Failure Reports

5/9/01  Lecture #34  16.070

• Examine failure reports due to software
  ➢ Mars Pathfinder
  ➢ Ariane V
  ➢ Mars Polar Lander
  ➢ Therac 25
Mars Pathfinder Priority Inversion

- A few days after landing, spacecraft began experiencing total system resets, each resulting in losses of data
- Pathfinder contained an "information bus" -- shared memory area used to pass info between tasks
  - Access to bus was synchronized with mutual exclusion locks (mutexes)
  - **Bus mgmt task:** frequent, high priority to move data in/out of info bus
  - **Meteorological data gathering task:** infrequent, low priority. Acquires info bus mutex, writes to bus, then releases mutex
  - If interrupt caused bus mgmt task to be scheduled while mutex was held, and if bus mgmt task then attempted to acquire same mutex, bus mgmt task is blocked, waiting until meteorological task releases mutex
  - Problem: Medium priority **communications task** (long-running) scheduled while high-priority bus mgmt task is blocked.
Flight #1 of the Ariane V - The Controller Design

- Flight Control System
  - On-Board Computer executes flight program and controls solid booster nozzles and Vulcain cryogenic engine, via servo-valves and hydraulic actuators
  - Inertial Reference System (SRI)
    - Measures launcher attitude and movements in space
    - Has internal computer which calculates angles and velocities based on info from a "strap-down" inertial platform, with laser gyros and accelerometers
    - Data from SRI transmitted through data-bus to On-Board Computer (OBC)
Ariane V Robustness

- Redundancy used to improve reliability
  - Two SRIs operating in parallel with identical hardware/software
  - One SRI active, one "hot" stand-by
  - If OBC detects that active SRI has failed, it switches to other SRI, if it is functioning properly
- Two OBCs
- Other Flight Control System units are also duplicated
Inheritance from Ariane IV

- SRI s of Ariane V essentially common to a system which flies on Ariane IV
  - Software which caused the interruption in SRI computers is used before launch
    - to align inertial reference system
    - In Ariane IV, to enable rapid re-alignment of system in case of late hold in countdown
  - Realignment function does not serve any purpose on Ariane V
  - Realignment function retained for commonality reasons and allowed to operate for ~40 seconds after liftoff.
Ariane V Explosion: Chain of Events

• Launcher started to disintegrate at L+39 sec due to high aerodynamic loads due to angle of attack >20° that led to separation of boosters from main stage, in turn triggering self-destruct

• Angle of attack caused by full nozzle deflection of solid boosters and Vulcain main engine

• Nozzle deflections commanded by OBC software based on data from active SRI-2. Part of these data showed diagnostic bit pattern of SRI-2 computer, which was interpreted as flight data

• SRI-2 sent diagnostic data because unit had declared a failure due to a software exception

• OBC could not switch to back-up SRI-1 because that unit had already ceased to function for same reason as SRI-2
Ariane V Explosion: Chain of Events (cont.)

• Internal SRI software exception
  ➢ Caused during execution of data conversion from 64-bit floating point to 16-bit signed integer value.
  ➢ Floating point number larger than what could be represented by 16-bit signed integer
  ➢ Resulted in Operand Error
  ➢ Data conversion instructions were not protected from Operand Error, although other conversions of comparable variables were protected

• Operand Error occurred due to unexpected high value of Horizontal Bias (BH), horizontal velocity sensed by platform.

• BH value much higher than expected because Ariane V trajectory differs from Ariane IV trajectory and results in considerably higher horizontal velocity values
Build-in Software Protection

- Why wasn't the Horizontal Bias variable conversion protected?
  - Maximum workload target of 80% set for SRI computer ➔ not all conversions could be protected
  - Analysis performed on every operation which could produce an exception to determine vulnerability of unprotected code
  - Seven variables were at risk of leading to Operand Error
  - Protection added to four of the seven variables, three left unprotected, including Horizontal Bias
    - Physically limited?
    - Large margin of safety?
Inappropriate Response to Software Error

• Spec: In the event of any kind of exception, SRI processor should be shut down
  - Shut-down is typical response for random hardware failures
  - In this case, caused shutdown of two healthy critical units
  - Not appropriate for software design errors! Loss of critical software function
  - What might be a more appropriate response?
    - Restart? Not feasible -- too difficult to recalculate attitude
    - Computer to provide best estimate of required attitude information
    - Prime and redundant units developed independently
Code that Caused Exception Was Not Required After Launch

- Operand Error occurred in software that performs alignment of strap-down inertial platform
  
  - Complex math filter functions align x-axis to gravity axis and find north from Earth rotation sensing
  - Assumes launcher is positioned at known, fixed position

- Software module computes meaningful results only before lift-off. After lift-off, this function serves no purpose

- Alignment function designed to continue after flight for Ariane IV to allow for count-down restart (after hold) without waiting for normal alignment (~45 minutes)

- Not required for Ariane V, but preserved for commonality reasons ("If it ain't broke, don't fix it")
  
  - Ran for ~40 seconds after lift-off
Conclusions and Recommendations

- Limitations of alignment software not fully analyzed and implications of allowing it to continue to function during flight not realized
- No software function should run during flight unless needed
- No sensor should stop sending data
- Identify all implicit assumptions made by the code; e.g., data ranges and restrictions on use of equipment
- Confine exceptions to tasks and devise backup capabilities
- Redefine critical components, taking software failures into account
Mars Polar Lander

• Plausible Failure Modes
  ➢ Premature shutdown of descent engines (compelling evidence)
  ➢ Surface conditions exceed landing design capabilities
  ➢ Loss of control due to dynamic effects
  ➢ Landing site not survivable
  ➢ Backshell/parachute contacts lander
  ➢ Loss of control due to center-of-mass offset
  ➢ Heatshield fails due to micrometeoroid impact

• No corroborating flight data to support finding (i.e., no telemetry during descent), so other failure modes cannot be ruled out
Premature Shutdown of Descent Engines

- Mag sensor provided in each of 3 landing legs to sense touchdown when lander contacts surface, initiating shutdown of descent engines
- Test data indicate spurious touchdown indication occurs in Hall Effect touchdown sensor during landing leg deployment
- Software logic accepts transient signal as valid touchdown event if it persists for two consecutive readings
- Tests indicate transient signals are long enough to be accepted as valid events
- Touchdown sensing logic enabled at 40 meters altitude
- At 40 meters, lander's velocity is ~13m/s, and accelerates to surface impact velocity of ~22m/s (nominal touchdown velocity = 2.4m/s)
Deployment Design

• Three landing legs deployed from stowed position to landed position at 1500 meters while lander is attached to parachute

• Each leg has Hall Effect magnetic sensor that generates voltage when leg contacts surface of Mars

• Descent engines shut down by flight software command when first landing leg senses touchdown
  
  ➢ Fault tolerant: If first leg fails to detect touchdown, second leg to touch will trigger engine shutdown
  
  ➢ Prevents lander from tipping over if attitude skewed relative to surface

• Engine must terminate within 50ms after touchdown to avoid overturning lander

• Flight software required to protect against premature touchdown signal or failed sensor in any leg
Software Design Logic

- Six variables used to identify and control landing
  - **Touchdown monitor** for Control: Started at 12 min prior to 40 meters
  - **Last Touchdown Indicator** for Sensing: previous reading (True=landed)
  - **Current Touchdown Indicator** for Sensing: current reading
  - **Event Enabled** for Control: Enabled at 40 meters to start checking sensors
  - **Indicator State** for Sensing: Touchdown declared when Last TI = Current TI = landed (True)
  - **Indicator Health** for Sensing: Identify Failed sensor if Last TI = Current TI = landed prior to 40 meters
Touchdown Monitor Start (TDM_Start)
Called once by command.

Data Variables Used:
- IndicatorHealth = (GOOD, FAILED)
- IndicatorState = (TRUE, FALSE)
- EventEnabled = (ENABLED, DISABLED)
- TouchdownMonitor = (STARTED, NOT-STARTED)

- IndicatorHealth = GOOD
- IndicatorState = FALSE
- EventEnabled = DISABLED
- TouchdownMonitor = STARTED

Touchdown Monitor Enable (TDM_Enable)
Called once by command.

Data Variables Used:
- TouchdownMonitor = (STARTED, NOT-STARTED)
- LastTouchdownIndicator = (TRUE, FALSE)
- CurrentTouchdownIndicator = (TRUE, FALSE)
- IndicatorState = (TRUE, FALSE)
- IndicatorHealth = (GOOD, FAILED)

- TouchdownMonitor = STARTED
- LastTouchdownIndicator = TRUE AND CurrentTouchdownIndicator = TRUE?
  - No
  - Yes

- IndicatorHealth = FALSE
- EventEnabled = ENABLED

Figure 7-8. Touchdown Monitor Functional Flow Diagram
Transients in Hall Effect Sensors

- Testing of lander leg deployments showed transients in Hall Effect Sensor output following initiation of leg deployment
- Mean duration of transient signal ranges from 5 - 33 msecs
  - Thermal Vac chamber test #1: 12, 26.5, 7.3 msec
  - Thermal Vac chamber test #2: 16, 12, 25 msec
- Sensor sampling rate is 10 msec
Process Assessment - Requirements Flowdown

- System Requirement defined to not use sensor data until reaching 40 meters altitude
  - Protect against premature descent engine thrust termination in event of failed sensor and possible transients
  - Requirement did not specifically state failure modes
  - Software designers did not include protection against transients, nor think they had to test for transients

- Omission in Software Requirements Specification meant failure to test that requirement in unit-level tests
**FLIGHT SOFTWARE REQUIREMENTS**

### Processing

a. The lander flight software shall cyclically check the state of each of the three touchdown sensors (one per leg) at 100 Hz during EDL.

b. The lander flight software shall be able to cyclically check the touchdown event state with or without touchdown event generation enabled.

Upon enabling touchdown event generation, the lander flight software shall attempt to detect failed sensors by marking the sensor as bad when the sensor indicates “touchdown state” on two consecutive reads.

The lander flight software shall generate the landing event based on two consecutive reads indicating touchdown from any one of the “good” touchdown sensors.

---

**SYSTEM REQUIREMENTS**

1) The touchdown sensors shall be sampled at 100-Hz rate.

   a. The sampling process shall be initiated prior to lander entry to keep processor demand constant.

   However, the use of the touchdown sensor data shall not begin until 12 meters above the surface.

2) Each of the 3 touchdown sensors shall be tested automatically and independently prior to use of the touchdown sensor data in the onboard logic.

   a. The test shall consist of two (2) sequential sensor readings showing the expected sensor status.

   b. If a sensor appears failed, it shall not be considered in the descent engine termination decision.

3) Touchdown determination shall be based on two sequential reads of a single sensor indicating touchdown.

---

*Figure 7-9. MPL System Requirements Mapping to Flight Software Requirements*
Lessons Learned

• All hardware inputs to software-decision logic must be identified. Test planning must have checklist that includes reqs to test logic in presence of transient/spurious signals

• Software test teams must examine every req to see whether there is a set of conditions that could cause the software to fail

• Systems engineering must stay on top of test results from all areas and communicate findings to other areas
• Summary/Review
  ➢ Mars Pathfinder Priority Inversion
  ➢ Ariane V software reuse
  ➢ Mars Polar Lander requirements flow-down
  ➢ Therac 25 race condition

• Exam Review
  ➢ Lecture and Recitation material since Spring Break: Weeks 9-13
  ➢ Readings:
    – C10.6-10.7, C11.4-11.6
    – R3, R6.1-6.6, R7, R11.1-11.4, R13
  ➢ Attend Recitation tomorrow for material review