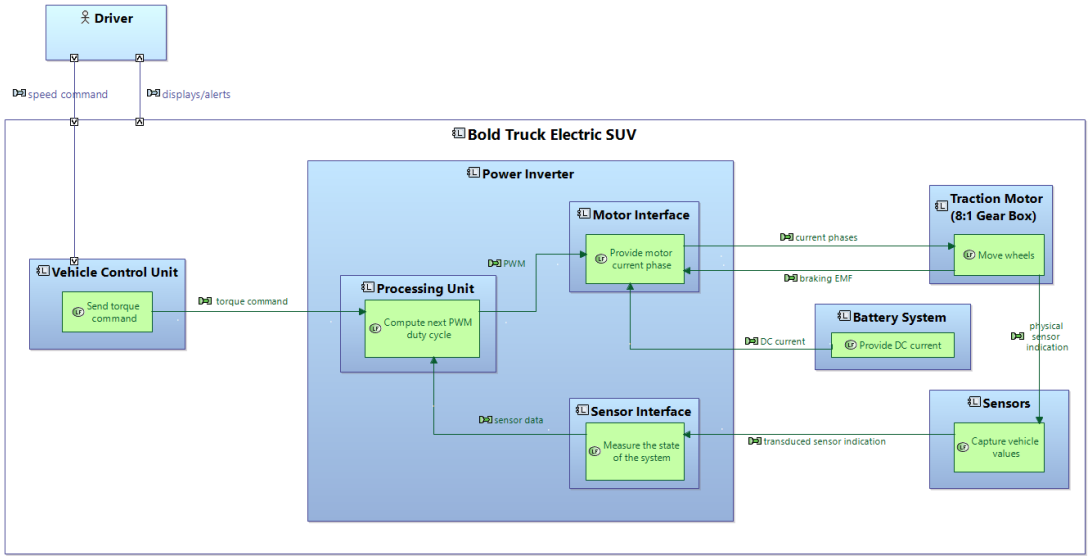

Figure 12 – Level 2: Power inverter internal control loop with Arcadia functions and functional exchanges

Within the vehicle, the power inverter controls the motors. The main subsystems involved, as shown in Figure 4, are now connected by means of functions and functional exchanges. The diagram we used is a "Logical Architecture Blank" diagram from Arcadia. This type of diagram allows us to represent not only logical components inside the system but also the functions allocated to the components. We can see the control loop just by following the sequence of functional exchanges: "torque command" going into the Processing Unit of the Power Inverter, then the outgoing pulse width modulation signal ("PWM"), which is used to create positive and negative three-phase alternating current ("current phases") to feed the motor. Coming back, current, temperature, and other physical properties ("physical sensor indication") are monitored by sensors that transduce the physical phenomena into electrical signals. Back inside the power inverter, these sensor electrical signals are transformed into meaningful digital data for use by the processing unit.

Arcadia also has a very useful concept called "Functional Chain" (which is missing from SysML). A functional chain is an ordered set of references to functions and the functional exchanges that link them, describing one possible path among all the paths forming the dataflow. Here we modeled the control loop as a specific functional chain. The functional chain is a model element itself, which means we will be able to assign non-functional properties such as requirements directly to the functional chain.

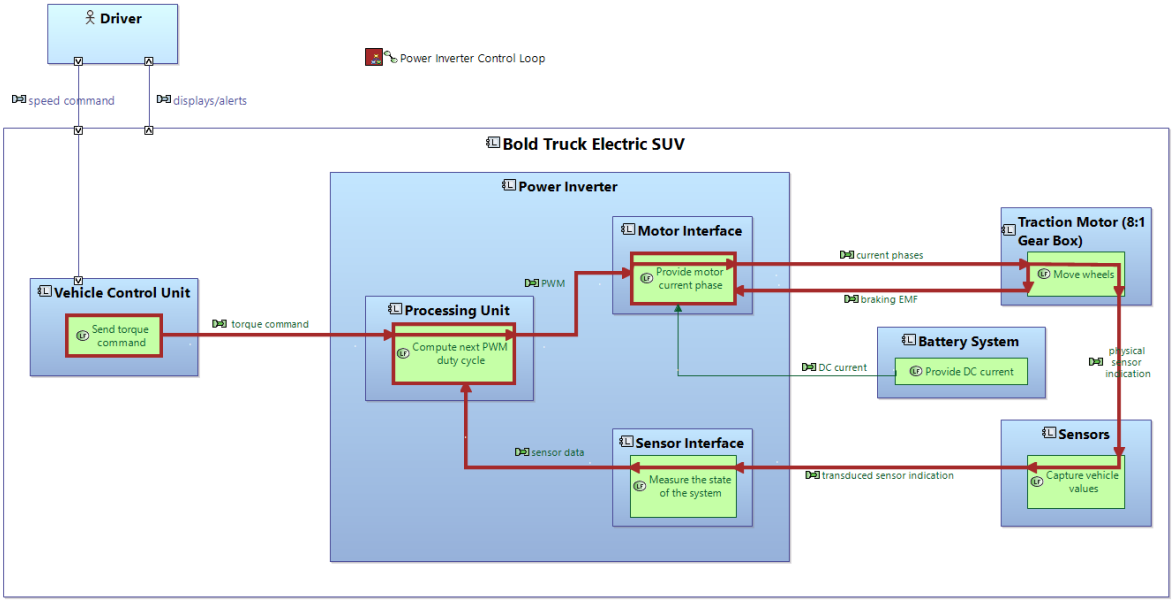

Figure 13 – Level 2: Power inverter control loop with Arcadia functional chain

For safety analysis (STPA or conventional methods like dependent failure analysis) the Arcadia functional chain is really useful. Essentially, the functional chain allows the safety engineer to depict a use case's flow through the system while showing both information moving between components/functions as well as information being processed within the components/functions. This duality is important because safety-related failures can occur both within the components/functions and also in the path between functions/components.

SysML can partially represent a similar concept, but the SysML internal block diagram does not allow the modeler to depict an item flow *through* a component, making it difficult to unambiguously show the sequential path. SysML sequence diagrams can also be used to some extent, but it is cumbersome to show a path failure inside a lifeline. In SysML it is also difficult to link a failure directly to an item flow or to a message in a sequence diagram. As we will see below, this sort of linking is quite easy in Arcadia.

STPA Step 3: Identify Unsafe Control Actions

Now we are ready to proceed to the third STPA step: "Identify Unsafe Control Actions".

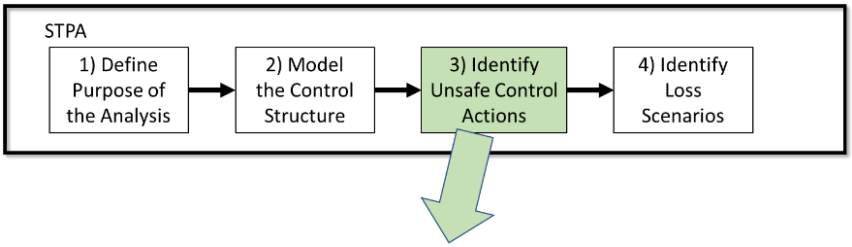

Figure 14 – STPA Step 3: Identify Unsafe Control Actions[11]

An Unsafe Control Action (UCA) is a control action that, in a particular context and worst-case environment, will lead to a hazard. Using Capella and the "Requirements Viewpoint" add-on, we will once again create a new requirement type for unsafe control actions. We can create specialized "UCA" requirements and link them with any Arcadia concept, such as a single link on a functional chain.

Looking at the functional chain shown in Figure 13 on page 7, when the processing unit sends the pulse width modulated signal (PWM) to the motor interface, this flow can be considered to be a control action "Provide PWM Signal". Using the standard STPA questions, this control action can be mirrored with four potential unsafe control actions.[12]

Control Action	Not Providing Causes Hazard	Providing Causes Hazard	Too Early, Too Late, Out of Order	Stopped Too Soon, Applied Too Long
Provide PWM Signal	UCA 1 - PWM signal not provided	UCA 2 - PWM signal provided erroneously	UCA 3 - PWM signal provided prematurely	UCA 4 - PWM signal halted prematurely UCA 5 - PWM signal provided after vehicle stopped

Table 2 - Potential unsafe control actions for Provide PWM Signal

[11] See [2] Adapted from STPA Handbook Figure 2.14 on page 35
[12] See [2] STPA Handbook Table 2.3 on page 36

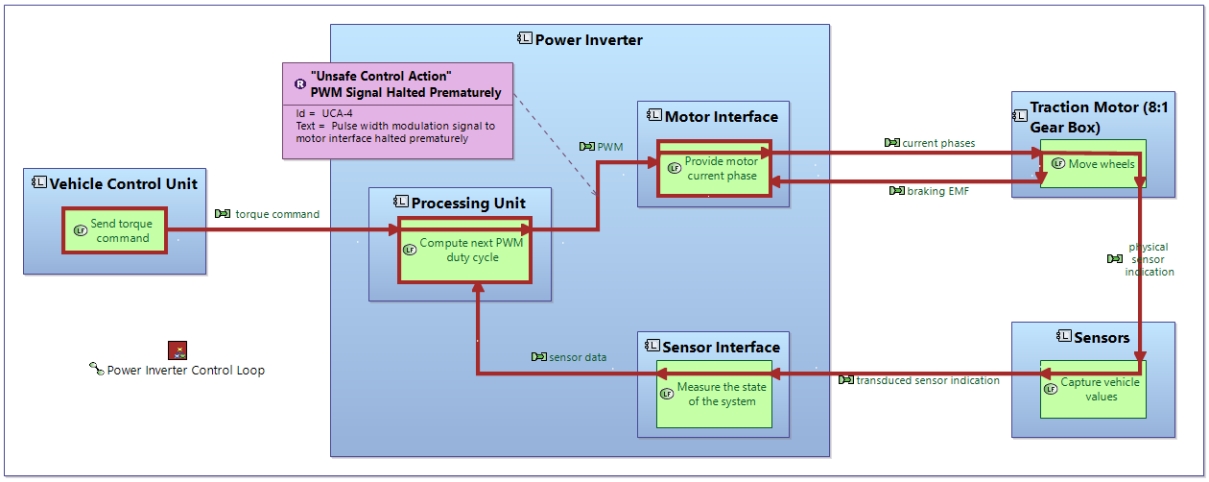

Figure 15 – Level 2: UCA linked to Arcadia functional chain

Once identified, the unsafe control action can be linked back to a specific hazard that it causes in a manner similar to the linking of the system constraints to hazards as shown in Figure 9 on page 6. In the interest of brevity, we have not provided a diagram for this step.

STPA Step 4: Identify Loss Scenarios

Now we are ready to proceed to the fourth STPA step: "Identify Loss Scenarios".

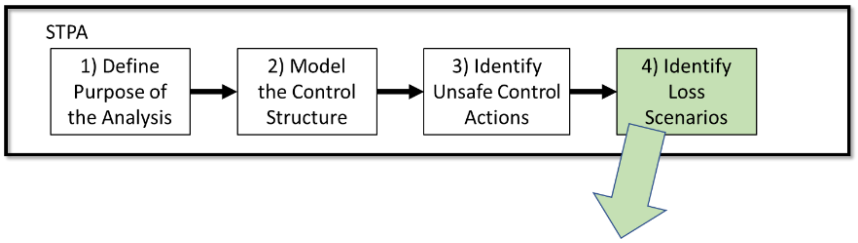

Figure 16 – STPA Step 4: Identify loss scenarios[13]

Loss scenarios are the final step in the STPA analysis technique. This step is where the cause of the hazard comes together with the resulting unsafe control action to cause the hazard. Again, we can create a specialized Arcadia requirement type for the STPA scenario.

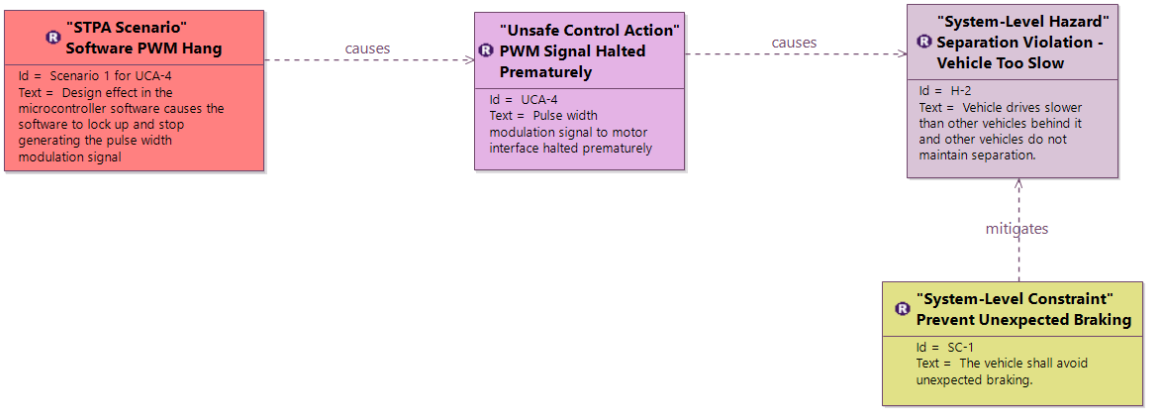

Figure 17 – Failure causes unsafe control action and hence hazard

[13] See [2] Adapted from STPA Handbook Figure 2.16 on page 42

In the example loss scenario shown in Figure 17, a bug in the processor software causes the software to hang, interrupting the PWM signal and causing the hazard that the vehicle brakes suddenly and is too slow relative to the traffic behind it.

Notice that the using the STPA analysis techniques and Arcadia functional chains, only the links between functions are candidates for unsafe control actions. The links that go through a function are the potential location of failures which become scenarios that cause the unsafe control action on the output link.

Discussion

While this example was necessarily brief, the Arcadia method, the Capella tool, and the STPA Hazard analysis technique can all contribute to a robust automotive safety program. Although STPA does not fully replace all other hazard analysis techniques, it can be used as an additional technique to cover weaknesses in the more conventional techniques regarding human factors as well as unexpected emergent behaviors caused by the complex interaction of a large number of software components sourced from different suppliers interacting with each other and less stable environments such as the internet.

STPA can be implemented using SysML tools or specialized safety analysis tools. However, the Arcadia method and Capella tool offer some convenient features that ease the analysis. In particular, functional chains are a very good fit for the fundamental control loop approach of STPA. More subtly, specialized requirement types can be created using the requirements add-on to represent the different STPA concepts like unsafe control actions and these can be placed on Arcadia diagrams and tied directly to specific links in a functional chain. While it is easy enough in SysML to create specialized requirement types, it is quite difficult or impossible in SysML to connect these directly to item flows, elements in a sequence diagram, or elements in an activity diagram.

Follow-Up

The techniques in this paper show promise, but we have not presented a full solution that would be ready for commercial deployment. As described in the introduction of [2], STPA is a hazard analysis technique. While hazard analysis is a crucial first step in a functional safety process, there is a lot more to functional safety than just identifying the hazards. A more complete implementation of the Arcadia method demonstrated in this paper would integrate the STPA method shown into a full, end-to-end automotive safety process.

Both authors will be delighted to answer follow-up questions on this paper and the Arcadia technique presented. If you would like a copy of the Arcadia model used in this paper, and created with Capella 5.1, please send an e-mail to either or both authors.

References

[1] ISO 26262 - https://en.wikipedia.org/wiki/ISO_26262

[2] STPA Handbook - https://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf

[3] SOFTIF - ISO/PAS 21448:2019 Road vehicles — Safety of the intended functionality https://www.iso.org/standard/70939.html

[4] RAAML - https://www.omg.org/spec/RAAML/

[5] Safety Concept Overview Of High-Voltage (HV) Traction Inverter
Antoine Dubois, Erik Santiago, Sandeep Kumar (NXP)
https://www.avnet.com/wps/wcm/connect/onesite/c7840908-f6bd-4df7-a07e-dd380d7f9bdc/nxp-SAFETY-CONCEPT-OVERVIEW-HV-TRACTION-INVERTER.pdf?MOD=AJPERES&CVID=nHyIEM9&CVID=nHyIEM9&CVID=nHyIEM9&CVID=nHyIEM9&CVID=nHyIEM9&CVID=nHyIEM9&CVID=nHyIEM9&CVID=nHyIEM9&CVID=nHyIEM9

[6] Arcadia Datasheet: https://www.eclipse.org/capella/resources/Datasheet_Arcadia.pdf

[7] Capella Download web site: https://www.eclipse.org/capella/download.html