*Managing Software Development – The Hidden Risk **

Dr. Steve Jolly
Sensing & Exploration Systems
Lockheed Martin Space Systems Company

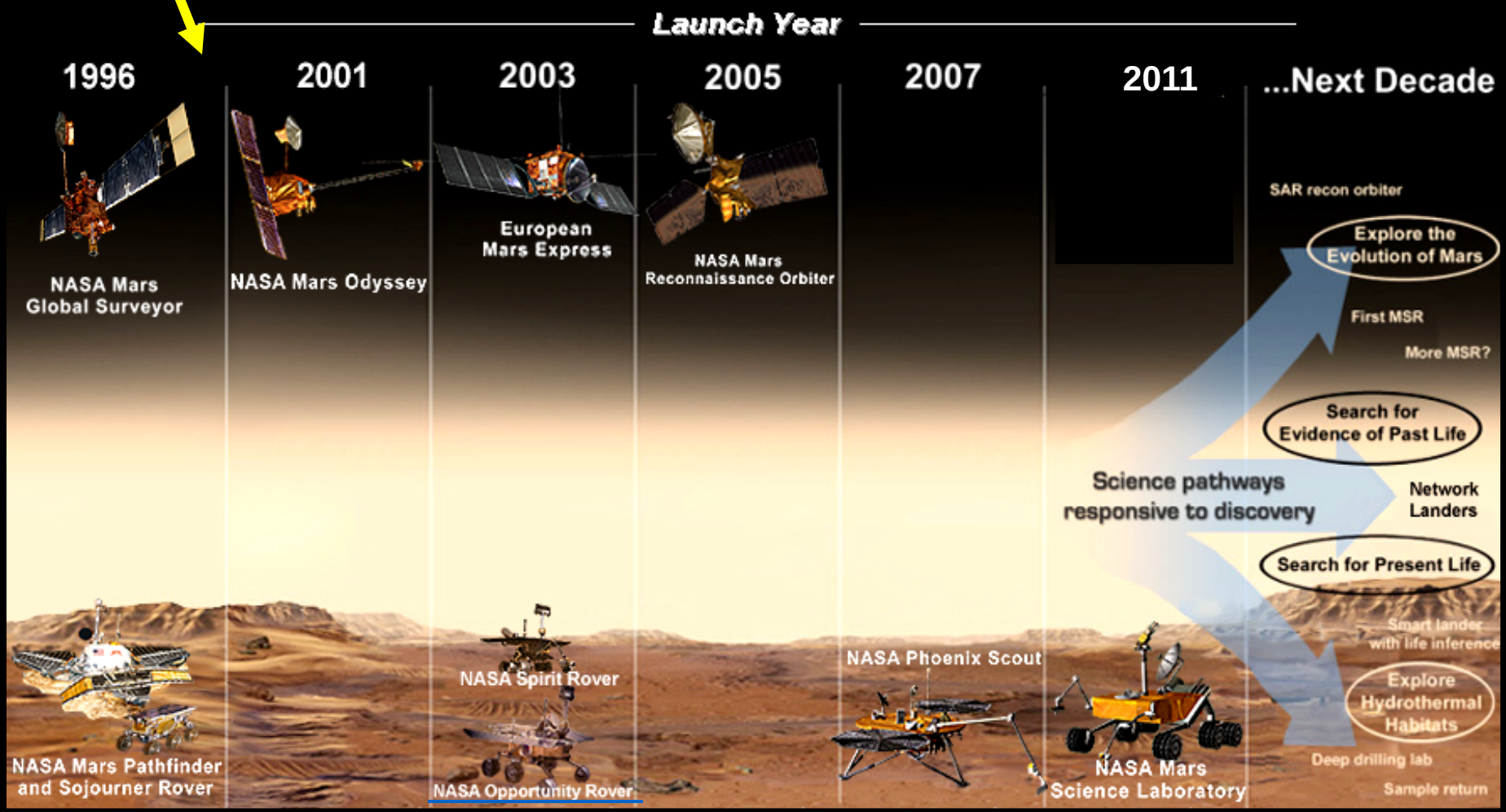
*Based largely on the work: “Is Software Broken?” by Steve Jolly, NASA ASK Magazine, Spring 2009; and the IEEE Fourth International Conference of System of Systems Engineering as “System of Systems in Space Exploration: Is Software Broken?”, Steve Jolly, Albuquerque, New Mexico June 1, 2009

Brief history of spacecraft development

- Example of the Mars Exploration Program
  - Danger
  - Real-time embedded systems challenge
  - Fault protection

Mars '98

Robotic Mars Exploration



Search: Aqueous Minerals Found

Search: Subsurface Ice Found

Search: Evidence for water Found

Determine: Global Extent of Habitable Environments In Work

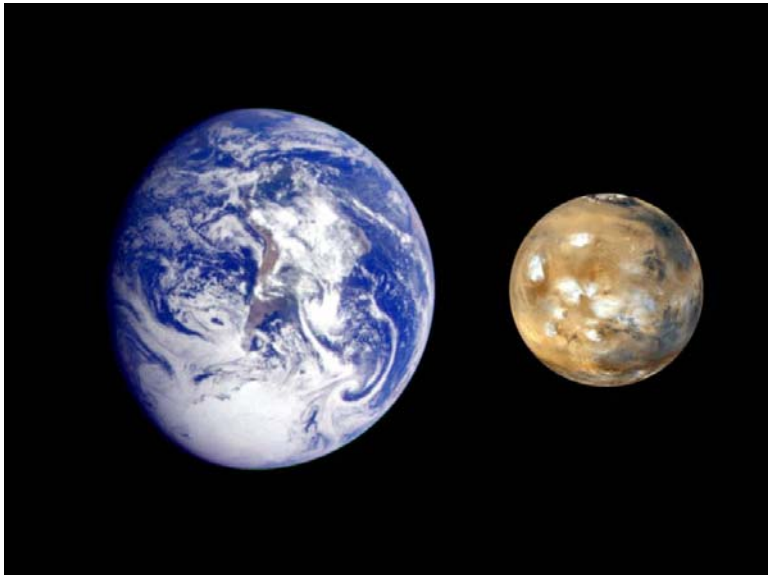
Characterize: Subsurface Ice Found

Determine: Bio Potential of Site

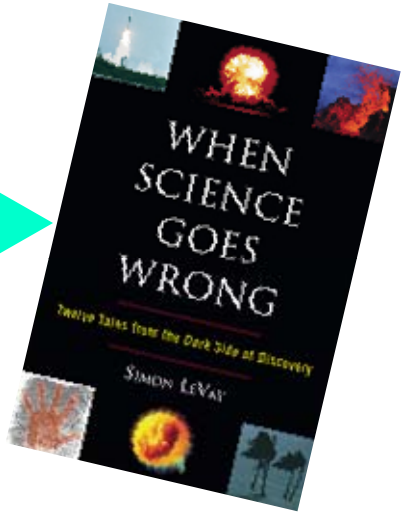
Image Credits: NASA/JPL

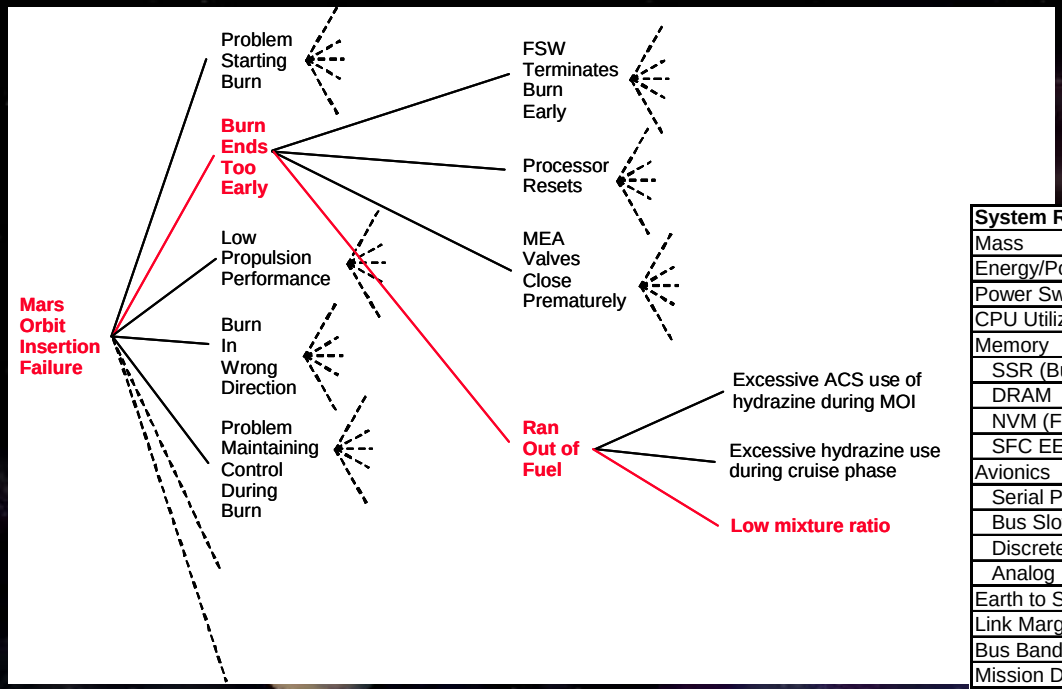
Mars: Easy to Become Infamous …

1. [Unnamed], USSR, 10/10/60, Mars flyby, did not reach Earth orbit
2. [Unnamed], USSR, 10/14/60, Mars flyby, did not reach Earth orbit
3. [Unnamed], USSR, 10/24/62, Mars flyby, achieved Earth orbit only
4. Mars 1, USSR, 11/1/62, Mars flyby, radio failed
5. [Unnamed], USSR, 11/4/62, Mars flyby, achieved Earth orbit only
6. Mariner 3, U.S., 11/5/64, Mars flyby, shroud failed to jettison
7. Mariner 4, U.S. 11/28/64, first successful Mars flyby 7/14/65
8. Zond 2, USSR, 11/30/64, Mars flyby, passed Mars radio failed, no data
9. Mariner 6, U.S., 2/24/69, Mars flyby 7/31/69, returned 75 photos
10. Mariner 7, U.S., 3/27/69, Mars flyby 8/5/69, returned 126 photos
11. Mariner 8, U.S., 5/8/71, Mars orbiter, failed during launch
12. Kosmos 419, USSR, 5/10/71, Mars lander, achieved Earth orbit only
13. Mars 2, USSR, 5/19/71, Mars orbiter/lander arrived 11/27/71, no useful data
14. Mars 3, USSR, 5/28/71, Mars orbiter/lander, arrived 12/3/71
15. Mariner 9, U.S., 5/30/71, Mars orbiter, in orbit 11/13/71 to 10/27/72
16. Mars 4, USSR, 7/21/73, failed Mars orbiter, flew past Mars 2/10/74
17. Mars 5, USSR, 7/25/73, Mars orbiter, arrived 2/12/74, lasted a few days
18. Mars 6, USSR, 8/5/73, Mars orbiter/lander, arrived 3/12/74, little data
19. Mars 7, USSR, 8/9/73, Mars orbiter/lander, arrived 3/9/74, little data
20. Viking 1, U.S., 8/20/75, orbiter/lander, orbit 6/19/76-1980, lander 7/20/76-1982
21. Viking 2, U.S., 9/9/75, orbiter/lander, orbit 8/7/76-1987, lander 9/3/76-1980
22. Phobos 1, USSR, 7/7/88, Mars/Phobos orbiter/lander, lost 8/89 en route
23. Phobos 2, USSR, 7/12/88, Mars/Phobos orbiter/lander, lost 3/89 near Phobos
24. Mars Observer, U.S., 9/25/92, lost just before Mars arrival 8/21/93
25. Mars Global Surveyor, U.S., 11/7/96, Mars orbiter, arrived 9/12/97
26. Mars 96, Russia, 11/16/96, orbiter and landers, launch vehicle failed
27. Mars Pathfinder, U.S., 12/4/96
28. Nozomi (Planet-B), Japan, 7/4/98, Mars orbiter, failed to capture
29. Mars Climate Orbiter, U.S., 12/11/98, lost upon arrival 9/23/99
30. Mars Polar Lander, U.S., 1/3/99
31. Deep Space 2, Probes, U.S., 1/3/99
32. 2001 Odyssey, U.S., Mars Orbiter, launched 4/7/01
33. Mars Express, ESA, Mars Orbiter, launched 6/03
34. Beagle 2, ESA, Mars Lander, launched 6/03, no contact since EDL
35. Spirit, U.S., Mars Rover, launched 6/10/03
36. Opportunity, U.S., Mars Rover, launched 7/7/03
37. Mars Reconnaissance Orbiter, U.S., Mars Orbiter, Launched 8/12
38. Phoenix, U.S., Mars Lander, Launched 2007



37% success rate …

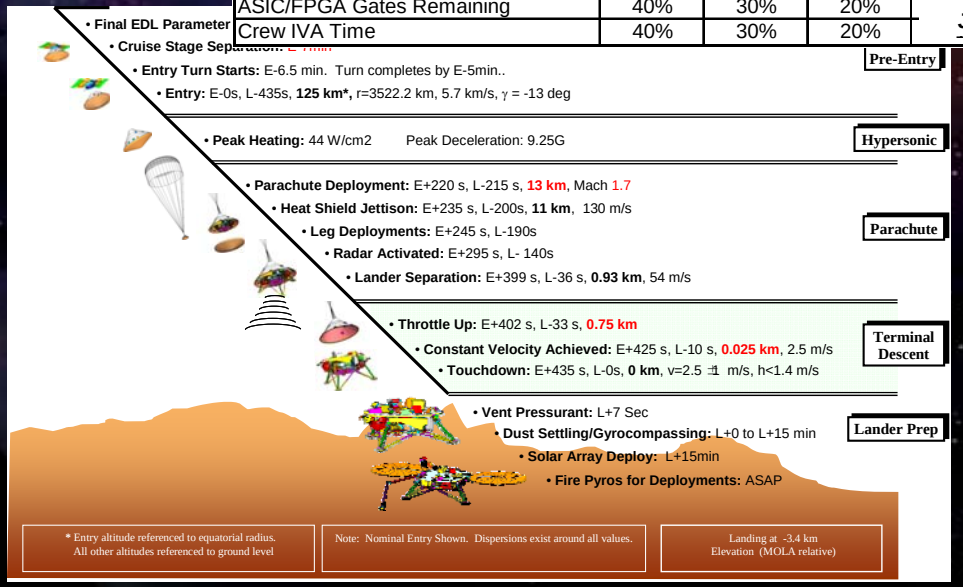




System Resource/Mission Phase	SDR	PDR	CDR	ATLO start	Launch
Mass	25%	20%	15%	10%	3%
Energy/Power	30%	20%	15%	10%	10%
Power Switches	35%	30%	20%	15%	10%
CPU Utilization	75%	60%	50%	30%	20%
Memory					
SSR (Bulk storage)	30%	20%	20%	15%	10%
DRAM	75%	60%	50%	30%	20%
NVM (Flash)	75%	60%	50%	40%	30%
SFC EEPROM	75%	60%	50%	40%	30%
Avionics					
Serial Port Assignments	3	3	2	2	2
Bus Slot Assignments	3	2	2	1	1
Discrete I/O	30%	20%	15%	12.50%	10%
Analog I/O	30%	20%	15%	12.50%	10%
Earth to S/C Link(C)	3 db	3 db	3 db	3 db	3 db
Link Margin Bit Error Rate (3 sigma)	1.00E-06	1.00E-05	1.00E-05	1.00E-05	1.00E-05
Bus Bandwidth	60%	60%	55%	55%	50%
Mission Data Volume	20%	20%	15%	10%	10%
ASIC/FPGA Gates Remaining	40%	30%	20%		
Crew IVA Time	40%	30%	20%		

Jamshidi 2009

Designing for Failure Tolerance is Increasingly More Difficult

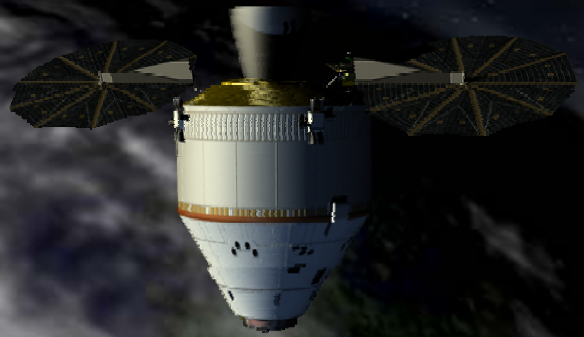


Modern Spacecraft Development Drama

- Many spacecraft development programs across the industry are experiencing various levels of software schedule difficulties and overruns associated with late and/or defective software
  - Can lead to major cost overrun due to “marching army effect”
  - Can lead to major delay in delivery of assets to orbit
- Three years ago Lockheed Martin Systems Engineering and Software experts held a Kaizen™ Event to discover root cause
  - Had data from many programs
  - Expected to drill down to a handful of root causes, probably process related
- Results were unexpected – 130 root causes, how can this be? They can’t be independent …
  - I came away from the event concerned and confused ….
- Team reconvened to find the underlying causes, but didn’t materially improve the list or find the “real” root causes
  - But several of us began to note a pattern emerging!
- Even though we couldn’t definitively link the large majority of causes, we found that problems in requirements issues, development, testing, and validation and verification of the actual code – <u>all revolved around interfaces!</u>

In stark contrast to Apollo, Orion's FSW has more interfaces than any other subsystem save Structure!

Orion - Crew Exploration Vehicle



Orion Crew Module Physical and Functional Interfaces	Structures	Thermal Protection System	Passive Thermal Control	Mechanisms	Propulsion	Power	Command & Data Handling	Communication & Telemetry	Crew Interface	**Software**	Guidance, Nav. and Control	ECLSS	Crew Systems	Recovery & Landing System	Active Thermal Control
Structures															
Thermal Protection System															
Passive Thermal Control															
Mechanisms										1					
Propulsion										2					
Power										3					
Command & Data Handling										4					
Communication & Telemetry										5					
Crew Interface										6					
Software											7	8	9	10	11
Guidnace, Nav. and Control															
ECLSS															
Crew Systems															
Recovery & Landing System															
Active Thermal Control															

Jamshidi 2009

Software revolution

- Gemini flight computer and the later Apollo Guidance Computer (AGC) were limited to 13,000–36,000 words of storage lines (Tomayko 1988)
  - AGC's interaction with other subsystems was limited to those necessary to carry out its guidance function.
  - Astronauts provided input to the AGC via a keypad interface; other subsystems onboard were controlled manually, by ground command, or both combined with analog electrical devices
- Apollo's original 36,000 words of assembly language have grown to one million lines of high-level code on Orion
- Any resemblance of a modern spacecraft to one forty years ago is merely physical
  - advent of object-oriented code
  - the growth in parameterization
  - the absolute explosion of the use of firmware in ever-more sophisticated devices like FPGAs (now reprogrammable)
- Has amplified flight software to the forefront of development issues.; underneath lurks a different animal, and the development challenges have changed. But what about ten years ago?

Recent Evidence

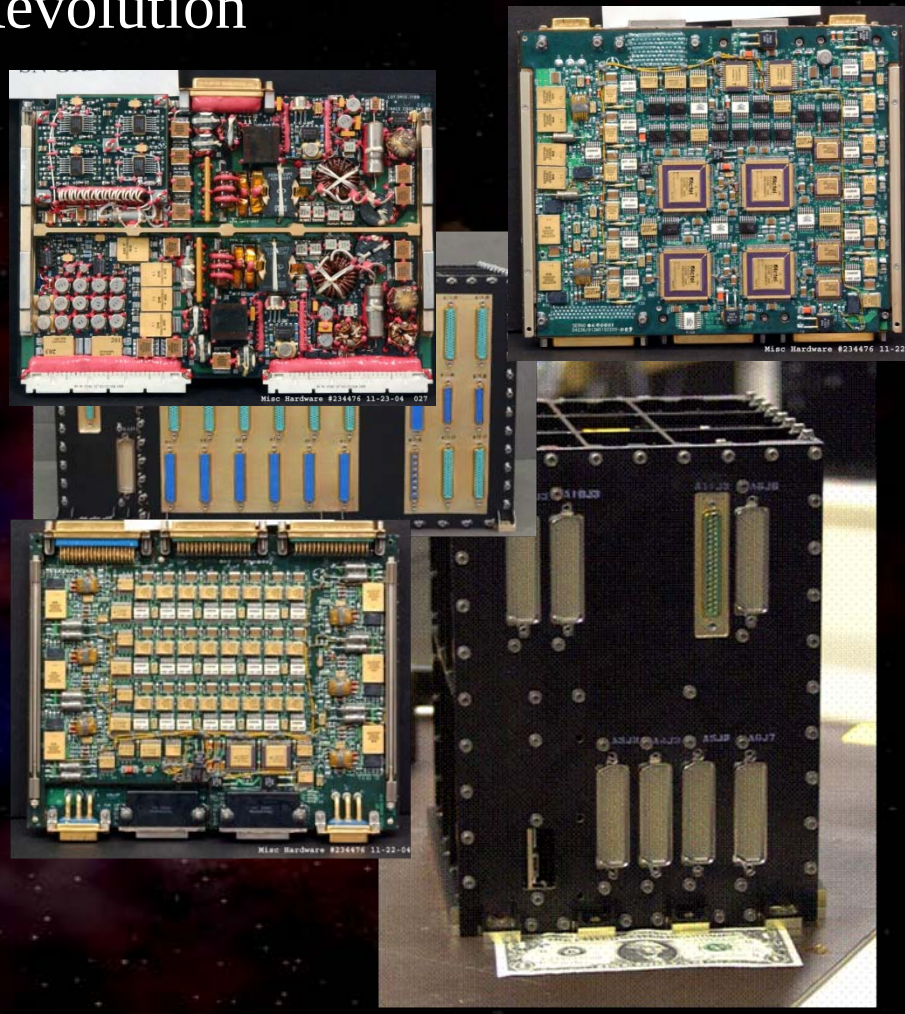
- Compare the 1996 Mars Global Surveyor (MGS) era of the mid-nineties to the 2005 Mars Reconnaissance Orbiter (MRO) spacecraft
  - Code growth in logical source lines of code (SLOC) more than doubled from 113,000 logical SLOC to 250,000 logical SLOC
  - This comparison does not include the firmware growth from MGS to MRO, which is likely to be an order of magnitude greater
- From Stardust and Mars Odyssey (late 1990s and 2001) to MRO
  - Parameter databases necessary to make the code fly these missions grew from about 25,000 for Stardust to more than 125,000 for MRO
  - Mars Odyssey had a few thousand parameters that could be classified as mission critical (that is, if they were wrong the mission was lost)
  - MRO had more than 20,000!!
- We now have the advantage of being able to reuse a lot of code design for radically different missions by simply adjusting parameters
  - we also have the disadvantage of tracking and certifying thousands upon thousands of parameters, and millions of combinations
- This is not confined to NASA; it is true throughout other industries

Other Industries (Dvorak 2009)

- Software functionality in military aircraft has grown from 8% to 80% from 1960 to 2000
  - The F-22A (Raptor) has a reported 2.5 million lines of code
- The average GM car in 1970 had 100 thousand lines of code, projected to be 100 million lines of code in 2010
  - On 70 to 100 microprocessor-based electronic control units

But this isn't all about Software: It is Strongly Coupled to the Electronics Revolution

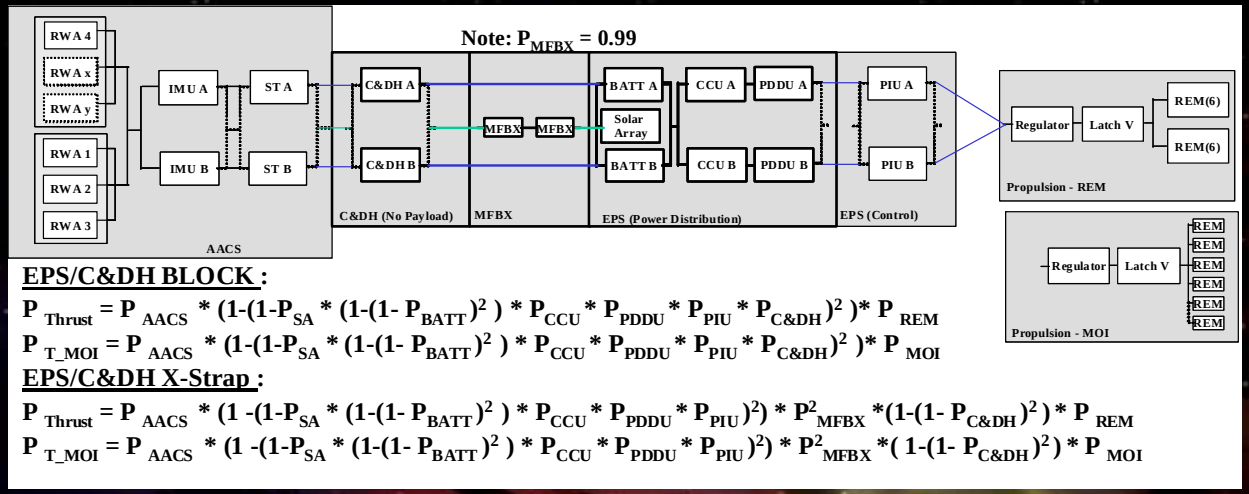
- 1990s spacecraft typically had many black boxes in the Command and Data Handling (C&DH) and power subsystems.
- New electronics and new packaging techniques have increased the physical and functional density of the circuit card assembly
- Several boxes became several cards in one box
  - 22 Boxes of Mars Global Surveyor (MGS) was collapsed into one box on Stardust and Mars Odyssey
- The ever-increasing capability of FPGAs and ASICs and simultaneous decrease in power consumption and size, several cards became FPGAs on a single card!
- When you hold a card from a modern C&DH or power subsystem, you are holding many black boxes of the past
- The system is now on a chip!!!
- And yet we still have 20 boxes???



Analysis: Software went from embedded, to subsystem, to system

Together with Avionics, Software has become THE SYSTEM!

Implications on Space Mission Development



EPS/C&DH BLOCK :

$P_{Thrust} = P_{AACS} * (1-(1-P_{SA} * (1-(1-P_{BATT})^2) * P_{CCU} * P_{PDDU} * P_{PIU} * P_{C\&DH})^2) * P_{REM}$

$P_{T\_MOI} = P_{AACS} * (1-(1-P_{SA} * (1-(1-P_{BATT})^2) * P_{CCU} * P_{PDDU} * P_{PIU} * P_{C\&DH})^2) * P_{MOI}$

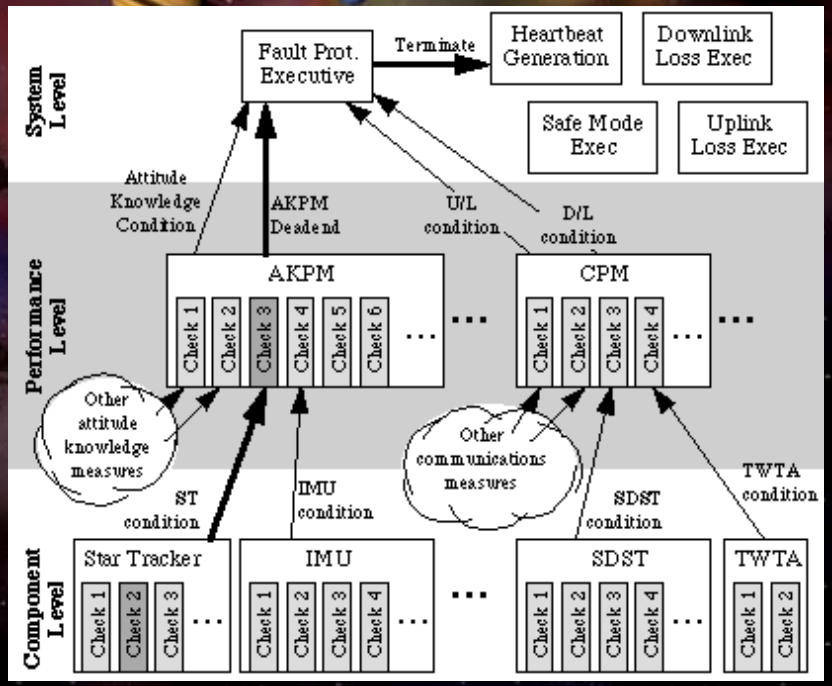
EPS/C&DH X-Strap :

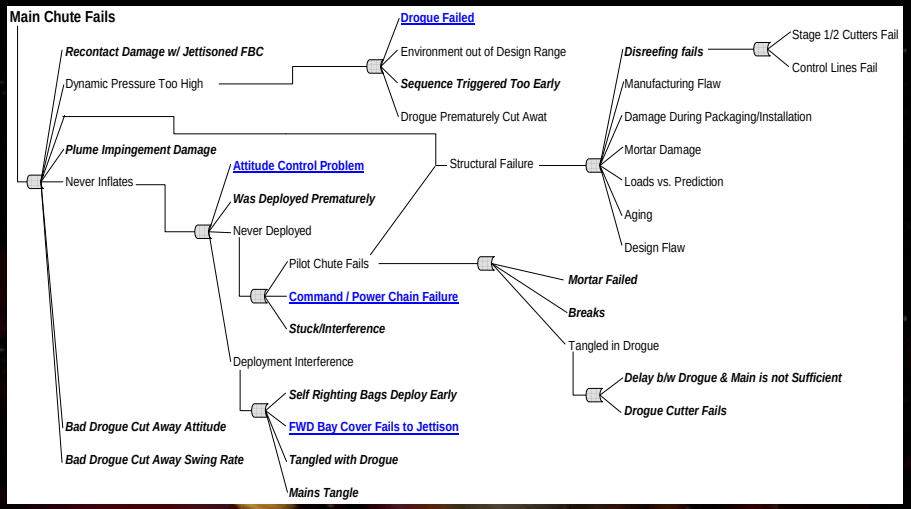
$P_{Thrust} = P_{AACS} * (1-(1-P_{SA} * (1-(1-P_{BATT})^2) * P_{CCU} * P_{PDDU} * P_{PIU})^2) * P^2_{MFBX} * (1-(1-P_{C\&DH})^2) * P_{REM}$

$P_{T\_MOI} = P_{AACS} * (1-(1-P_{SA} * (1-(1-P_{BATT})^2) * P_{CCU} * P_{PDDU} * P_{PIU})^2) * P^2_{MFBX} * (1-(1-P_{C\&DH})^2) * P_{MOI}$

Common mode failure …

The simultaneous loss of 4 of 6 Russian module computers on ISS, in 2007 due to water condensation in a zero-g environment





There are thousands of ways to fail … most have not been explored

"… the use of modern electronics and software has actually increased the failure modes and effects that we must deal with in modern space system design" Jolly 2009

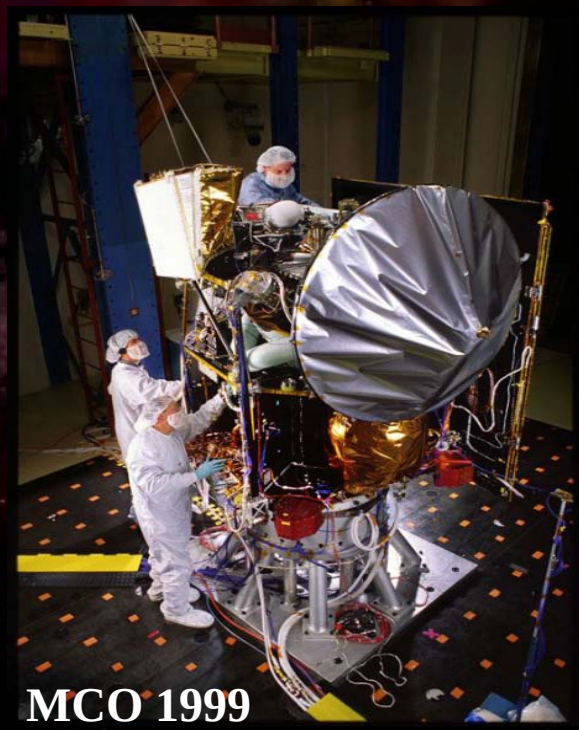
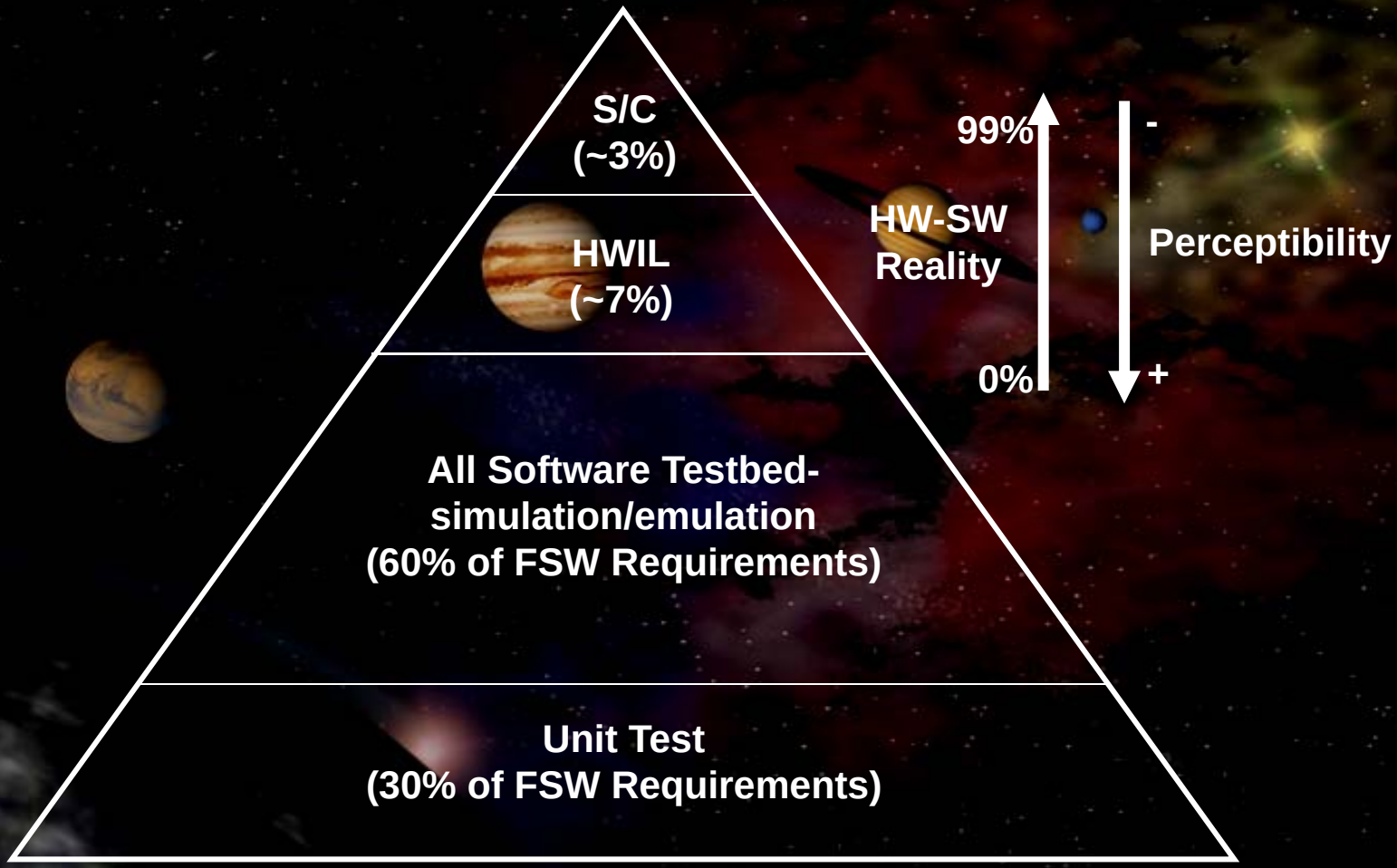


Image Credits: Pat Corkery, Lockheed Martin

Reality of Modern FSW Verification



A Call to Transform Systems Engineering

- Software/firmware can no longer be treated as a subsystem
  - Systems Engineering Organizations Need to Engineer the Software/Avionics System (change in leadership technical background)
- We need agile yet thorough systems engineering techniques and tools to define and manage these numerous interfaces
  - Cannot be handled by the system specification /software specification
  - This includes parameter assurance and management
- Using traditional interface control document techniques to accomplish this will likely bring programs to their knees
  - Due to the sheer overhead of such techniques (e.g., a 200-page formally adjudicated and signed-off interface control document)
- Employing early interface validation is an absolute must
  - via exchanged simulators, emulators, breadboards, and engineering development units with the subsystems and payloads
- If ignored, interface incompatibility will ultimately manifest itself during assembly, test, and launch operations
  - Flight software changes will be the only to make the launch
  - Creating an inevitable marching-army effect and huge cost overruns.

Conclusion and the future

- Bottom line: the game has changed in developing space systems.
  - Software and avionics have become the system
  - Structures, mechanisms, propulsion, etc., are all supporting this new system (apologies to all you mechanical types out there).

- Today's avionics components that make up the C&DH and power functions are systems on a chip (many boxes of the past on a chip)
  - Together with the software and firmware, constitute myriad interfaces to everything on the spacecraft.
- To be a successful system integrator we must engineer and understand the details of these hardware–software interfaces, down to the circuit level or deeper
  - Refernce to the core avionics that constitute the system, those that handle input-output, command and control, power distribution, and fault protection, not avionics components that attach to the system with few interfaces (like a star camera)
- Merely procuring the C&DH and power components as black boxes will result in overruns and schedule delays
  - Results in not understand their design, their failure modes, their interaction with the physical spacecraft and its environment, and how software knits the whole story together

References

1. Jolly S., “Is Software Broken?”, NASA ASK Magazine, Spring 2009
2. Jamshidi M., Editor, System of Systems Engineering: Innovations for the 21st Century, Chapter 14 by Jolly and Muirhead, pp. xxxx, Wiley, 2009
3. Tomayko J., Computers in Spaceflight: The NASA Experience, NASA Contractor Report 182505, Washington, DC: NASA History Office, 1988
4. Dvorak D., “NASA Study on Flight Software Complexity”, AIAA 2009-1882, AIAA Infotech@Aerospace Conference, 6 - 9 April 2009, Seattle, Washington


Q&A