Building Safer UGVs with Run-time Safety Invariants

Michael Wagner, Phil Koopman, John Bares, and Chris Ostrowski

mwagner@cmu.edu

October 28, 2009

The Message

- **To be useful, unmanned ground vehicles (UGVs) must safely operate alongside personnel, although this is not yet reliable enough with today's technology.**
- **The use of physical safety barriers and large stand-off distances is acceptable only during testing; it is infeasible for use in the real world.**
- **We are developing safeguards to reduce dependence on physical barriers and large standoff distances for UGV operating alongside personnel in real, dynamic operations.**

Presentation Synopsis

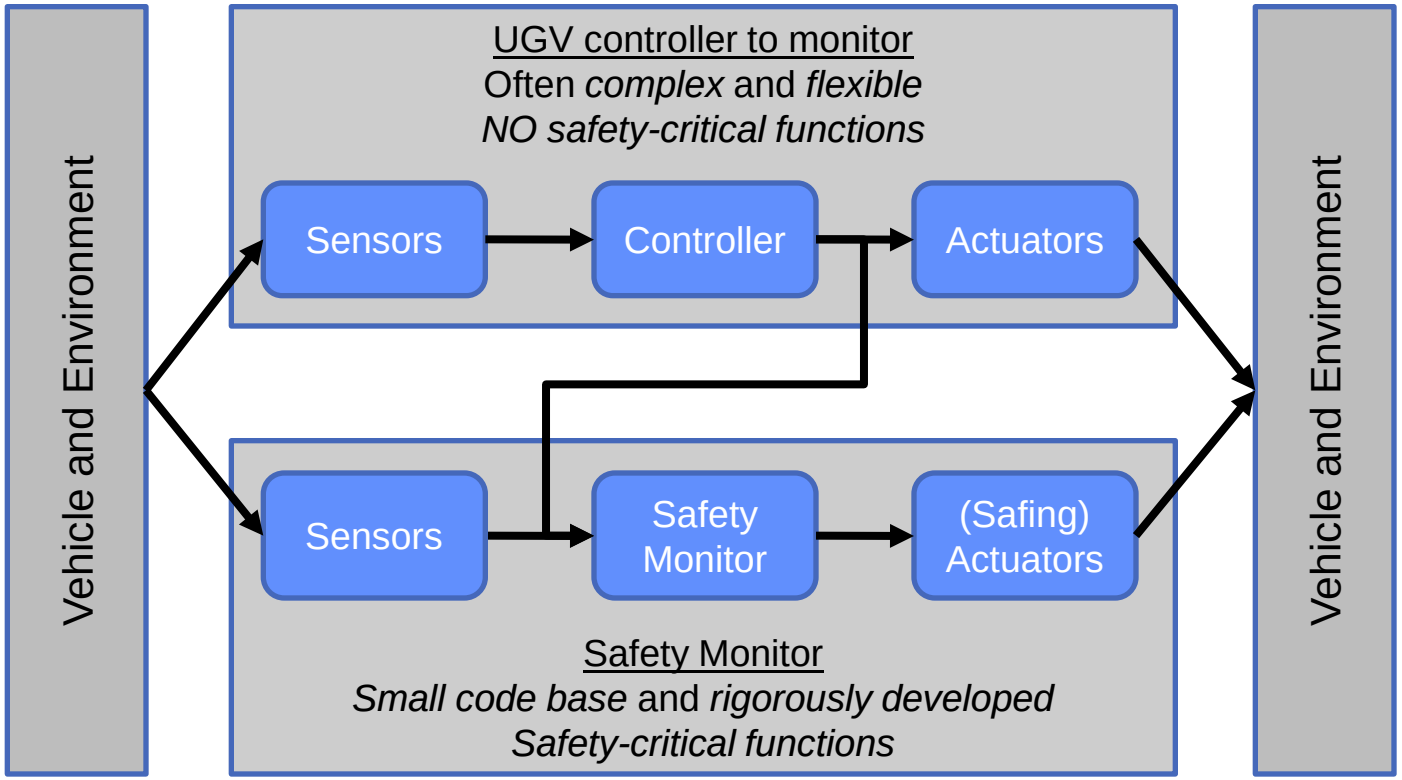
- **Our approach is based on *run-time safety invariants* enforced by a *Safety Monitor***
- **Benefits of our approach involve**
  - →**A clear definition of "safety"**
  - →**Firewalling safety-criticality to a small set of components**
  - →**Streamlined V&V of safety-critical components**
- **We are implementing our approach on the Autonomous Platform Demonstrator project**
- **We will discuss our process for developing a Safety Monitor using the Autonomous Platform Demonstrator (APD) as an example**

Our Approach

- ***Run-time safety invariants*** **are concise, formal expressions of critical system properties that define system safety**
  - →**E.g., "vehicle speed doesn't exceed operator-specified limit"**
  - →**We needn't enumerate detailed causes of hazards**
  - →**Rather, we create a dependable outer bound on what it means to be "safe"**
  - →**Do this based on fault-tree analysis**

Our Approach (2)

- **We then build a *Safety Monitor* that safes the UGV whenever any invariant is violated**
  - →**Has a dependable means of sensing invariant state**
  - →**Has a dependable means of safing the system**



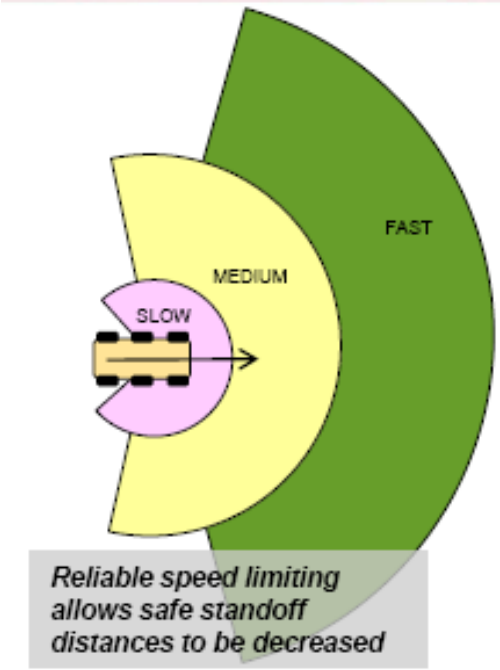
Demonstration Vehicle: APD

- **APD is developing, integrating, and testing next generation UGV mobility technologies such as hybrid electric drive systems, advanced suspension systems, and efficient auxiliary systems.**

TARGET GVW: 8,500 kg
TARGET SPEED: 80 km/hr

APD Safety Goals

- **Initial focus is on mitigating hazards involved with driving the APD vehicle**
  - →**Ensure the vehicle can be stopped when commanded**
  - →**Ensure the vehicle maintains a commanded speed limit**
- **Meeting both these goals helps to decrease safe standoff distances**

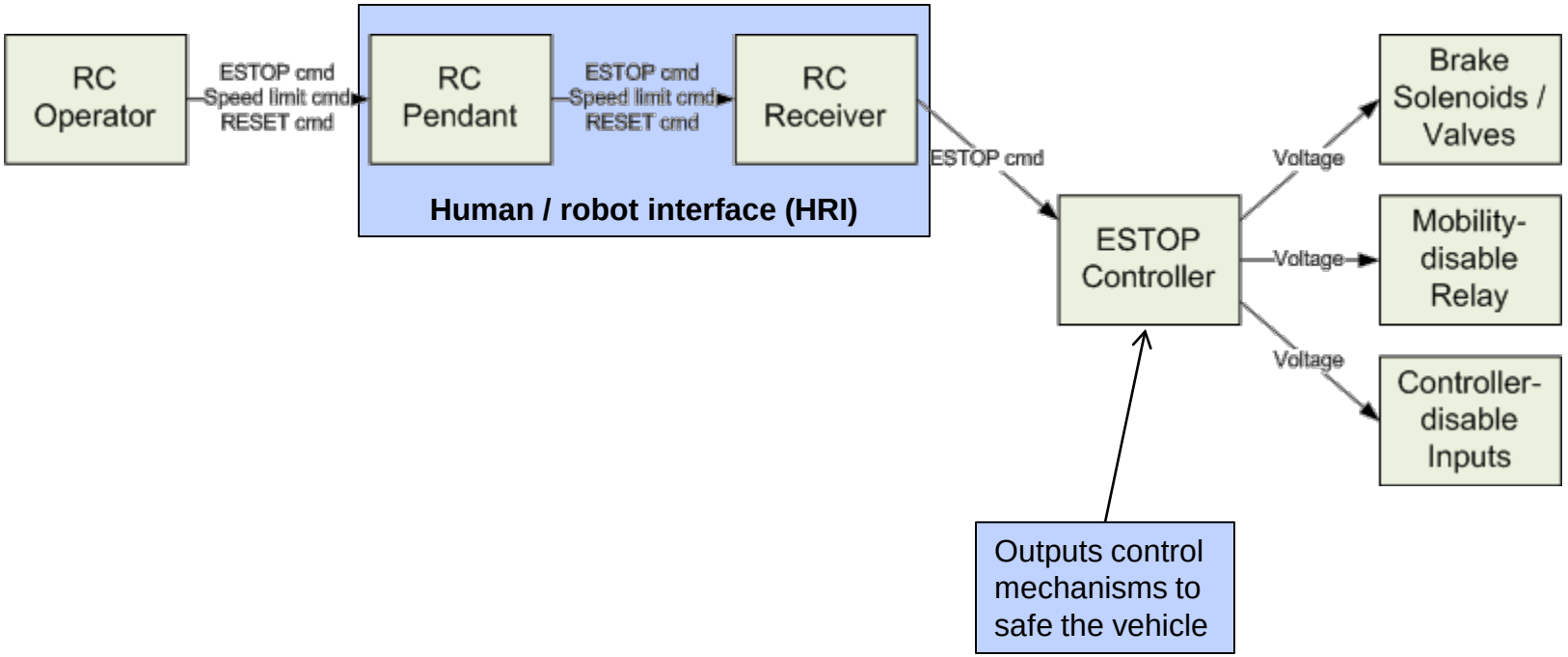


Development Process

1. Build fault model
2. Mitigate hazards with electromechanical safeguards
3. Identify remaining high-priority hazards
4. Define these hazards behaviorally to generate invariants
5. Design means of sensing invariant state
6. Design means of safing vehicle when invariant is violated
7. Build safety monitor
8. Test safety monitor to requirements

Update fault model with new detectors and safing components

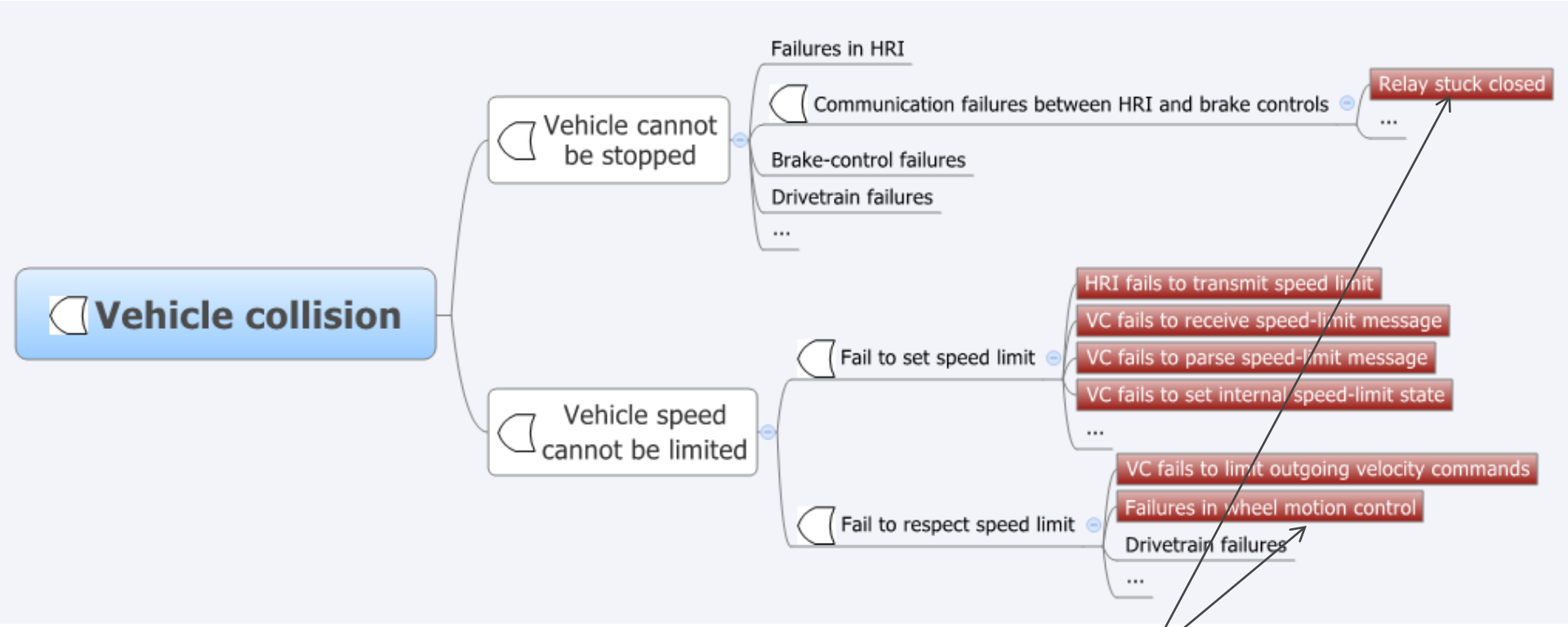
APD Safety Architecture



ACRONYM DECODER:

ESTOP	Emergency stop
RC	Radio controller

APD Fault Model Example



Red hazards are those not mitigated through hardware redundancy

ACRONYM DECODER:

HRI	Human/robot interface
VC	Vehicle controller software

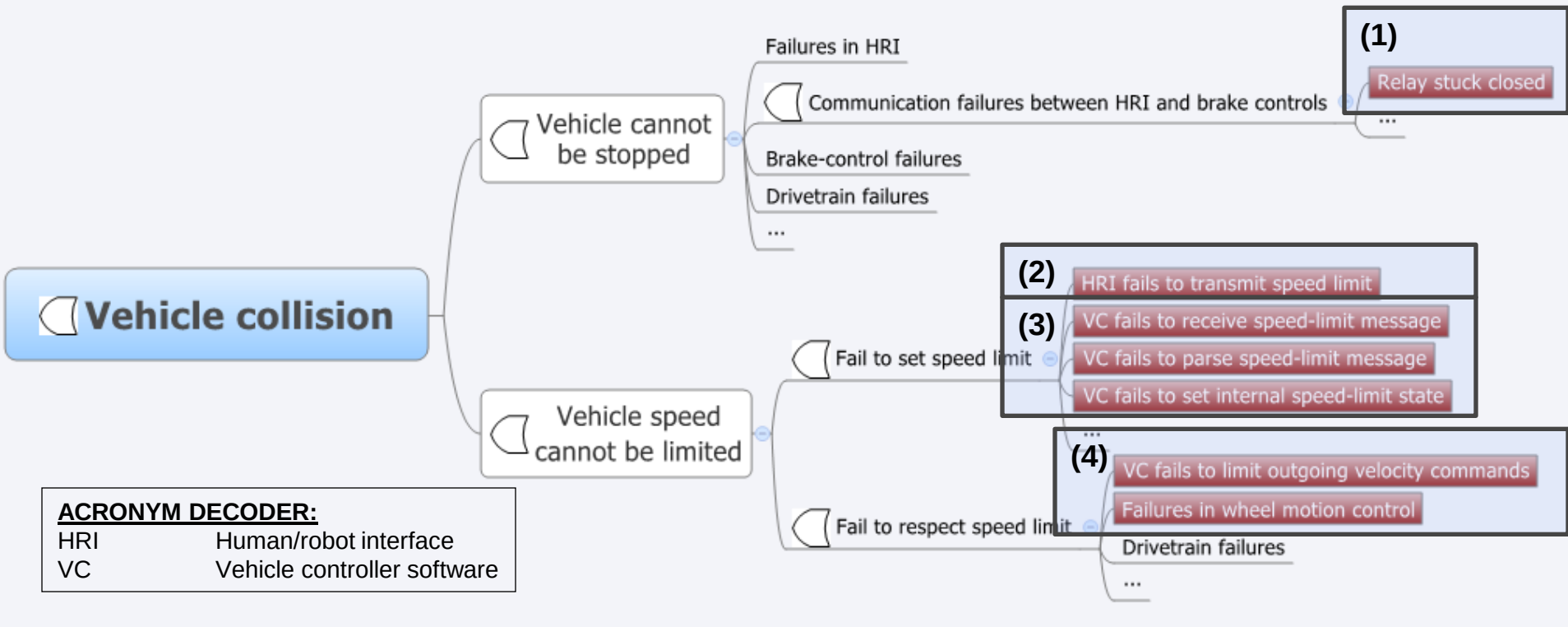
Hazard Behavioral Definition

Hazard	Behavioral Definition
(HRI) relay stuck closed	HRI reports ESTOP signal over serial line but relay is closed
Failures in HRI	No valid heartbeat from HRI
Communication failures between HRI and VC	No valid heartbeat from HRI No valid heartbeat from VC
VC fails to parse speed-limit message	Vehicle exceeds speed limit specified by HRI
VC fails to set internal speed-limit state	
VC fails to limit outgoing velocity commands	
Failures in wheel motion control	

APD Safety Invariants

Safe the vehicle if:

1. HRI ESTOP is commanded, OR
2. HRI is inactive, OR
3. VC is inactive, OR
4. Vehicle speed exceeds limit specified by HRI



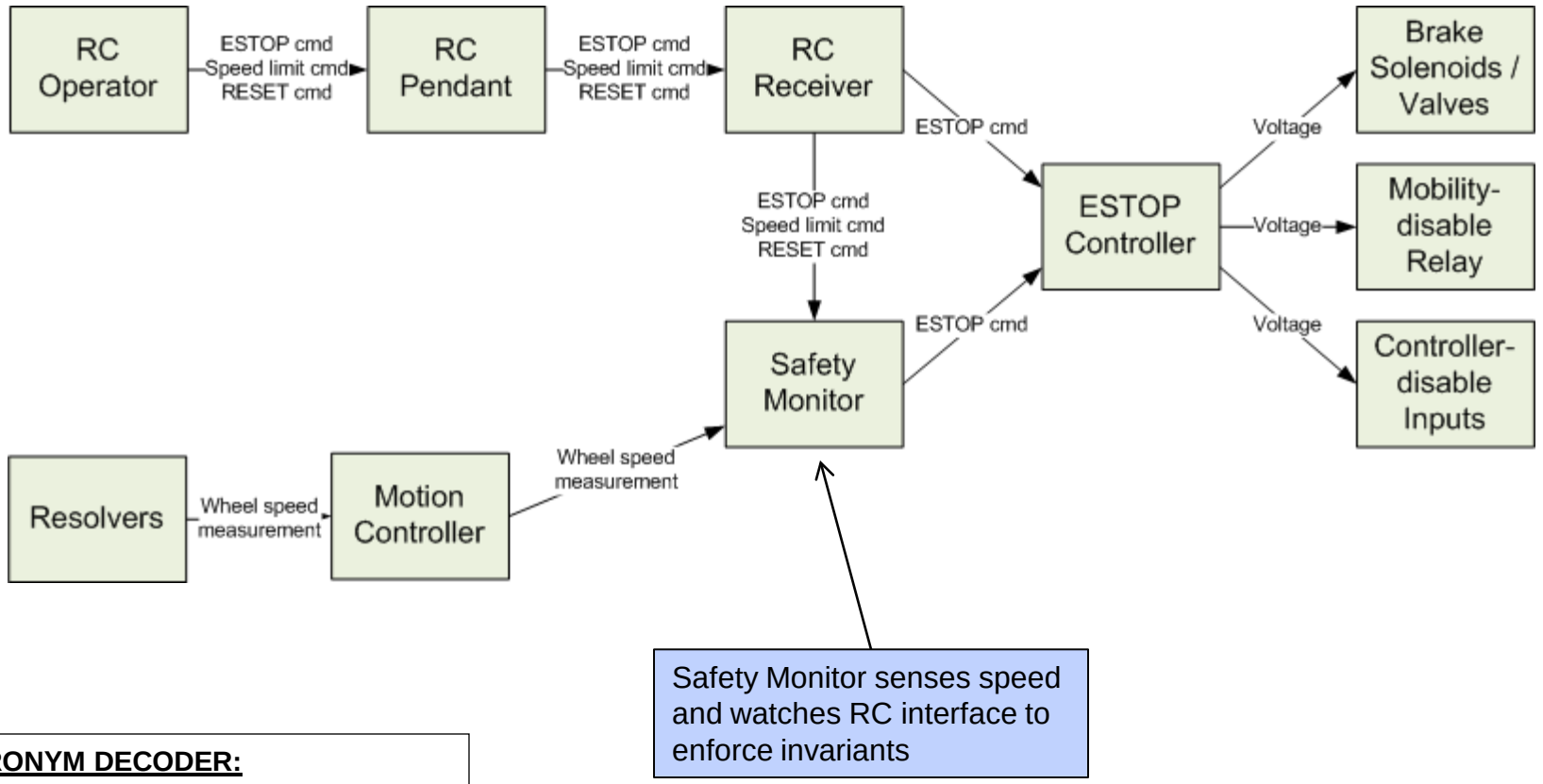
Means of Sensing Invariants

1. **HRI ESTOP command**
   - → Data packets received from HRI
   - → Packets include error-detection code
2. **HRI is inactive**
   - → Valid packet received from HRI
3. **VC is inactive**
   - → Valid driving command received sent by VC and snooped by Safety Monitor
4. **Vehicle speeds exceed limit specified by HRI**
   - → Wheel velocities are reported through telemetry from low-level traction drive controllers
   - → Data packets from HRI specify setting of a speed-limit switch

Means of Safing Vehicle

- **Must be…**
  - →**Independent of non-safety critical components**
  - →**Unable to be overridden or disabled**
  - →**Fail-safe**
- **On APD, an ESTOP-controller applies fail-safe mechanical brakes if any of a set of inputs drop low**
- **The safety monitor has control over one of these inputs**

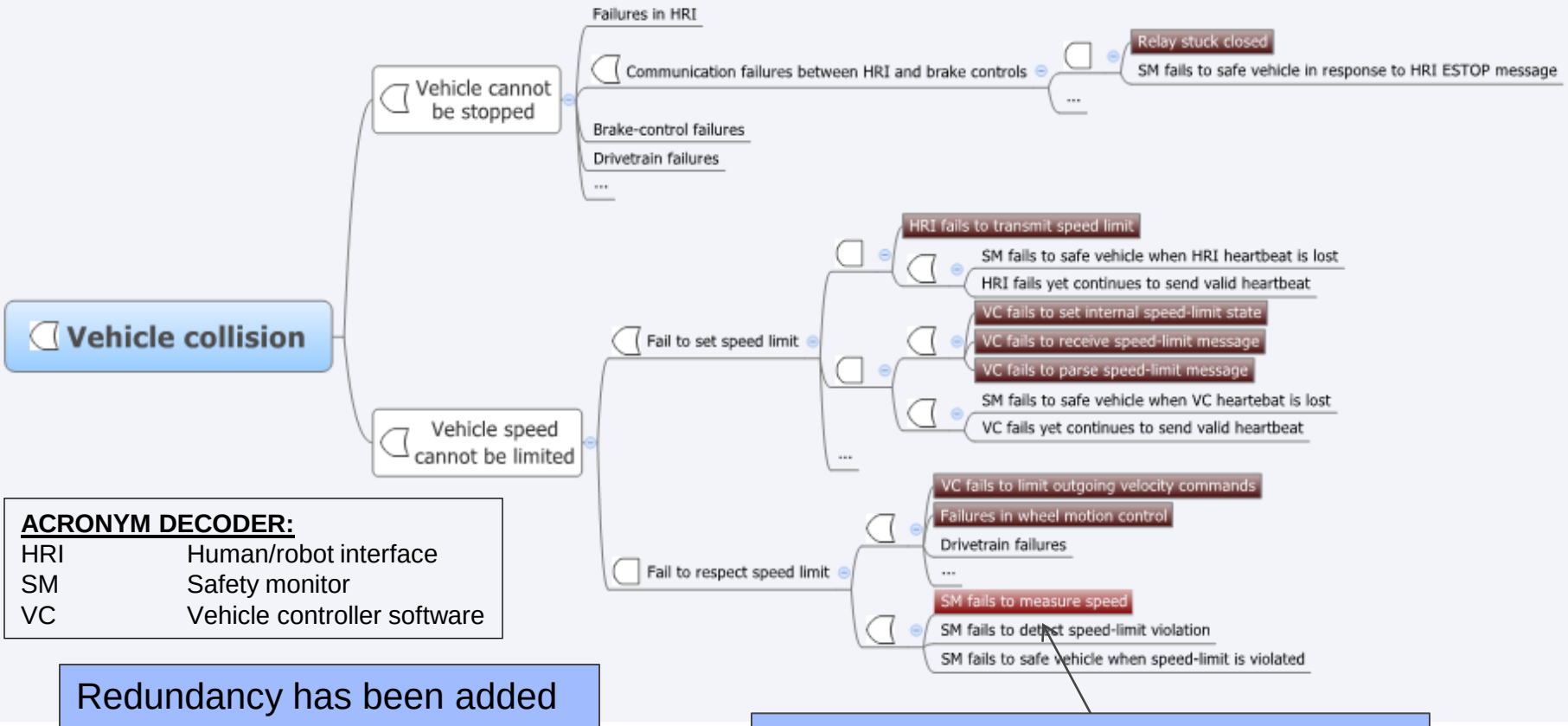
Updated Safety Architecture



ACRONYM DECODER:

ESTOP	Emergency stop
RC	Radio controller

Updated Fault Model



ACRONYM DECODER:

HRI	Human/robot interface
SM	Safety monitor
VC	Vehicle controller software

Redundancy has been added for previously-identified high-priority hazards

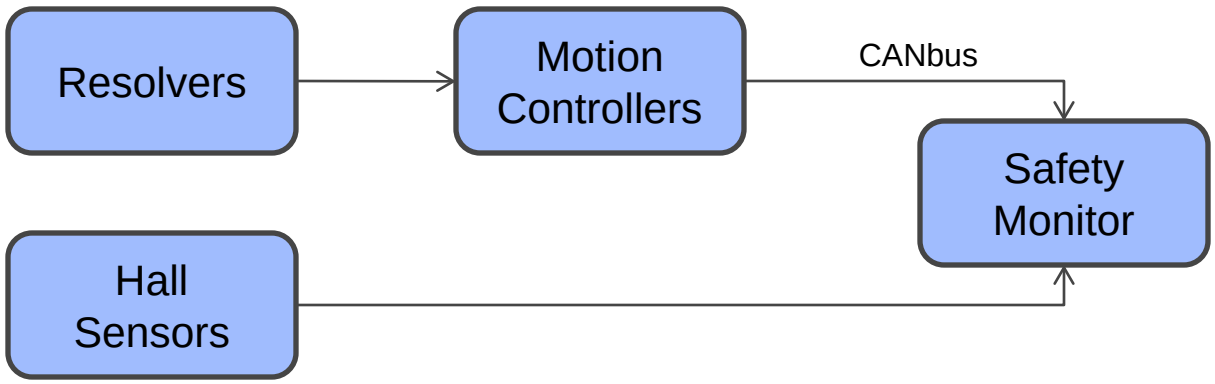
However, we now have another high-priority hazard ("SM fails to measure speed")

Updated Hazard Definitions

- **Speed is sensed through telemetry from motion-control hardware**
  - →**Vehicle speed is estimated as an average of wheel speeds**
- **These motion controllers are "black boxes" supplied by a vendor, so thorough V&V is infeasible**
  - →**Control hardware could report false readings**
  - →**Firmware changes could have unintended consequences**
  - →**Resolvers could fail**

Speed-sensing Safeguard

- **To address these risks we added redundant wheel-speed sensing**
- **Hall-effect sensors are placed in hubs that are wired directly to the safety monitor**
- **Use these sensors to check the validity of measurements from the motion controllers**



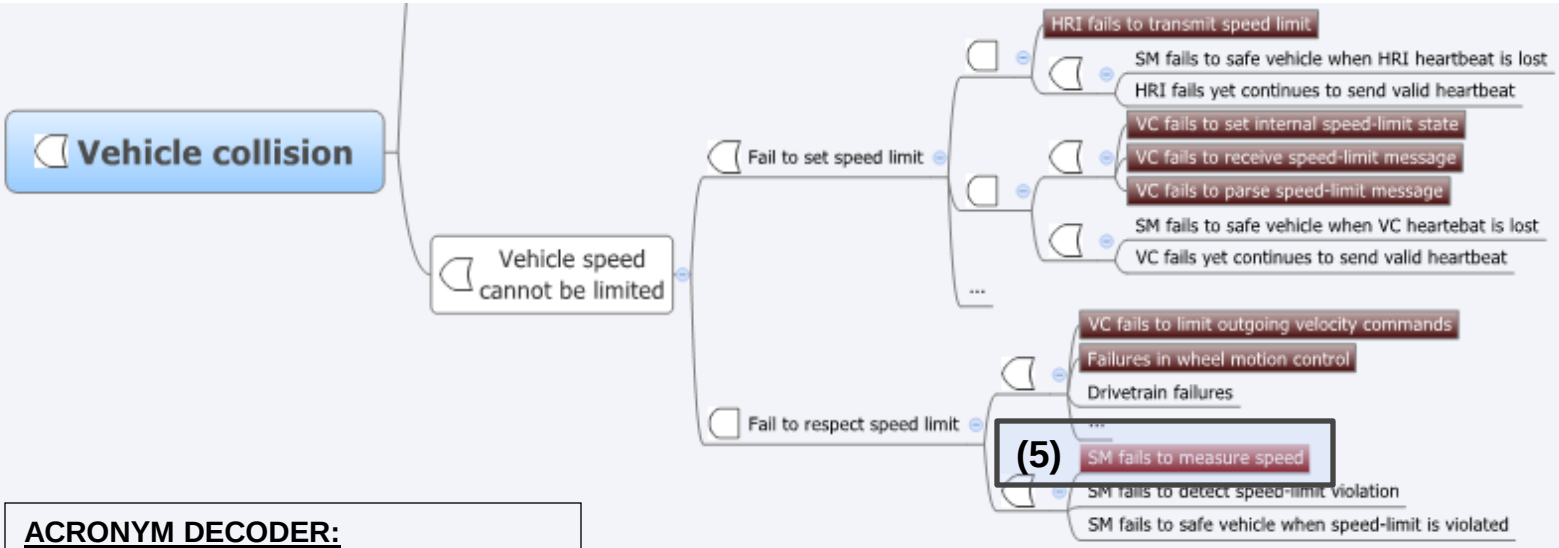
Disjoint Failure Modes

- **A failure of one sensing modality will not affect readings from the other:**
  - →**Largely separate power supplies**
  - →**Motion control firmware completely separate from hall sensors**
  - →**Motion controllers communicate via CAN bus, hall sensors use separate dedicated inputs**
  - →**Resolvers and hall sensors mounted in different locations**

Updated Safety Invariants

Safe the vehicle if:

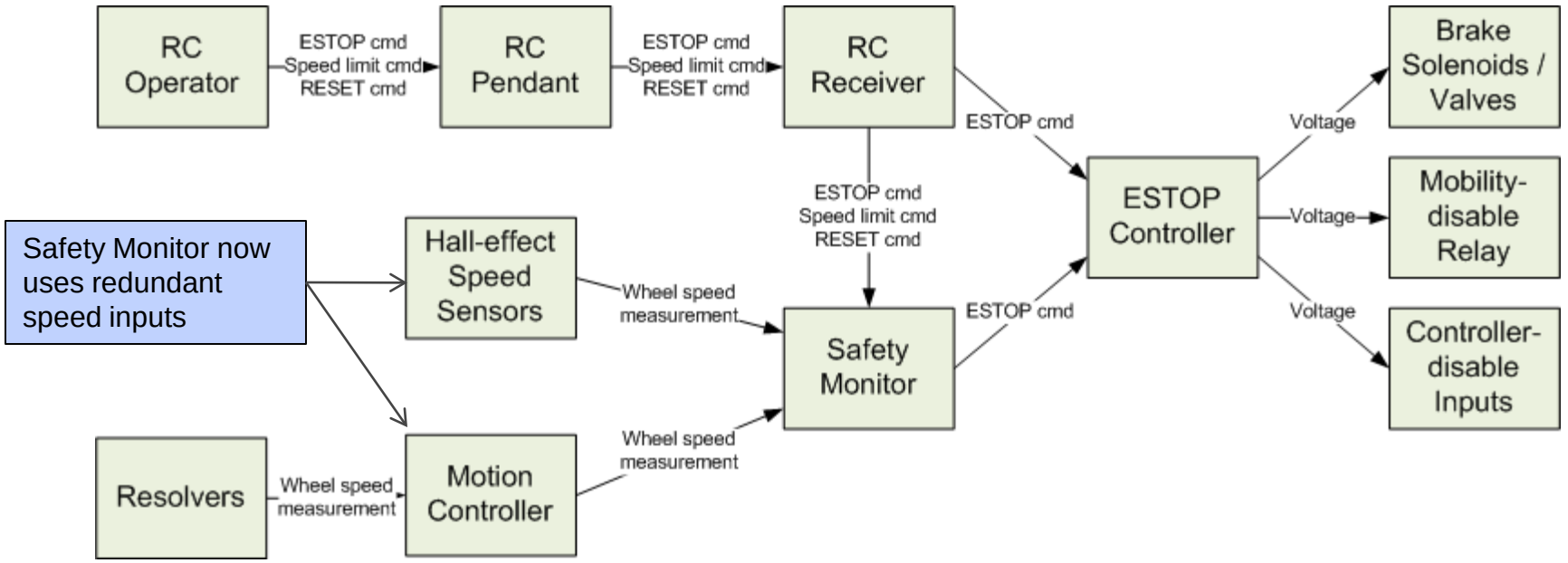
1. HRI ESTOP is commanded, OR
2. HRI is inactive, OR
3. VC is inactive, OR
4. Vehicle speed exceeds limit specified by HRI, OR
5. **Vehicle-speed measurements disagree**



ACRONYM DECODER:

HRI	Human/robot interface
SM	Safety Monitor
VC	Vehicle controller software

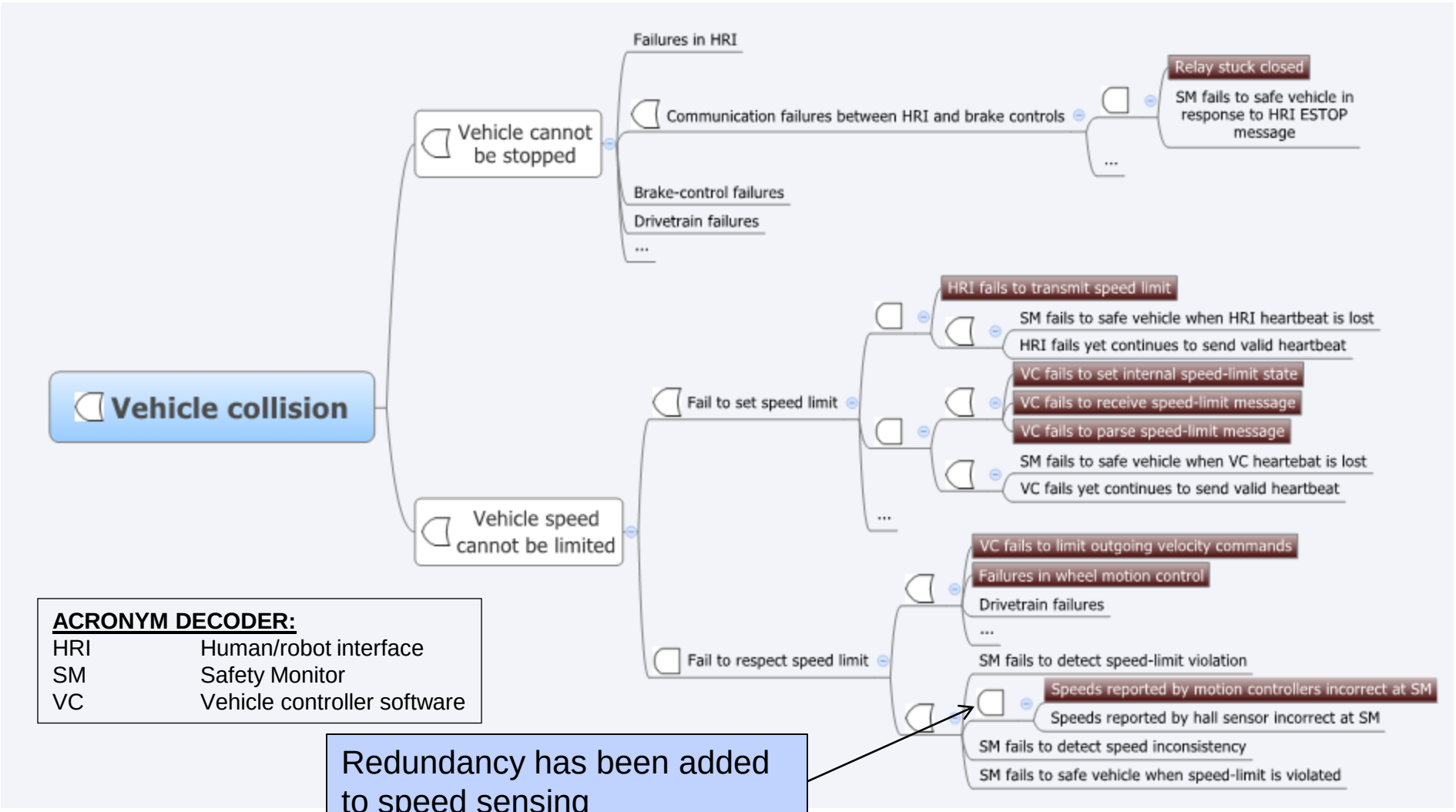
Final Architecture



ACRONYM DECODER:

ESTOP	Emergency stop
RC	Radio controller

Final Fault Model



ACRONYM DECODER:

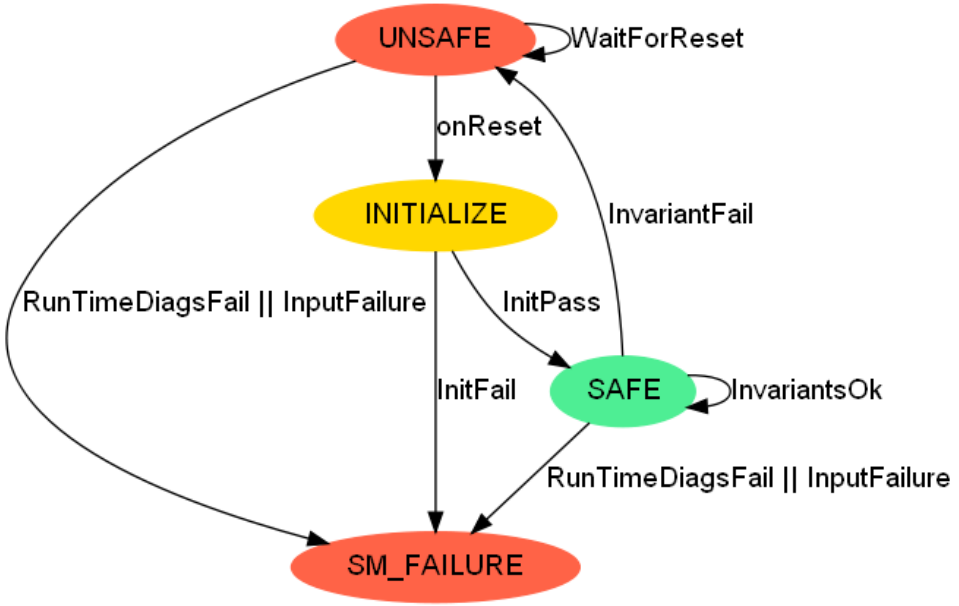
HRI	Human/robot interface
SM	Safety Monitor
VC	Vehicle controller software

Redundancy has been added to speed sensing

Safety Monitor Design

- **Simple finite state machine design**
- **If an invariant is violated, enter UNSAFE state and trigger ESTOP**
  - → **Return to SAFE state once invariants again hold and operator issues RESET**
- **If any self-checks fail, assume SM cannot evaluate invariants**
  - →**Enter SM_ASSERT state, which halts execution and triggers ESTOP with an independent hardware watchdog**

Safety Monitor Master State Chart



Safety Monitor Implementation

- **Implement as a single work loop**
  - →**Minimize use of interrupt I/O as much as possible**
- **Separate processing of input sources (e.g., conversion of hall-sensor readings to vehicle speed) from invariant evaluation**
- **Evaluate invariants based on simple boolean functions**

```
while (true)
{
  process_input_data()
  evaluate_invariants()
  update_SM_state()
  set_ESTOP_output()
  send_status_output()
}
```

Safety Monitor V&V Plan

- **The approach results in simpler test goals than we'd have if we had to verify a complex safety system**
  - → **80% of project resources are typically spent on V&V**
  - → **So streamlining V&V results in bigger payoffs than improving development tools**
- **Safety invariants are testable safety requirements**
- **For each invariant, carry out:**
  - → **System test that the SM issues an ESTOP if the invariant is violated**
  - → **Bench test that the SM issues an ESTOP if invalid input signals are received**
  - → **Unit test that the SM transitions to UNSAFE state upon any time-based combination of invariant-violation**
  - → **Code review that the processing of input data for the evaluation of invariants is correct**
- **Prove and Document that the means of safing the system is fail-safe**

References

1) J. Black and P. Koopman, "System safety as an emergent property in composite systems," DSN 2009

2) J. Black, “System Safety as an Emergent Property in Composite Systems”, doctoral dissertation, Electrical and Computer Engineering Department, Carnegie Mellon University, April 2009.

3) P. Koopman, J. Black, and T. Maximo, “Position Paper: Deeply Embedded Survivability”, ARO Planning Workshop on Embedded Systems and Network Security, Raleigh NC, February 22-23, 2007.

4) N. G. Leveson, T. J. Shimeall, J. L.Stolzy, and J. C. Thomas, “Design for safe software,” in Proceedings of the 21st AIAA Aerospace Sciences Meeting, Reno, NV, USA, Jan. 1983.

5) D. K. Peters and D. L. Parnas, “Requirements-based monitors for real-time systems,” IEEE Transactions on Software Engineering, vol. 28, no. 2, pp. 146–158, Feb. 2002.