



Independence of Inhibits - Stored Command Sequences

2012 IV&V WORKSHOP

NASA IV&V - SMA SUPPORT OFFICE (SSO)

Chad Schaeffer (Chad.Schaeffer@TASC.com)

Ryan Schmidt (Ryan.Schmidt@TASC.com)



Agenda

- **Inhibit Overview**
- **Verifying Independence Within Stored Command Sequences**
- **Verification Rule Example**
- **Software to the Rescue**
- **GPM Example**
- **Conclusion**



Inhibit Overview

- **NASA uses inhibits to mitigate potential hazards**
 - From the NASA General Safety Program Requirements (NPR 8715.3C)
 - 1.7.3.1 a. Operations that require the control of a condition, event, signal, process, or item for which proper recognition, performance, or tolerance is essential to safe system operation, use, or function are designed such that an inadvertent or unauthorized event cannot occur (inhibit) (Requirement).
 - 1.7.3.1 b. Operations have three inhibits where loss of life can occur (Requirement).
 - 1.7.3.1 c. Operations have two inhibits where personal injury, illness, mission loss, or system loss or damage can occur (Requirement).
 - 1.7.3.1 d. The capability of inhibits or control procedures when required in operations by this paragraph are verified under operational conditions including the verification of independence among multiple inhibits (Requirement).
- **Inhibits must be independent and include a hardware component (not just software)**
 - From the NASA Expendable Launch Vehicle Payload Safety Requirements (NASA-STD-8719.24 Annex, Vol 3)
 - 3.2.6.1 Each design inhibit shall be independent of any other inhibit (i.e., loss or removal of one inhibit shall not result in the loss or removal of any other inhibit). Additionally, control of inhibits shall also be independent.
 - 3.2.7 Design inhibits shall consist of electrical and/or mechanical hardware.
 - 3.2.8 Operator controls shall not be considered a design inhibit. Operator controls are considered a control of an inhibit.



Inhibit Overview

- **Software often controls the inhibits using commands**
- **Stored Command Sequences (SCS) are a bundle of commands that can be initiated by a user or autonomously initiated by the software**
 - SCSes pose a risk to the independence of the inhibits
- **These commands and stored command sequences are likely stored in a configurable table that is updated many times through the software lifecycle**
- **These tables must be analyzed to ensure inhibit independence as soon as the software is attached to hardware (when hazards become possible)**

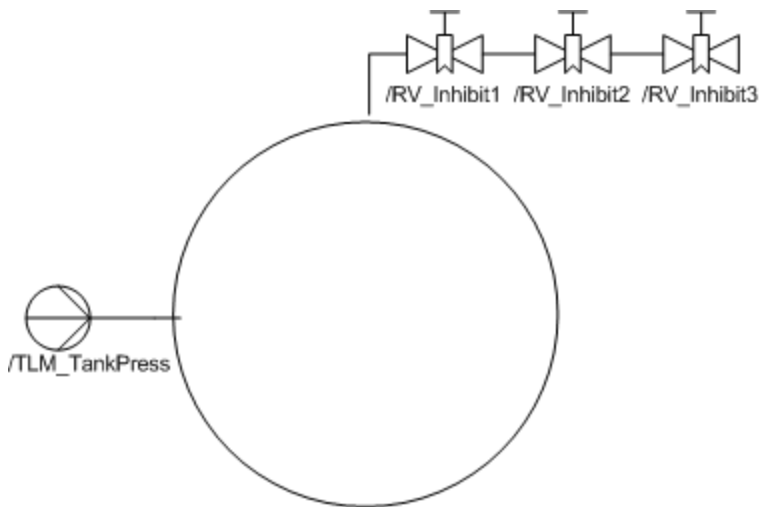


Verifying Independence Within Stored Command Sequences

- **Verifying inhibit independence can be time consuming**
 - The number of SCSes to analyze may be large (may be thousands)
 - Analysis must be repeated (performed for each table release used during mission phases where safety can be impacted (test, operation, decommission, etc.))
 - Verification rules may be tedious to manually perform (i.e., SCS that calls another SCS that calls another SCS...)
- **Verifying inhibit independence can be difficult**
 - Verification rules will depend on software architecture

Verification Rule Example

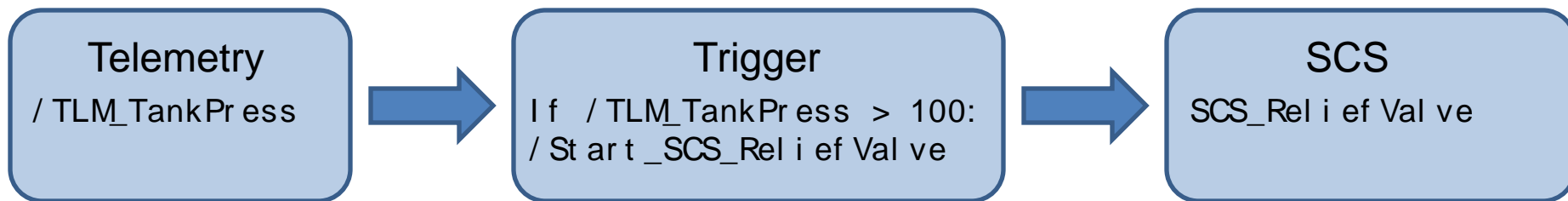
Example System



Inadvertent operation of the relief valve is mitigated by having three valves in series each controlled by a software command:

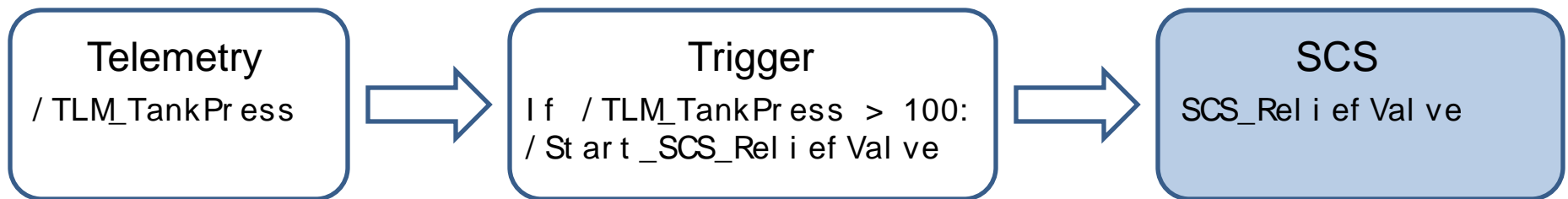
- `/RV_Inhibit1`
- `/RV_Inhibit2`
- `/RV_Inhibit3`

Example Software Architecture





Verification Rule Example




1) No SCS may remove more than one inhibit from a single hazard

SCS

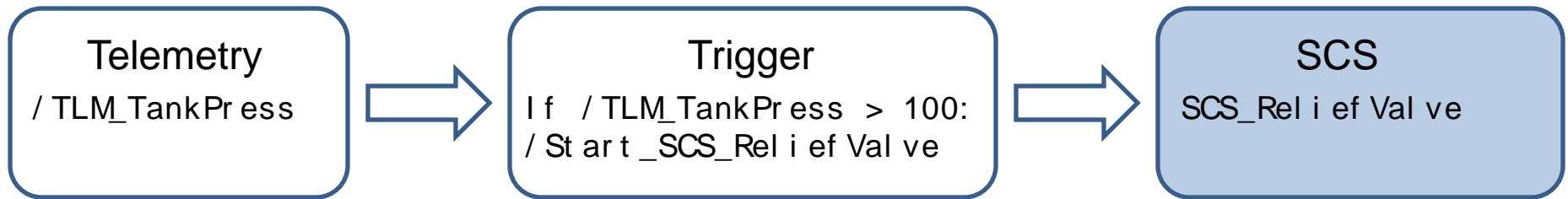
SCS_Relief Valve:

- / RV_inhibit 1
- / RV_inhibit 2

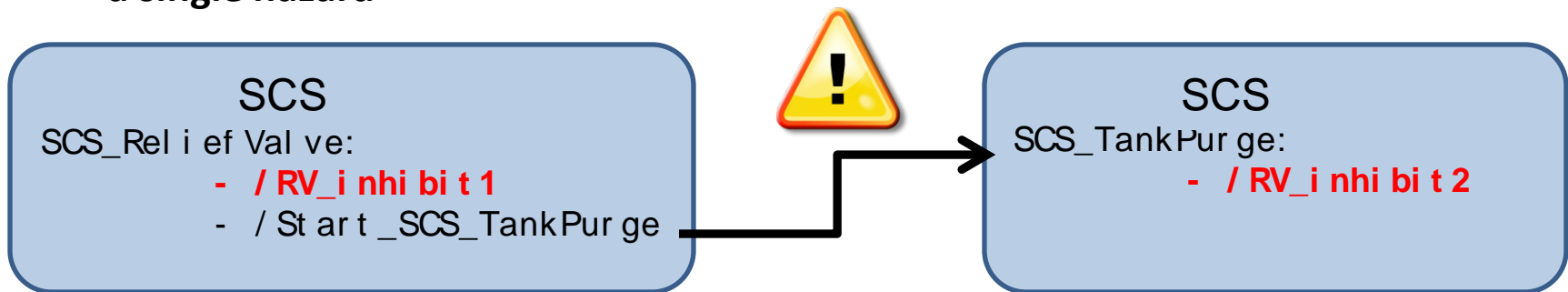




Verification Rule Example

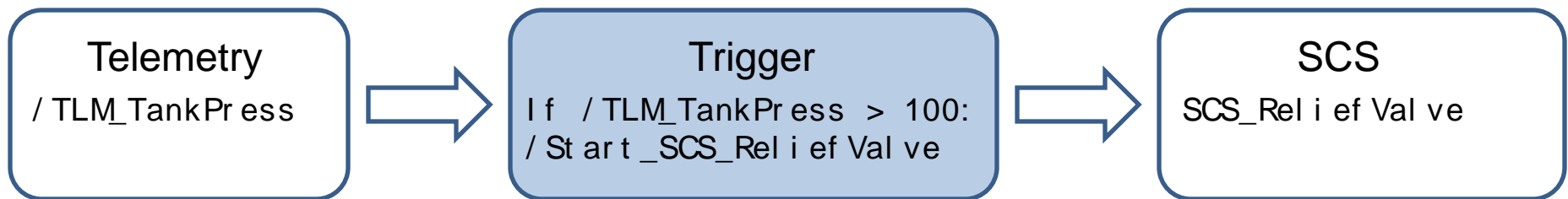


2) No SCS may call another SCS that ultimately removes more than one inhibit from a single hazard





Verification Rule Example

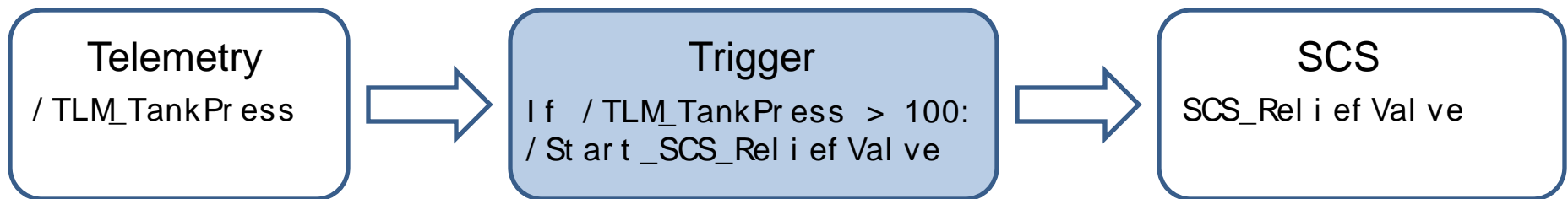


- 3) No trigger may initiate more than one SCS that ultimately removes more than one inhibit from a single hazard

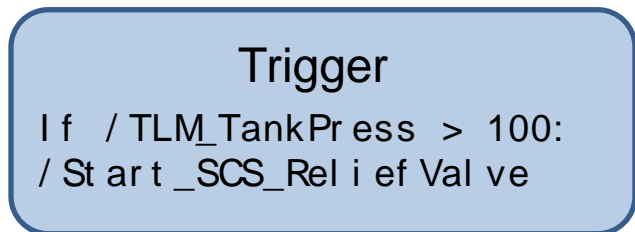




Verification Rule Example

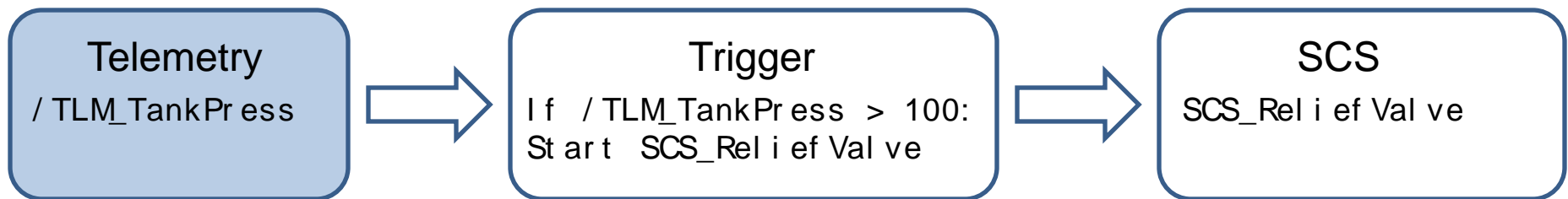


- 4) No trigger may initiate a SCS based on the state of another trigger or SCS that ultimately removes more than one inhibit from a single hazard

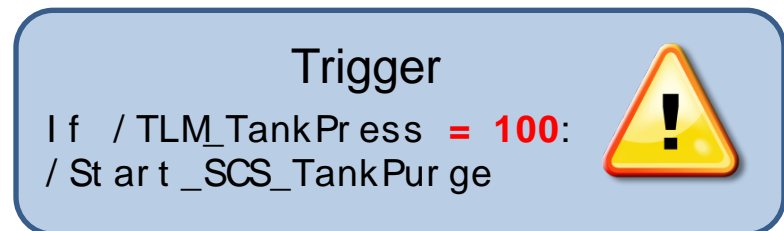
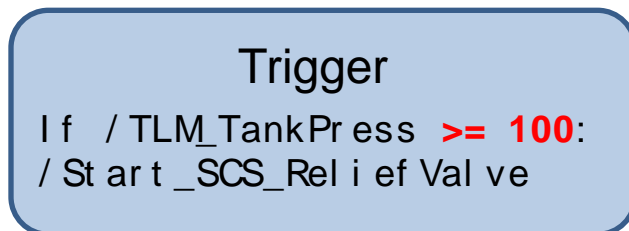




Verification Rule Example

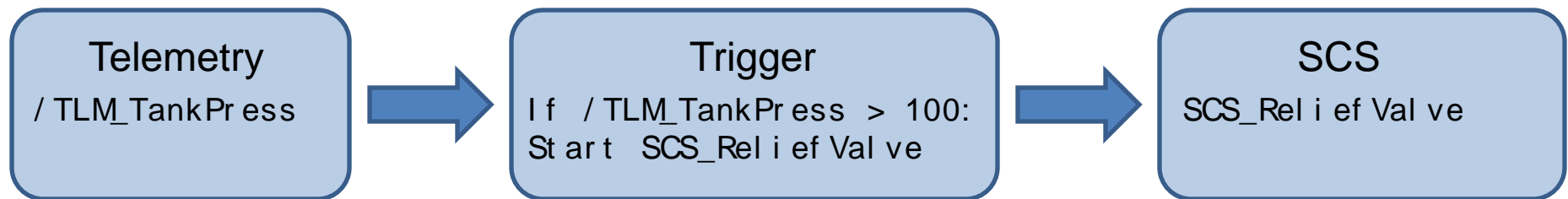


- 5) No single telemetry point can initiate more than one SCS that ultimately removes more than one inhibit from a single hazard





Verification Rule Example



- **This simple example produced five rules to be assessed on each SCS**
- **More complex architectures result in more rules with more complexity**
- **More hazards introduce more inhibit commands and SCSes to analyze**



Software to the Rescue

- **Software can assist inhibit independence verification**
- **Software can completely assess most verification rules**
- **For rules that are too complex to completely assess, software can eliminate/narrow the items that require manual analysis**
 - Advantages:
 - Dramatically reduces time for each analysis iteration
 - Less prone to human error
 - Disadvantages:
 - Must validate another piece of software



Global Precipitation Measurement (GPM) Example

- **GPM has 5 hazards with three inhibits controllable by software (~50 inhibit commands)**
- **Six inhibit independence verification rules were defined**
- **95 SCSes, 94 triggers, 153 telemetry Points, with an additional layer between telemetry and triggers**
- **A VBA macro within MS Excel was created**
 - ~900 LOC with comments
 - ~3 days to create
 - Fully assesses 4 of 6 rules, minimizes manual analysis on remaining 2 rules



Conclusion

- **It is important that inhibits (and inhibit controls) are truly independent**
- **It is a challenge to verify independence of inhibits that are controlled by stored command sequences**
 - Volatility of software (many releases to verify)
 - Number and complexity of independence “rules”
- **Software can also be the answer!**
 - Macros/scripts can automatically assess many independence “rules”