High Technology Systems with Low Technology Failures

Some Experience with Rockets on Software Quality & Integration

Larry G. Craig
NASA Launch Services Program
Kennedy Space Center

February 10, 2010
Launch Vehicles (aka Rockets)
Background

John F. Kennedy Space Center

LAUNCH SERVICES PROGRAM

• What is a launch vehicle (rocket) and what is it supposed to do?
  – A system which provides a spacecraft with the correct velocity and position at the correct time to be in the desired orbit
  – It just keeps adding velocity by accelerating until it gets to the desired orbit velocity

• Where did rockets come from?
  – The Chinese invented gunpowder and put it in bamboo tubes for essentially fireworks for festivals in about the 1st century AD

• Why did this technology not advance very far until the 20th century AD?
  – Inertial Guidance which consists of Navigation, Guidance & Control
  – This is the major functionality difference which separates model rockets from real rockets
Model Rockets vs. Real Rockets

Model Rockets
$5.00 - $200.00

Real Rockets
$30 M - $500 M
What is Navigation, Guidance & Control

• Navigation – Where am I?
• Guidance – What is the path I want to take to get to where I want to go?
• Control – How do I keep on the path to get there?

• The invention of the accelerometer and gyroscopes allowed us to measure the value and direction of acceleration

• Numerically integrate acceleration and you get velocity
• Numerically integrate again and you get position
• If you know where you started you can guide to your desired end point

• BUT it took a computer to perform the numerical computations
Question #1
What do you get when you cross a computer with a battleship?

Question #2
What do you get when you cross a computer with a camera?

Question #3
What do you get when you cross a computer with an automobile

Question #4
What do you get when you cross a computer with a rocket?
• Anytime you introduce computer technology into an engineered system you inherit the failure modes of the computer including software

• What is a system?
  – A construct or collection of different elements that together produce results not obtainable by the elements alone
  – Rockets have system requirements to accomplish their function as a system

• What is quality?
  – Conformance to requirements (does it work at the system level?)
  – Sometimes confused with luxury which is the presence or absence of certain product requirements or features
History of Rocket Failure Causes

John F. Kennedy Space Center

LAUNCH SERVICES PROGRAM

• Early Flights
  – Design & Environment Errors

• More Recent Flights
  – Undetected processing errors (damaged systems)
  – Systems integration errors (systems do not play together like they should)
  – Collateral damage (propagated failures in complex systems)

• Some Rather Notable Failure Case Studies in Rocket Software Integration
  – Commercial Titan 3 Flight 2 (CT 2)
  – Ariane 5 Flight 1 (501)
  – Titan IV Centaur Flight 14 (TC 14)
Commercial Titan 3 Launch Vehicle
Commercial Titan 3 Flight 2 (CT 2)

- Commercial Version Modification of DoD/NASA Titan
  - Intended to compete in the commercial communication satellite market
  - Single and Dual Spacecraft Payload Configurations
- First Flight was a Dual Payload Configuration
  - Successfully Injected and Separated the Skynet 4A & JCSAT2 payloads on December 31, 1989
- Second Flight was a Single Payload Configuration
  - Failed to Separate the Single INTELSAT 603 Payload when commanded when launched on March 14, 1990
  - $265 M Loss (Uninsured $150 M Payload)
- Incorrect Software/Electrical Configuration when switching from Dual to Single Payload Configuration
  - Inadequate testing to detect the design error
- CT 3\textsuperscript{rd} flight with the same INTELSAT 6 payload was a success
Commercial Titan 3 Payload Configurations

John F. Kennedy Space Center

LAUNCH SERVICES PROGRAM

Dual Payload Configuration	Single Payload Configuration

Spacecraft 1

Spacecraft 2

Separation Command Wire Harnesses
The Rest of the INTELSAT 603 Story

STS - 49 Retrieval Mission
Ariane 5 Launch Vehicle
Ariane 501

- First Flight of a major upgrade from Ariane 4 to Ariane 5
  - Addition of large solid rocket boosters
  - Intended to capture a large share of the commercial communication satellite launch market
  - $7 Billion development program
  - Launched on June 4, 1996

- 37 seconds after engine ignition the vehicle abruptly changed attitude, broke apart due to aerodynamic forces and initiated the vehicle flight termination system
  - $370 M Loss

- Control system sent an unneeded major attitude change signal to the engine control actuators

- Guidance system sent erroneous data to the control system due to a shutdown of the inertial measurement unit (IMU) computer
Ariane 501 Flight

Liftoff!

37 seconds later !!!
What happened to Ariane 501?

- Data transmitted from the IMU was not proper flight measurement data but was a diagnostic bit pattern of the IMU computer which was interpreted as flight data
- The IMU computer had declared a failure due to a software exception
- The internal IMU software exception was caused during execution of a data conversion from 64 bit floating point to a 16 bit signed integer value
  - The floating point number which was converted had a value greater than what could be represented by a 16 bit signed integer
  - Related to a horizontal velocity measurement
  - Resulted in an Operand Error
- The data conversion instructions in Ada code were not protected from causing an Operand Error
What happened to Ariane 501? (cont)

- The exception handling mechanism for the IMU computer was
  - The failure was to be indicated on the databus
  - Failure context stored in EEPROM
  - Processor shut down

- Attempted to switch to a redundant IMU processor
  - Could not do so because the redundant IMU processor had failed during the previous data cycle for the same reason

- Error occurred during in a part of the software which is used for strap down inertial platform alignment
  - Provides meaningful results only prior to liftoff
  - After liftoff this function serves no purpose

- The alignment function was operative for 47 sec after liftoff
  - Time sequence was based on requirements for Ariane 4 and was not required for Ariane 5 (Reuse/commonality of software)
  - Used for rapid realignment of IMU on Ariane 4
What happened to Ariane 501? (cont)

- The Operand Error occurred due to an unexpected high value of a variable called horizontal bias
  - Related to sensed horizontal velocity
  - Horizontal velocity value for Ariane 5 was about 5 times the value for Ariane 4

- IMU System was not tested with the simulated Ariane 5 trajectory
  - IMU specification did not contain the Ariane 5 trajectory as a functional requirement
  - When they did so the failure was duplicated
  - IMU specifications did not indicate operational restrictions
• Titan IV Centaur is a US launch vehicle used for DoD & NASA missions

• Launched on April 30, 1999

• Upper stage (Centaur) tumbled out of control after spinning itself at a value an order of magnitude too high
  – Spacecraft placed in useless orbit
  – Due to an incorrect roll sensor gain software parameter value off by an order of magnitude (or one decimal place)
  – $1 Billion Payload Loss (Uninsured US Air Force payload)

• Roll is motion about the longitudinal axis of a rocket
  – Occurs while sitting on the earth due to earth rotation

• Incorrect software value could have been detected in prelaunch testing data analysis
  – In specification but out of family
Failure Mitigation Strategies

- **Practice Systems Engineering**
  - Know how everything works as a system
  - Have domain knowledge of functionality/criticality
  - In a control system everything matters (sensors, computation, actuators and their data)
  - Software has imbedded assumptions in its logic
- **Think about what can go wrong**
  - Success is eliminating/mitigating causes of failure
  - Maybe use some formal failure analysis techniques
    - FMECA, Event Sequence Diagrams
  - Think about how to make systems robust
  - Study technology history and learn from others mistakes
- **Devise simple sanity tests/data analysis to eliminate errors**
  - Think functionality
  - Analyze the data
  - Isolate one function at a time
Failure Mitigation Strategies (cont)

- **Testing**
  - Tests to prove no possibility of a negative function
  - Graceful degradation/failure handling
  - Try to break the software

- **Progressive levels of integration testing**
  - Elements may work by themselves but not together

- **End to End and Integrated Testing**
  - Test Like You Fly
  - Fly Like You Test

- **Random Hardware Failures are Rare**
  - Design, Integration, Testing & Data Analysis Failures are NOT