



# **Model Checking Verification and Validation at JPL and the NASA Fairmont IV&V Facility -**

**Frank Schneider, Jet Propulsion Laboratory, Steve  
Easterbrook, NASA IV&V Facility, Jack Callahan,  
Todd Montgomery, West Virginia University**

*Funded by NASA's Software Program, Office of  
Safety and Mission Assurance UPN #232-08-5L*

**Model Checking:** We use model checking to mean the process of (1) abstracting a partial specification from requirements and design elements for a reactive system and (2) applying reachability analysis to the resulting partial specification to validate that it has properties of interest. A reactive system is one that takes input from its environment at unpredictable times and responds according to a specific set of rules.

**The Problem and its solution:**

**Goal:** Show applicability for and to gain acceptance for use of model checking methodology in space craft development efforts in NASA spacecraft development projects

**Response:** In response to this goal a project manager made a design specification for a complex spacecraft (S/C)controller available to us

## **The Specification:**

- 1. Dual system with identical separate backup platform**
- 2. Communication of state information from one to the other**

**System - design purpose is two fold:**

- 1. Respond to and repair must-fix-external faults while**
- 2. Completing high priority sequence execution**

**Checkpointing scheme allows:**

- 1. Sequence execution to be frozen when fault occurs**
- 2. Fault to be repaired**
- 3. Rollback to start of last incomplete subsequence**
- 4. Resumption of sequence execution.**

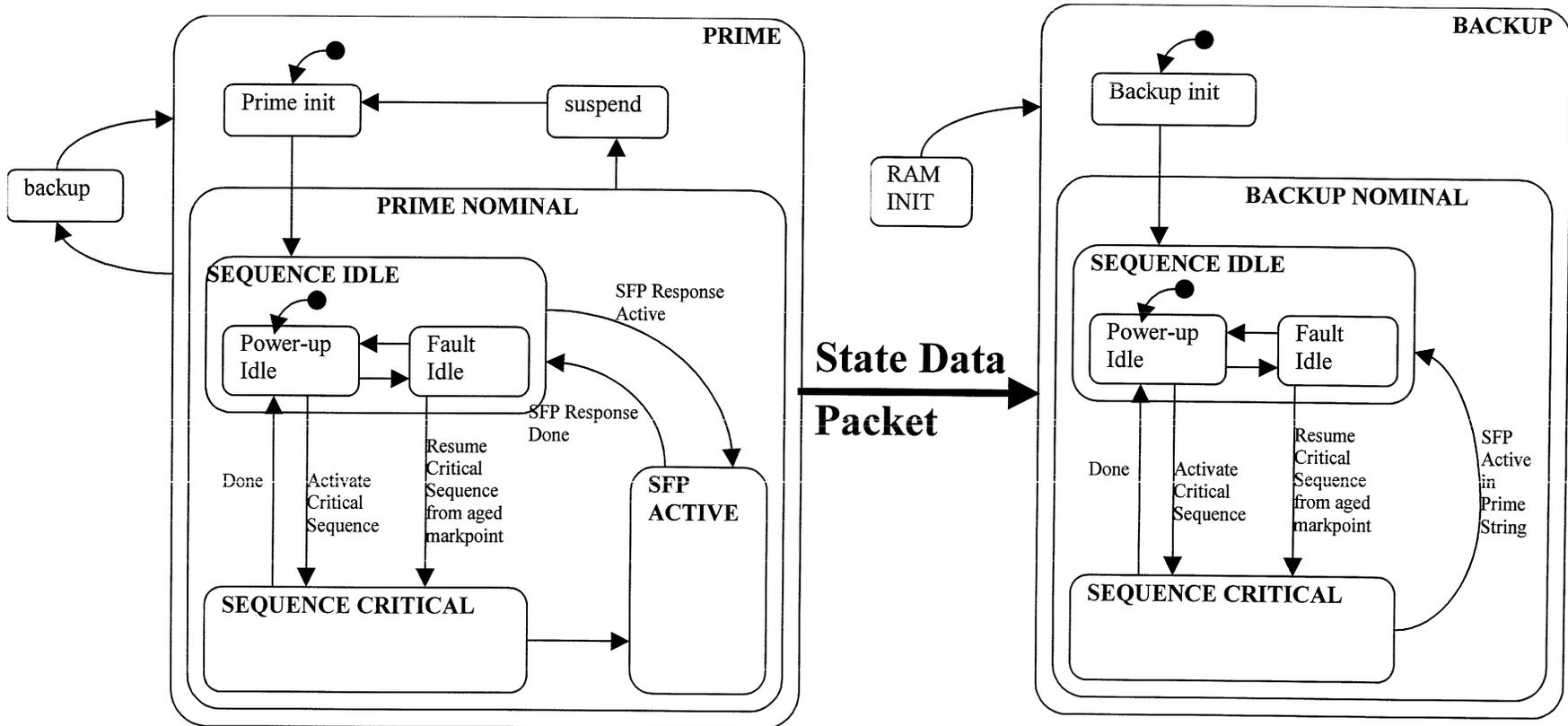
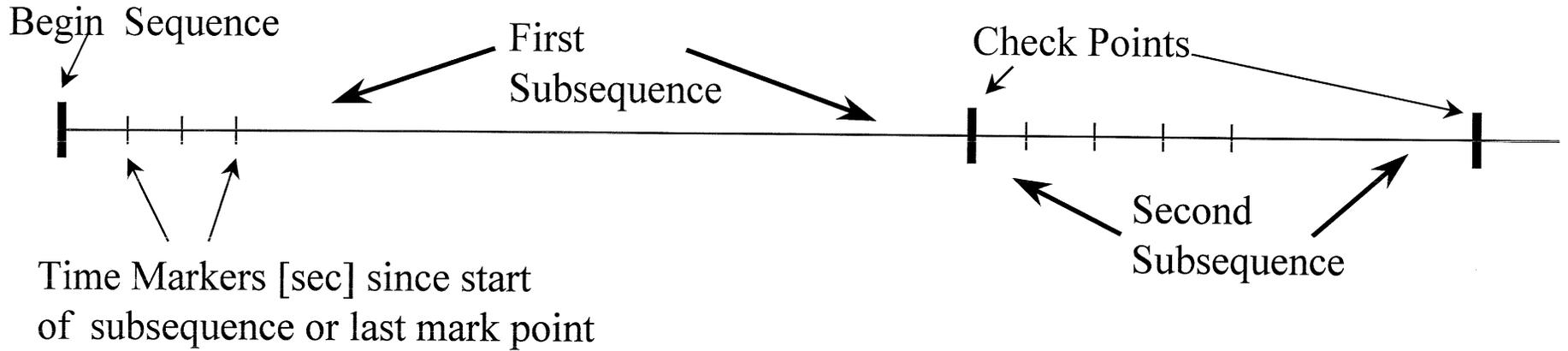
**Checkpointing scheme requires**

- 1. Three seconds of aging for completed subsequences**

**Overall redundancy allows entire controlling subsystem to fail:**

- 1. Failure is detected via communication mechanism**
- 2. Backup system becomes Control & repairs fault**
- 3. New Control system rolls back and resumes execution**

# Example: Sequence execution segment:



## **Design Validation:**

- **Original state space without design abstraction contained about  $2^{87}$  states.**
- **Partitioned requirements into 5 equivalence classes**
- **Validated 6 Checkpoint requirements**
- **Reduced state space to about about 130, 000 states**
- **Found 3 Anomalies using SPIN Model Checker [Gerard Holzman, Bell Labs, N.J.]**
- **Run Time was about 30 seconds for each anomaly:**
  1. **Repeated prime failure caused loss of synch with backup**
  2. **Prime failure immediately at end of sequence resulted in no rollback**
  3. **Prime failure at second 2 after check point gave invalid rollback point in backup system**

## **Implementation Validation:**

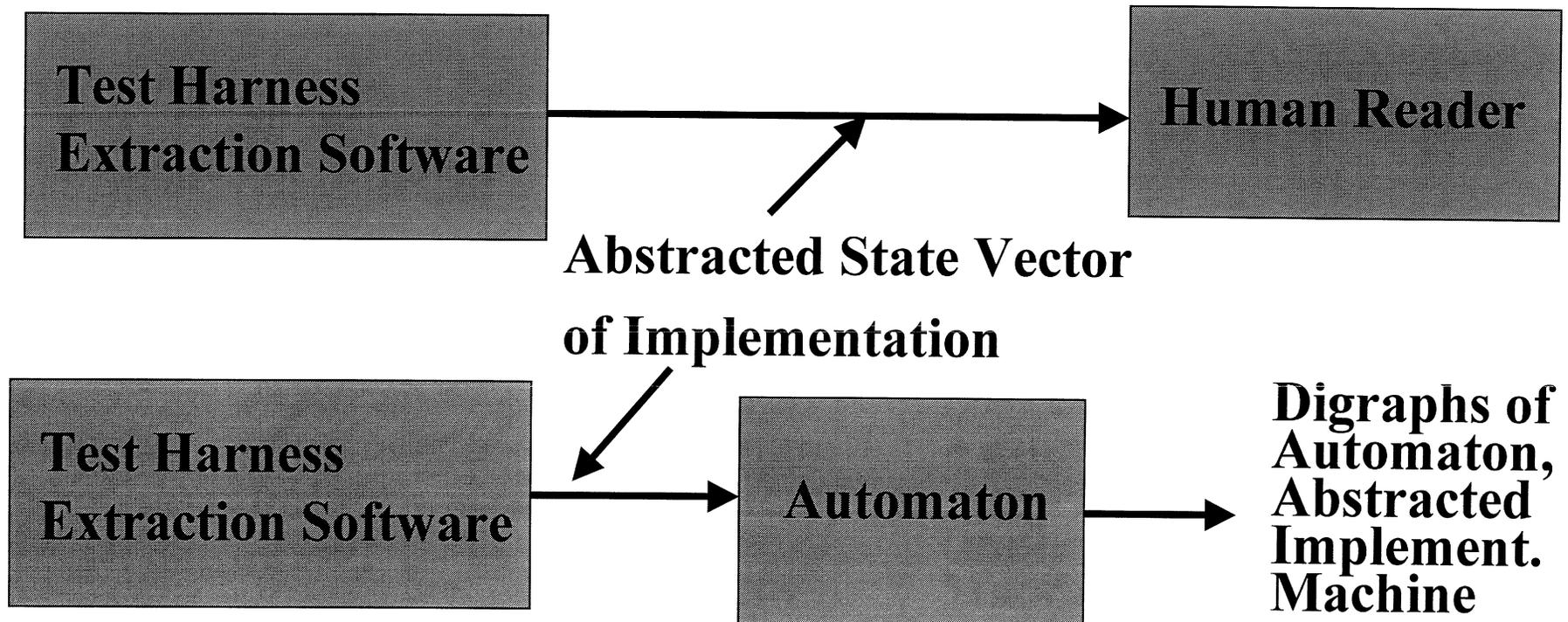
- 1. Used test harness for validation vehicle**
- 2. Language is Tool command Language (Tcl)**
- 3. Specify sequence to execute**
- 4. Inject design faults**
- 5. Select data structures to output for state vector sequence to examine**
- 6. Look for design faults in output software implementation state vector sequence**

## **Implementation Validation of Design Anomalies**

- 1. We did not see the first anomaly in the system - system engineers noted that all faults take at least several minutes to repair. Therefore , repair time was extended so that anomaly one would not be seen**
- 2. Our technique demonstrated that the second anomaly was in the implementation.**
- 3. Our technique demonstrated that the third anomaly was in the implementation.**
- 4. Project is addressing the issues of the three anomalies**

## Automation of Implementation:

1. Exactly as above but with an automaton replacing human scan of output state vector sequence
2. Automaton accepting state corresponds to counter example:  
If automaton is driven into accepting state requirement is violated.
3. Anomaly three was validated with automaton



## **Future work:**

- 1. Replace Test harness output by appropriately abstracted state vector from executing spacecraft application**
- 2. Develop counterexample requirements automaton for currently executing application**
- 3. Drive automaton from abstracted state vector in 1.**
- 4. Use accepting state condition of automaton as fault detection criterion to start fault recovery scheme.**

## **Benefits:**

- 1. Puts current fault detection and recovery schemes into analytic framework**
- 2. Counterexample automaton could be derived on-the-fly for autonomous spacecraft applications**

## **Tying it all Together**

**Having shown that we could**

- **Abstract a high level design from a specification for a spacecraft**
- **Validate the design using model checking**
- **Validate the implementation**

**We decided to incorporate the model checking process into our Quality Assurance processes:**

- **Currently there are about 21 processes.**
- **Process chart is flow chart that specifies logistics to follow in carrying out a quality assurance process for a quality process such a formal methods theorem proving, model checking, software inspection, and the like**
- **Process Chart is accompanied by written description**