Modeling Modes of Operation Using AADL and AGREE

Danielle Stewart

April 16, 2019

1 The Point of it All

In critical system development, it's important to have redundancy in the system and introduce modes of operation. For instance when everything works properly, we are in normal mode. If a failure occurs in normal mode, we can switch to a backup or alternate mode, and so on. When modeling this in AADL and AGREE for a large scale Wheel Brake System, we ran into some issues once we generated minimal cut sets. For minimal cut sets of cardinality 1, faults were showing up in normal and alternate mode. This shouldn't be happening if the behavior of the components is modelled correctly. If a fault occurs in normal mode, the system should automatically switch to alternate mode and still provide the service as intended. For a system with n backup modes (or redundancy plans), we should see tolerance up to n faults.
First I will briefly describe the Wheel Brake System and then describe the small scale system created to sort out what contracts are needed to enforce switching modes and testing such behavior. For a more detailed description of the large scale WBS, see the Tech Report: `https://www.cs.umn.edu/sites/cs.umn.edu/files/tech_reports/18-007_0.pdf`.
The small scale system used to implement and test switching modes is located in repository: `https://github.com/dkstewart/University-Research/tree/master/Models` and is titled *Modes of Operation*.

1.1 WBS Problem Description

The overall behavior of the WBS is simple enough. When the brakes are commanded, we should recieve brake force at the wheel. There are three operating modes in the WBS model. In *normal* mode, the system uses the *green* hydraulic circuit. In *alternate* mode, the system uses the *blue* hydraulic circuit. The last mode of operation is the *emergency* mode. This mode is entered if the blue hydraulic line fails. The accumulator pump has a reserve of pressurized hydraulic fluid and will supply this to the blue circuit in emergency mode.
In Figure 1, the simplified architectural structure of the WBS can be seen. The blue, green, and accumulator pumps are shown and the main idea of the

structure is shown. The *selector* valve selects which line the system utilizes and the accumulator input occurs when no blue output is provided from the selector valve.

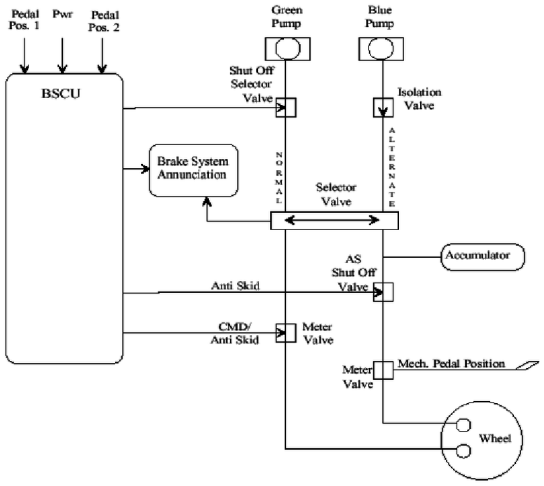

Figure 1: Simple Structure of the WBS System

Once the system switches modes, it does not switch back. This point becomes important later when we talk about enforcing mode behavior through contracts. What is not shown in the figure is the presence of a *monitor system* in the WBS that monitors the health of the system and determines if the system is providing valid electrical commands. This component provides a *sys_valid* variable to signal if we have healthy commands coming from the *braking system control unit*.
When generating MinCutSets, single points of failure were located in the normal and alternate modes of operation. The point of multiple modes of operation is to *avoid* single points of failure. Thus it was clear there was something wrong with the contracts of the model. Given the size of the WBS, it was prudent to put together a small scale model similar in nature to the WBS. This allows the easy addition of contracts and testing to make sure that we have desired behavior with regard to these modes.

2 Modes of Operation Model Description

The smaller model was built based off of the general architectural structure (subcomponents and connections) of the WBS. Top level has supply as inputs. This "supply" is always positive and never zero.
Subcomponents:

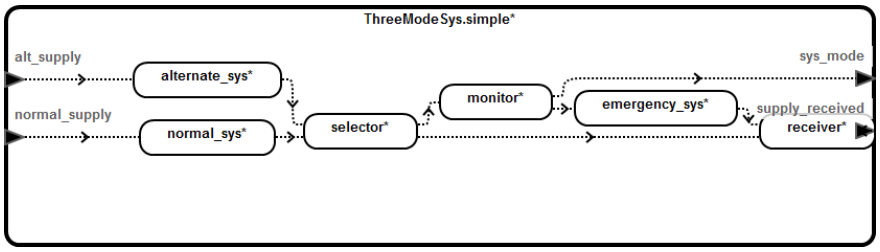

Figure 2: Information Flows of the Modes of Operation System

- Normal : Takes supply input from top level. Contract guarantees that supply out equals supply in. Output sent to Selector. Fault defined for this subcomponent is: stuck at zero output.
- Alternate : Takes supply input from top level. Contract guarantees that supply out equals supply in. Output sent to Selector. Fault defined for this subcomponent is: stuck at zero output.
- Selector : is a valve that takes the supply from both normal and alternate components and selects between them. Its behavior is very much like the selector valve in the WBS. The behavior is a bit more involved for this component.

  Inputs: normal_supply, alt_supply
  Outputs: valve_position, supply_out

  The valve position indicates which supply the system is currently using. It is defined as follows.
  Position 1: Normal supply
  Position 2: Alternate supply
  Position 3: Neither (emergency signal)

  Normal supply behavior:

  Normal is supplied and we were previously in normal mode
  $(normal\_supply > 0) \wedge (prev(valve\_position) = 1)$

  Valve position is 1.

  Alternate supply behavior:

  Case 1: Previously normal mode, normal supplies nothing, alternate is supplied
  $(((normal\_supply \leq 0) \wedge (alt\_supply > 0) \wedge prev(valve\_position) = 1)$

  Case 2: Previously alternate mode, alternate is supplied
  $((alt\_supply > 0) \wedge prev(valve\_position) = 2))$

  In both cases, valve position is 2.

Emergency supply behavior:

Case 1: Previously normal, neither normal nor alternate are supplied
$prev(valve\_position) = 1) \wedge (alt\_supply \leq 0) \wedge (normal\_supply \leq 0)$

Case 2: Previously alternate, alternate supplies zero
$prev(valve\_position) = 2 \wedge (alt\_supply \leq 0)$

Case 3: Previously emergency
$prev(valve\_position) = 3$

In all three cases, we set valve_position to 3.

This logic also ensures that we do not switch back to previous modes. This is clearly seen by the use of the *prev* statements. When previously in a certain mode, we keep it that way (or move on to a more serious mode). If it is desired to let the system switch back to previous modes, this is the component that would regulate that logic for the most part.

The supply output guarantee links the valve position with which input we use to equate with the output.

if $(valve\_position = 1)$ then $(normal\_supply = supply\_out)$
else if $(valve\_position = 2)$ then $(alt\_supply = supply\_out)$
else $(supply\_out = 0)$

- Monitor : Monitors the mode of the system (similar to the MonitorSystem in WBS). It gets input from the Selector on whether its getting supply from normal or alternate and if the supply is not received at Selector, then it sends command to Emergency. Emergency then sends pressure to Receiver. The system mode is sent to the top level for testing purposes.

  The logic of the monitor is surprisingly simple. The selector sends valve postion and based on the valve position, the system mode is determined. When the valve is in position 3 (indicating selector is getting no supply from normal or alternate), the monitor sends emergency command to emergency subcomponent. The system mode is then sent to the top level for testing purposes.

- Emergency : Backup reserve supply. If there is a command from the monitor to activate emergency system, then supply output is a constant value (5). Else output is zero. Fault defined for this subcomponent is: stuck at zero output.

- Receiver : the "end of the flow" subcomponent that needs to receive the supply. Similar to the wheel in WBS. It takes input from the selector and from the emergency component. The only output is to the top level stating that it has (or has not) received supply.

The logic is very basic. If no normal supply at selector, then switch to alternate. Likewise alternate switch to emergency. Faulty behavior is defined when the supply is zero. The bulk of the necessary logic in order to make these mode

changes resides in the selector. The cases that cover what mode system is in was important to get right and took some trial and error (i.e. studying counterexamples that showed the missing cases).

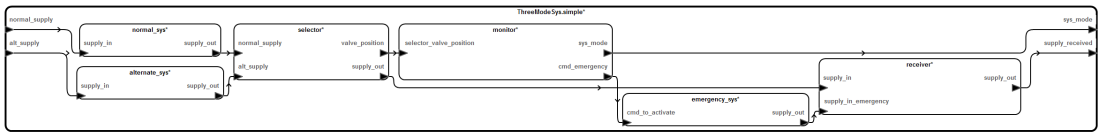

Figure 3: Architecture of the Modes of Operation System

3 Top Level Contracts

The main contracts of interest are shown in Figure 4. In particular, the first lemma *"Receiver supply is always positive"* can indicate single points of failure when faults are activated. Given either 1 or 2 active faults in the system, this lemma still proves. This is sufficient to show that the system does in fact switch modes.

```
-- Supply is always provided at receiver.
lemma "Receiver supply is always positive." :
    (supply_received > 0);

-- Make sure system modes are bounded.
lemma "System modes are bounded." :
    (sys_mode > 0) and (sys_mode < 4);

-- Lemma to make sure we do not switch back to normal mode.
lemma "When we switch to alternate or emergency mode, we do not change back to normal." :
    ((prev(sys_mode, 1) = 2)) => not((sys_mode = 1));

-- Lemma to make sure we do not switch back to alternate once in emergency mode.
lemma "When we switch to emergency mode, we do not change back to alternate or normal." :
    ((prev(sys_mode, 1) = 3)) => (not(sys_mode = 2) and not(sys_mode = 1));

lemma "System mode reflects selector position." :
    (selector.valve_position) = (sys_mode);
```

Figure 4: Top Level Contracts

There are no MinCutSets of cardinality one for this system. The simple fault

tree is shown in Figure 5.

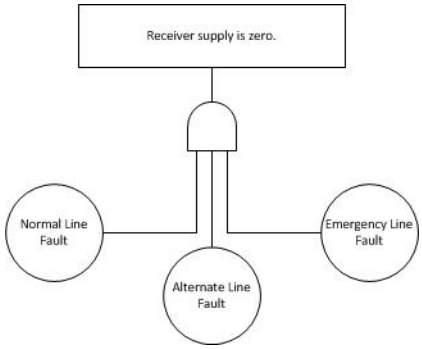

Figure 5: Fault Tree

4 Difficulties, Thoughts, Tips

The biggest key was the behavior of the selector valve. Every counterexample displayed a missing case and that was incredibly helpful to make sure all cases were covered. Once that was in place, the job was finished. Since the smaller model was based on the WBS, the behavior/contracts of the selector valve and monitor will have to be studied to make sure they are compatible with the smaller model. Easier said than done, right? :)