

# Model Checking Dead Man Functionality in a Tram

Meine van der Meulen  
mjpm@adelard.com



# Outline

---

- The project
- The dead man functionality
- EN50126
- The safety case
- Experiences with FMEA, FTA and Model Checking
- Conclusion

# Rotterdam Citadis

---



# The Project

---

- Built by
  - Alstom Ridderkerk, NL: electronics, traction
  - Alstom La Rochelle, F: the rest
- For: Rotterdam Tram Company, RET
- This means splitting up the safety case



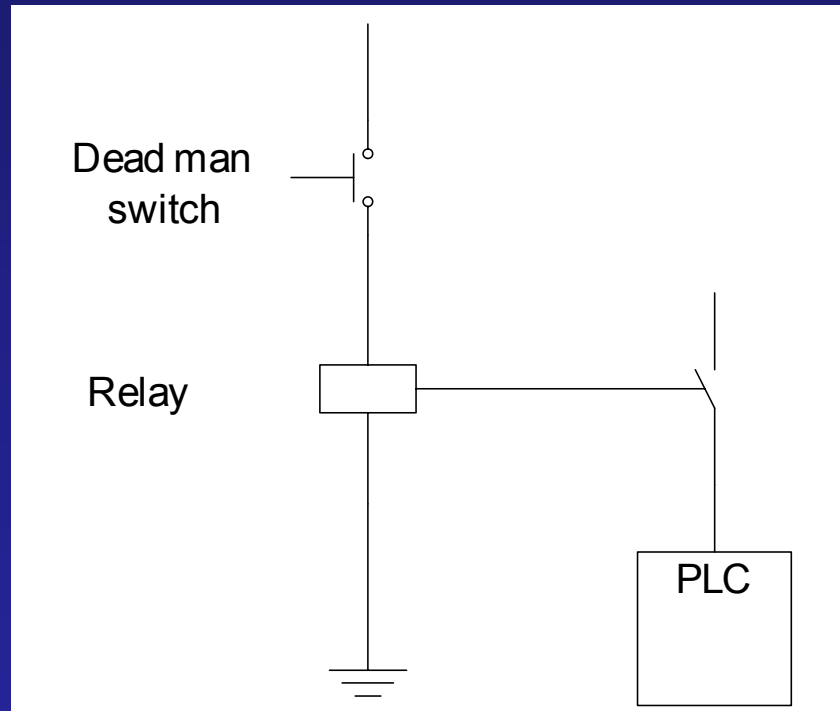
# The Dead Man Switch

---



# Dead man electronics

---

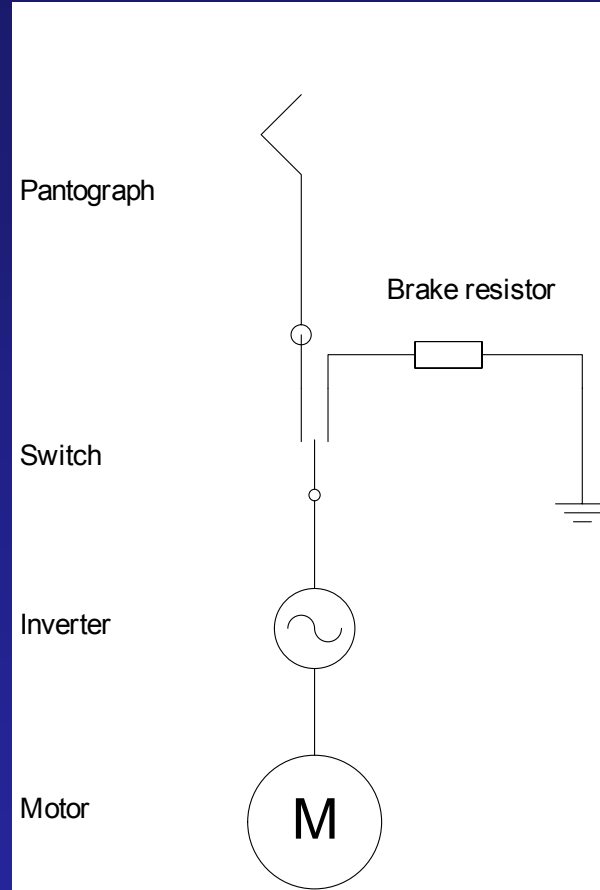


# Braking a Tram

---

- Recuperating brakes
- Brake discs
- Rail brakes

# Recuperating Brakes



# Pantograph

---



# Inverter

---



# Communication PLC to Inverter

---

- MVB – Vehicle Bus
- Vehicle Wires (fail-safe), e.g.:
  - Torque
  - No\_Emergency\_Brake
  - No\_Brake
  - Torque\_100%

# Dependability Requirements

---

- Apply EN50126: Railway applications; The specification and demonstration of dependability, reliability, availability, maintainability and safety (RAMS)
- Compare part I of IEC61508
- Almost no technical guidance
- Very usable

# Activities

---

- FMEA
- Fault Tree
- Model Checking Using SMV

# Observations from FMEA

---

- Large amount of work
- But necessary: gives safety engineer insight
- Leads to fruitful discussions with design team
- Some failure modes difficult to detect: acceptable?

# Observations from FTA

---

- Almost no redundancy: acceptable?
- Improvement by automatic testing
- Software is single point of failure
- Comparison to state-of-the-art

# Model Checking Using SMV

---

- SMV is very suitable for modelling PLC software
- Properties proved:
  - If dead man switch is released, then PLC will command braking action
  - If dead man switch is always closed, then PLC will command braking action

# Greater-than Translation

---

```
module gt_int(in1, in2, _out) {  
  INPUT in1, in2: -_max.._max;  
  OUTPUT _out: boolean;  
  case{  
    in1>in2: _out:=1;  
    default: _out:=0;  
  }  
}
```

# Off-timer Delay Translation

---

```
module tof(_in, q) {
  INPUT _in: boolean;
  OUTPUT q: boolean;
  state_tof: {idle, wait, active};
  init(state_tof):=idle;
  case{
    state_tof=idle & _in=0: next(state_tof):=wait;
    state_tof=wait & _in=0: next(state_tof):={wait, active};
    state_tof=active & _in=0: next(state_tof):=active;
    default: next(state_tof):=idle;
  }
  case{
    state_tof=idle: q:=1;
    state_tof=wait: q:=1;
    state_tof=active: q:=0;
  }
}
```

# Properties

---

```
deadman18_model: assert (  
  (F (IDeadmanIn & (X G ~IDeadmanIn)) &  
  F (NIDeadmanInMc1 &  
  (X G ~NIDeadmanInMc1)) &  
  (G MCabinActive) & G MCabinID))  
-> F G (~QEmerTorque1 & ~QEmerTorque2 & ~QNoBrake));  
using fair_tof_163_10, fair_tof_163_74 prove deadman18_model;
```

# Observations from Model Checking

---

- Identified no faults in software
- Modelling time is difficult
- Automatic conversion from PLC logic diagrams to SMV logic would help:
  - Is my rendering correct (especially after changes)?
  - Definitions of logic blocks?
- Design team understands and appreciates it

# Conclusion

---

- Model checking PLC gives evidence for safety claim
- Model checking work appreciated by client

