Bugs in the Space Program:
The Role of Software in Systems Failure

Prof. Steve Easterbrook

Dept of Computer Science,
University of Toronto

http://www.cs.toronto.edu/~sme

Key ideas

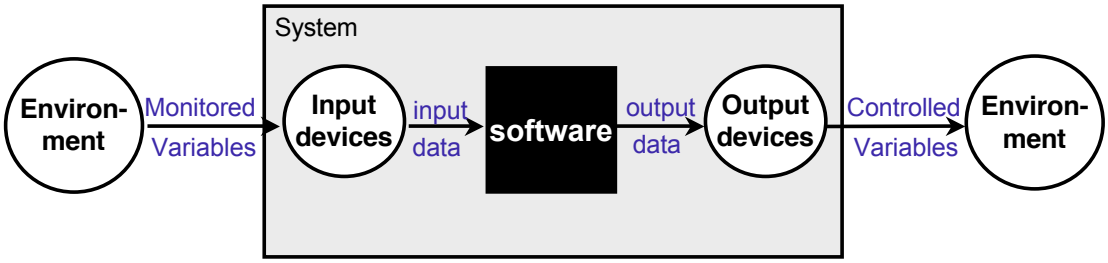
Complex systems rely more and more on software as the "glue"

→ Is software just like any other component?

→ Are there additional risks when we choose to allocate system functions to software?

→ Are systems engineering and software engineering similar disciplines?

How Software Engineers see the world

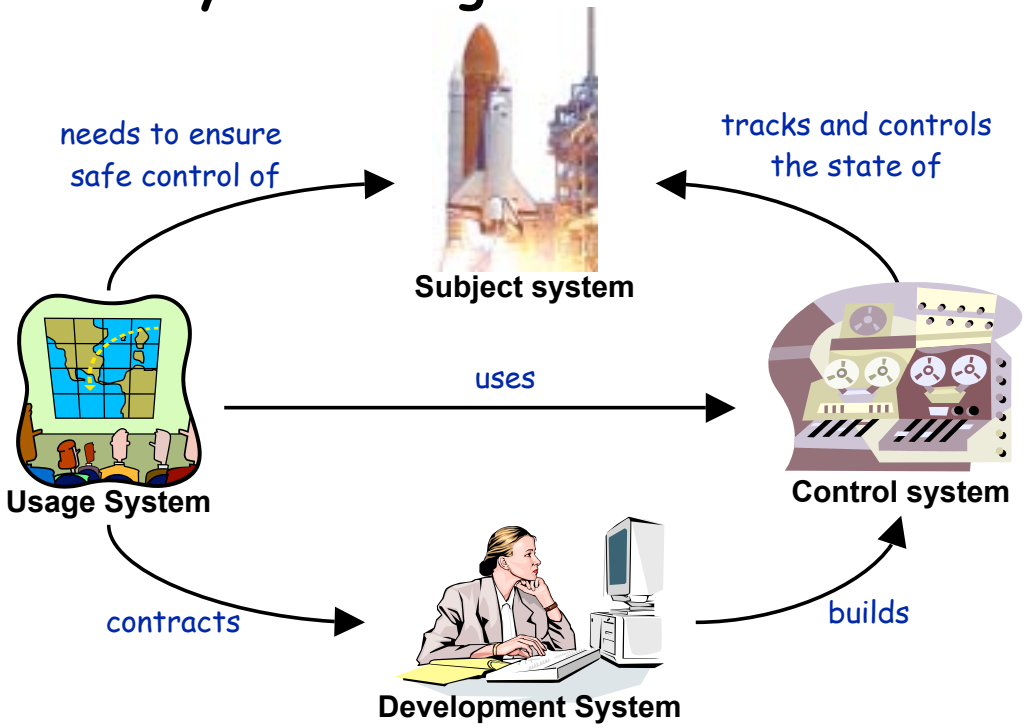


Specifications state the software functions in terms of inputs & outputs

Requirements express a desired relationship between monitored and controlled variables

Domain Properties constrain how the environment can behave

How Systems Engineers see the world

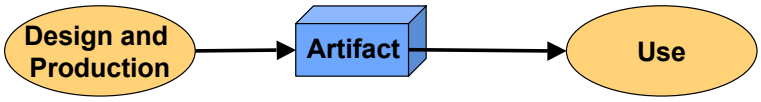


Why is software special?

- → Software is invisible, intangible, abstract
  - ↳ Software alone is useless - its purpose is to configure some hardware to do something
- → Software doesn't obey the laws of physics
  - ↳ Behaviour explained by discrete math, rather than continuous math
- → Software has no repeated components
  - ↳ Hence more complex for its "size" than other designed artifacts
- → Software never wears out
  - ↳ …statistical reliability measures don't apply
- → Software can be replicated perfectly
  - ↳ …no manufacturing variability
- → Software is not manufactured
  - ↳ …so can be re-designed even after deployment

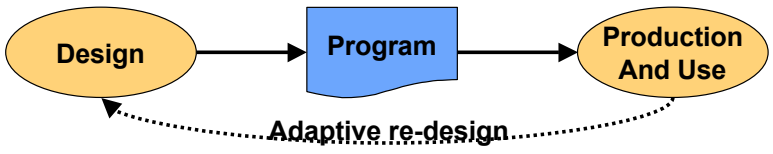
Software is not manufactured

Pre-industrial design:



Industrial design:

Software Design:



Mariner I

- Launched
  - 22 July 1962
- Mission
  - Venus Fly-by
- Fate:
  - Veered off course during launch
  - Destroyed by the Range Safety Officer 293 seconds after launch
- Cause:
  - "Missing hyphen in computer code"
- But not to worry…
  - Mariner II was an identical copy, developed as a backup
  - launched 27 August 1962
  - Successfully completed the mission

Lessons?

"Plan to throw one away - because you will anyway"
--Fred Brooks, The Mythical Man-Month

Ariane-5 flight 501

- Background
  - European Space Agency's reusable launch vehicle
  - Ariane-4 a major success
  - Ariane-5 developed for larger payloads
- Launched
  - 4 June 1996
- Mission
  - $500 million payload to put in orbit
- Fate:
  - Veered off course during launch
  - Self-destructed 40 seconds after launch
- Cause:
  - Unhandled floating point exception in Ada code

Ariane-5 Events

- Locus of error:
  - Platform alignment software (part of the Inertial Reference System, SRI)
  - This software only produces meaningful results prior to launch
  - Still operational for 40 seconds after launch
- Cause of error:
  - Unhandled Ada exception Converting 64-bit floating point to 16-bit signed integer
  - Requirements state that computer should shut down if unhandled exception occurs
- Launch+30s: Inertial Reference Systems fail
  - Backup SRI shuts down first
  - Active SRI shuts down 50ms later for same reason
- Launch+31s: On-Board Computer receives data from active SRI
  - Diagnostic bit pattern interpreted as flight data
  - OBC commands full nozzle deflections
  - Rocket veers off course
- Launch+33s: Launcher starts to disintegrate
  - Self-destruct triggered

Why did this failure occur?

- → Why was Platform Alignment still active after launch?
  - SRI Software reused from Ariane-4
  - 40 sec delay introduced in case of a hold between -9s and -5s
- → Why was there no exception handler?
  - An attempt to reduce processor workload to below 80%
  - Analysis for Ariane-4 indicated the overflow not physically possible
- → Why wasn't the design modified for Ariane-5?
  - Not considered wise to change software that worked well on Ariane-4
- → Why did the SRIs shut down in response?
  - Assumed faults caused by random hardware errors, hence should switch to backup
- → Why was the error not caught in unit testing?
  - No trajectory data for Ariane-5 was provided in the reqts for SRIs
- → Why was the error not caught in integration testing?
  - Full integration testing considered too difficult/expensive
  - SRIs were considered to be fully certified
  - Integration testing used simulations of the SRIs
- → Why was the error not caught by inspection?
  - The implementation assumptions weren't documented
- → Why did the OBC use diagnostic data as flight data?

Lessons?

Test what you fly, fly what you test
(...and test to the Operational Profile)

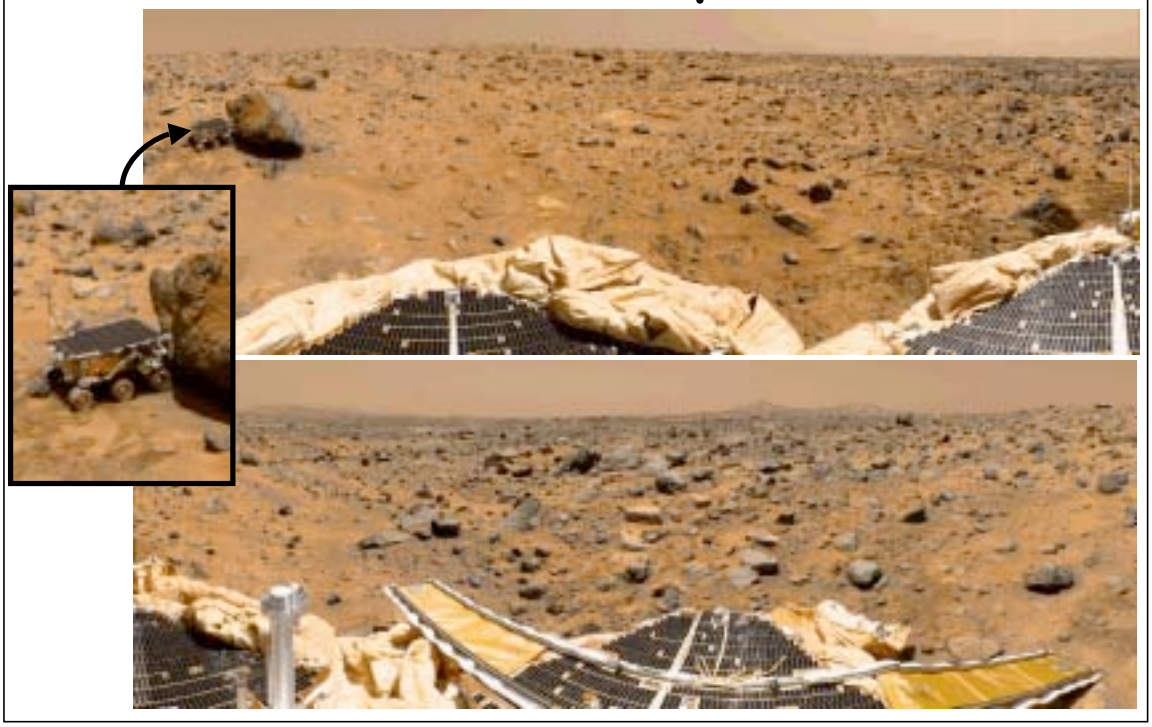
Software Redundancy Doesn't Work

Software Reuse is very Risky

Mars Pathfinder

- Mission
  - Demonstrate new landing techniques
    - parachute and airbags
  - Take pictures
  - Analyze soil samples
  - Demonstrate mobile robot technology
- Major success on all fronts
  - Returned 2.3 billion bits of information
  - 16,500 images from the Lander
  - 550 images from the Rover
  - 15 chemical analyses of rocks & soil
  - Lots of weather data
  - Both Lander and Rover outlived their design life
  - Broke all records for number of hits on a website!!!

Remember these pictures?



Pathfinder had Software Errors

→ Symptoms
- Software started doing total system resets
- … soon after Pathfinder started collecting meteorological data

→ Cause
- 3 Process threads, with bus access via mutual exclusion locks (mutexs):
  - High priority: Information Bus Manager
  - Low priority: Meteorological Data Gathering Task
  - Medium priority: Communications Task
- Priority Inversion - Bus Manager locked out
- Eventually a watchdog timer notices Bus Manager hasn't run for some time…

→ Factors
- Very hard to diagnose; Hard to reproduce
- Was experienced a couple of times in pre-flight testing
- Testers assumed it was a hardware glitch

Lessons?

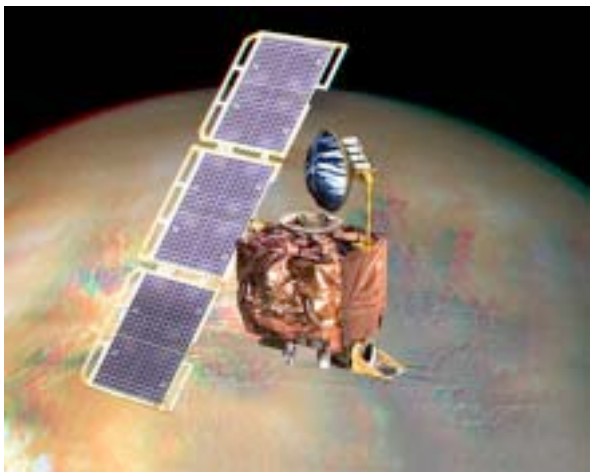
Sometimes you can get away with it

If it doesn't behave how you expect, it's not safe
(the exception proofs the rule)

Some software bugs are really butterflies
(see: the butterfly effect)

Mars Climate Orbiter

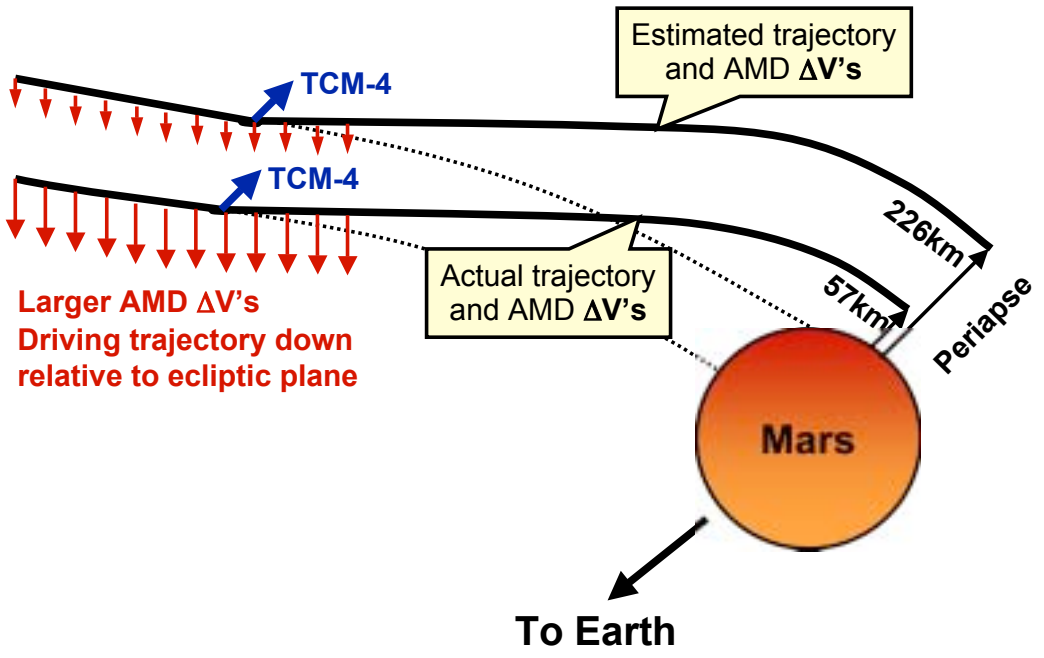
- Launched
  - 11 Dec 1998
- Mission
  - interplanetary weather satellite
  - communications relay for Mars Polar Lander
- Fate:
  - Arrived 23 Sept 1999
  - No signal received after initial orbit insertion
- Cause:
  - Faulty navigation data caused by failure to convert imperial to metric units



MCO Events

- Locus of error
  - Ground software file called "Small Forces" gives thruster performance data
  - This data is used to process telemetry from the spacecraft
  - Angular Momentum Desaturation (AMD) maneuver effects underestimated by factor of 4.45
- Cause of error
  - Small Forces Data given in Pounds-seconds (lbf-s)
  - The specification called for Newton-seconds (N-s)
- Result of error
  - As spacecraft approaches orbit insertion, trajectory is corrected
    - Aimed for periapse of 226km on first orbit
  - Estimates were adjusted as the spacecraft approached orbit insertion:
    - 1 week prior: first periapse estimated at 150-170km
    - 1 hour prior: this was down to 110km
    - Minimum periapse considered survivable is 85km
  - MCO entered Mars occultation 49 seconds earlier than predicted
    - Signal was never regained after the predicted 21 minute occultation
    - Subsequent analysis estimates first periapse of 57km

MCO Navigation Error



Contributing Factors

- → For 4 months, AMD data not used due to file format errors
  - Navigators calculated data by hand
  - File format fixed by April 1999
  - Anomalies in the computed trajectory became apparent almost immediately
- → Limited ability to investigate:
  - Thrust effects measured along line of sight using doppler shift
  - AMD thrusts are mainly perpendicular to line of sight
- → Poor communication between teams
  - Navigation team not involved in key design decisions
  - Navigation team did not report the anomalies in the issue tracking system
- → Inadequate staffing
  - Operations team monitoring 3 missions simultaneously (MGS, MCO and MPL)
- → Operations Navigation team unfamiliar with spacecraft
  - Different team from development & test
  - Did not fully understand significance of the anomalies
  - Surprised that AMD was performed 10-14 times more than expected
- → Inadequate Testing
  - Software Interface Spec not used during unit test of small forces software
  - End-to-end test of ground software was never completed
  - Ground software considered less critical
- → Inadequate Reviews
  - Key personnel missing from critical design reviews
- → Inadquate margins…

Lessons?

If it doesn't behave how you expect, it's not safe
(yes, really!)

If your teams don't coordinate,
neither will their software
(See: Conway's Law)

With software, everything is connected
to everything else -- every subsystem is critical

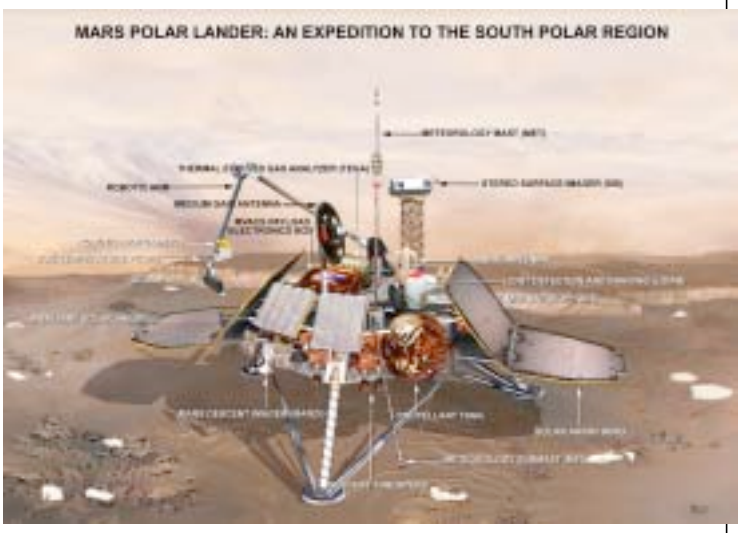
Sidetrack: SNAFU principle

Full communication is only possible among peers;
Subordinates are too routinely rewarded for telling
pleasant lies, rather than the truth.

Not a good idea to have the
IV&V teams reporting to the program office!!

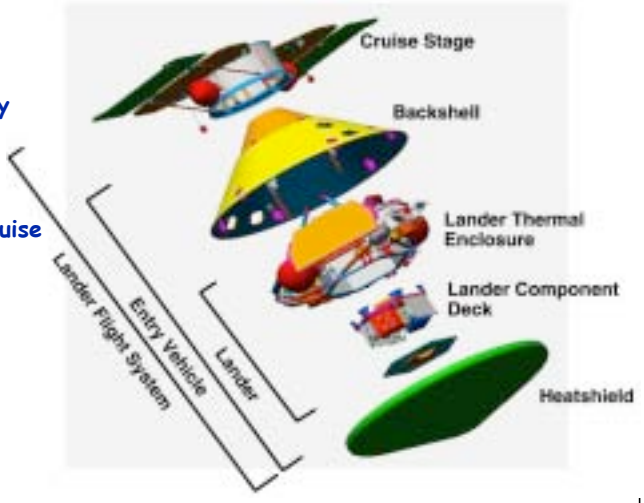
Mars Polar Lander

- Launched
  - 3 Jan 1999
- Mission
  - Land near South Pole
  - Dig for water ice with a robotic arm
- Fate:
  - Arrived 3 Dec 1999
  - No signal received after initial phase of descent
- Cause:
  - Several candidate causes
  - Most likely is premature engine shutdown due to noise on leg sensors



What happened?

- Investigation hampered by lack of data
  - spacecraft not designed to send telemetry during descent
  - This decision severely criticized by review boards
- Possible causes:
  - Lander failed to separate from cruise stage (plausible but unlikely)
  - Landing site too steep (plausible)
  - Heatshield failed (plausible)
  - Loss of control due to dynamic effects (plausible)
  - Loss of control due to center-of-mass shift (plausible)
  - Premature Shutdown of Descent Engines (most likely!)
  - Parachute drapes over lander (plausible)
  - Backshell hits lander (plausible but unlikely)



Premature Shutdown Scenario

- **Cause of error**
  - Magnetic sensor on each leg senses touchdown
  - Legs unfold at 1500m above surface
  - software accepts transient signals on touchdown sensors during unfolding
- **Factors**
  - *System* requirement to ignore the transient signals
  - But the *software* requirements did not describe the effect
  - Engineers present at code inspection didn't understand the effect
  - Not caught in testing because:
  - Unit testing didn't include the transients
  - Sensors improperly wired during integration tests (no touchdown detected!)
- **Result of error**
  - Engines shut down before spacecraft has landed
  - estimated at 40m above surface, travelling at 13 m/s
  - estimated impact velocity 22m/s (spacecraft would not survive this)
  - nominal touchdown velocity 2.4m/s

SYSTEM REQUIREMENTS	FLIGHT SOFTWARE REQUIREMENTS
	3.7.2.2.4.2 Processing
1) The touchdown sensors shall be sampled at 100-Hz rate.	a. The lander flight software shall cyclically check the state of each of the three touchdown sensors (one at 100 Hz during EDL.
The sampling process shall be initiated prior to lander entry to keep processor demand constant.	b. The lander flight software shall be able to cyclically check the touchdown event state with or without touchdown event generation enabled.
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.	c. Upon enabling touchdown event generation, the lan flight software shall attempt to detect failed sens marking the sensor as bad when the sensor indicat "touchdown state" on two consecutive reads.
The test shall consist of two (2) sequential sensor readings showing the expected sensor status.	d. The lander flight software shall generate the landin event based on two consecutive reads indicating touchdown from any one of the "good" touchdown sensors.
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.	

Adapted from the "Report of the Loss of the Mars Polar Lander and Deep Space 2 Missions -- JPL Special Review Board (Casani Report) - March 2000". See http://www.nasa.gov/newsinfo/marsreports.html

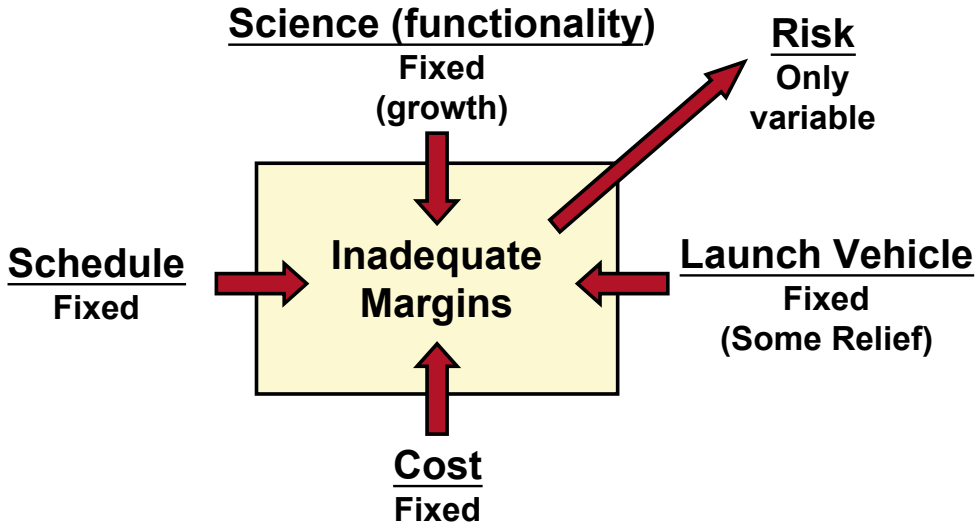
Lessons?

Documentation is no substitute for real communication

Software bugs hide behind other bugs
(full regression testing essential!)

Fixed cost + fixed schedule = increased risk

A Programmatic Failure at JPL



Adapted from MPIAT - Mars Program Independent Assessment Team Summary Report, NASA JPL, March 14, 2000.
See http://www.nasa.gov/newsinfo/marsreports.html

Learning the Right Lessons

"In most of the major accidents of the past 25 years, technical information on how to prevent the accident was known, and often even implemented. But in each case… [this was] negated by organisational or managerial flaws." (Leveson, Safeware)

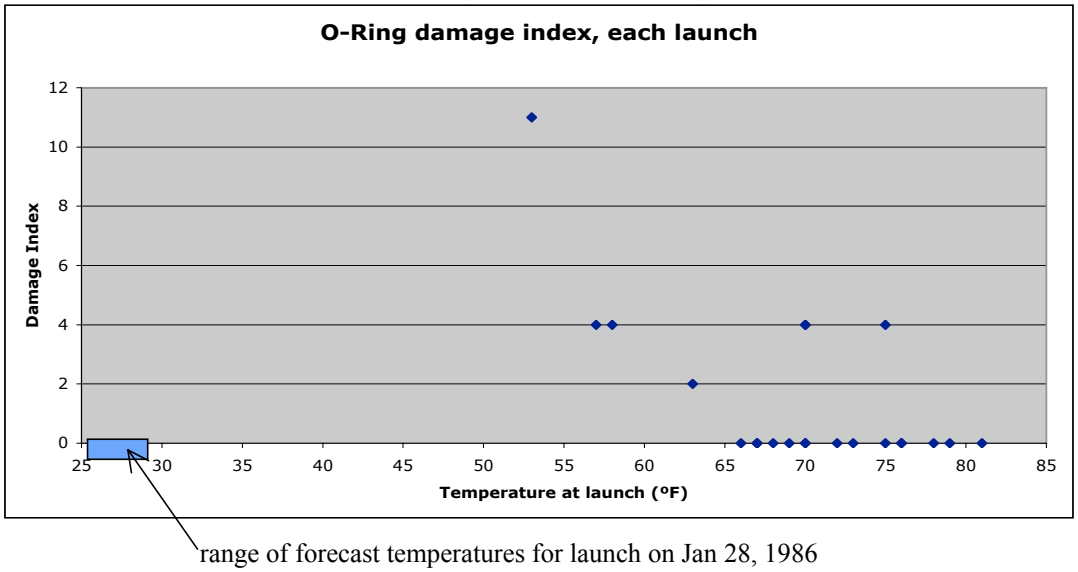
Factor	STS 51L	Ariane 501	Path-finder	MCO	MPL	STS 107
Didn't test to spec		●		●	●	
Insufficient test data	●	●			●	●
Tested "wrong" system		●			●	
No regression test					●	
Lack of integration testing		●		●		
System changed after testing					●	?
Requirement not implemented		?		●	●	
Lack of diagnostic data during operation			●	●	●	●
System deployed before ready	●			?	?	
Didn't use problem reporting system	●		●	●	●	
Didn't track problems properly	●	●	●	●	●	●
Didn't investigate anomalies	●		●	●		●
Poor communication between teams	●	●	●	●	●	●
Insufficient staffing	●			●	●	
Failure to adjust budget and schedule	●			●	●	
Inexperienced managers	?			●	●	
Commercial pressures took priority	●	●		●	●	●
'Redundant' design not really redundant	●	●				
Lack of expertise at inspections		●		●	●	
Different team maintains software				●	●	
Reused code w/o checking assumptions		●				

Why are warning signs ignored?

- Equal voice
  - E.g. Independence of Risk Assessment processes:
    - Managerial Independence; Financial Independence; Technical Independence
- Understanding & using the available controls
  - A manager can control four variables:
    - Resources (can get more dollars, facilities, personnel)
    - Time (can increase schedule, delay milestones, etc.)
    - Product (can reduce functionality - e.g. scrub requirements)
    - Risk (can decide which risks are acceptable)
  - None should be assumed to be fixed
- Proper measurement on both sides of the argument
  - E.g. don't just quantify the cost/schedule side
- Clear communication of technical factors
  - Engineering staff routinely fail to get the message across to management

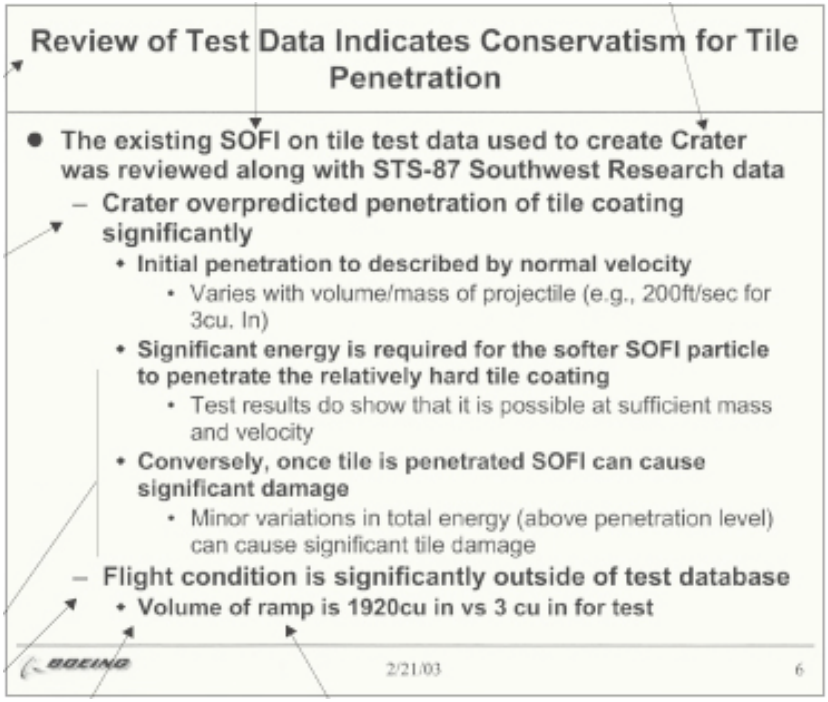
Importance of Communication

- The graph that was never drawn…
  - For the Challenger launch decision, this data was available
  - But was never collected and presented this way



Adapted from Tufte, "Visual Explanations"

The lessons of history…



Adapted from Tufte, "The Cognitive Style of Powerpoint"

Summary

→ **Where to improve systems engineering:**

- Examine and improve the organization's safety culture
- Measure risk properly
- Better conflict resolution processes
- Communicate the risks clearly
- Structure the organization so as to avoid masking problems
- Use design processes to eliminate hazards

→ **Software introduces extra risk:**

- It defeats redundancy designs
- It introduces many more subsystem interactions
- It continues to evolve even after deployment
- It's much harder to understand than other subsystems
- We don't yet have a mature science of software behaviour

Resource List (part 1)

- **Software Risk**
  - Endres, A., Rombach, D. "A Handbook of Software and Systems Engineering: Empirical Observations, Laws, and Theories" Addison Wesley, 2003.
  - MacKenzie, Donald. "Mechanizing Proof: Computing, Risk, and Trust". MIT Press, 2001.
- **Communication of Risk**
  - Tufte, Edward. "Visual Explanations, Images and Quantities, Evidence and Narrative". Cheshire, CT: Graphics Press, 1997
  - Feynman, Richard. "What do you care what other people think?". W.W. Norton & Company, New York, 1988.

Resource List (part 2)

- **General books on Safety & System failure**
  - Leveson, Nancy. "Safeware: System Safety and Computers". Addison Wesley, Reading, MA, 1995.
  - Perrow, Charles. "Normal Accidents: Living with High-Risk Technology". Basic Books, New York, 1984.
  - Petroski, Henry. "To Engineer is Human: The Role of Failure in Successful Design". St Martin's Press, New York, 1985
- **Space Shuttle**
  - Current info about the shuttle:
    - http://spaceflight.nasa.gov/shuttle/
  - Info about Challenger:
    - http://www-pao.ksc.nasa.gov/kscpao/shuttle/missions/51-l/mission-51-l.html
  - Rogers Commission Report (see especially appendix F, by Richard Feynman)
    - http://science.ksc.nasa.gov/shuttle/missions/51-l/docs/rogers-commission/table-of-contents.html
  - A Succinct summary of the key factors and issues:
    - http://ethics.tamu.edu/ethics/ethics/shuttle/shuttle1.htm
- **Ariane-5**
  - Info about ESA's launchers:
    - http://www.esa.int/export/esaLA/launchers.html
  - Inquiry report & Press release:
    - http://www.esrin.esa.it/htdocs/tidc/Press/Press96/press33.html
- **Mars Observer**
  - Project summary
    - http://www.msss.com/mars/observer/project/mo_loss/moloss.html
  - Brief summary of possible causes
    - http://catless.ncl.ac.uk/Risks/14.89.html#subj1

Resource List (part 3)

- Mars Pathfinder
  - Project info:
    - http://mars.jpl.nasa.gov/MPF/index1.html
  - Report on the priority inversion problem:
    - http://catless.ncl.ac.uk/Risks/19.49.html#subj1
- Mars Climate Orbiter
  - Project Info:
    - http://mars.jpl.nasa.gov/msp98/orbiter/
  - Investigation Report:
    - ftp://ftp.hq.nasa.gov/pub/pao/reports/2000/MCO_MIB_Report.pdf
- Mars Polar Lander & Deep Space 2
  - Project info:
    - http://mars.jpl.nasa.gov/msp98/lander/
    - http://mars.jpl.nasa.gov/msp98/ds2/
  - Investigation Reports:
    - http://www.nasa.gov/newsinfo/marsreports.html
- General Background
  - RISKS forum archive:
    - http://catless.ncl.ac.uk/Risks/
  - JPL's list of missions (past, present and future)
    - http://www.jpl.nasa.gov/missions/missions_index.html
  - Basics of Space Flight:
    - http://www.jpl.nasa.gov/basics/