Large Scale Aerospace Software Development

Heidi Perry
Draper Laboratory
Unified Lecture 23
May 12, 2004

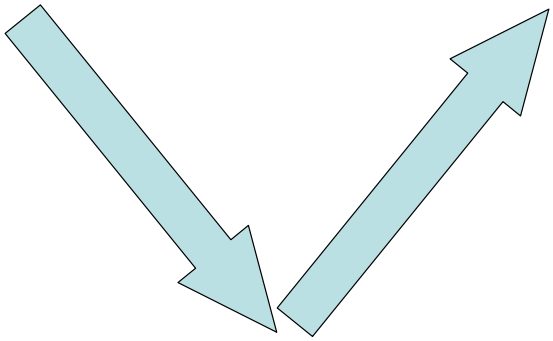
What Software Do You Need to Get To Mars?

Today's aerospace applications are extremely software intensive. The designs of all modern flight platforms, e.g. aircraft, rotorcraft, submersibles, missiles and spacecraft, have been revolutionized because of the integration of advanced microelectronics and complex sensor systems. All of these systems extensively use information generated by a myriad of sensor subsystems that are highly integrated into a real-time embedded software architecture. This lecture will explore what it takes to develop large scale software for a number of examples of highly reliable, flight critical software applications.

Software for planetary science

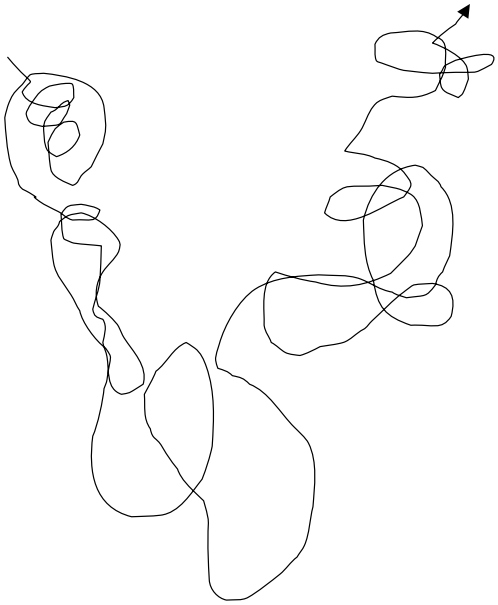
Software in an Ideal Solar System

- Software life cycle
  - Requirements
  - Analysis
  - Design
  - Unit Code
  - System Build
  - Test
  - Deliver / Integrate
  - Fly / Run



Software in the Real Solar System

- Requirements
- Analyze
- Iterate Requirements
- Analyze
- Design
- Iterate Requirements
- Analyze
- Design
- Code
- Requirements Creep
- Analyze Design Code Test Fail
- Analyze Design Code Test Build
- Test Code Build Test
- Deliver test code test launch test code test test test test

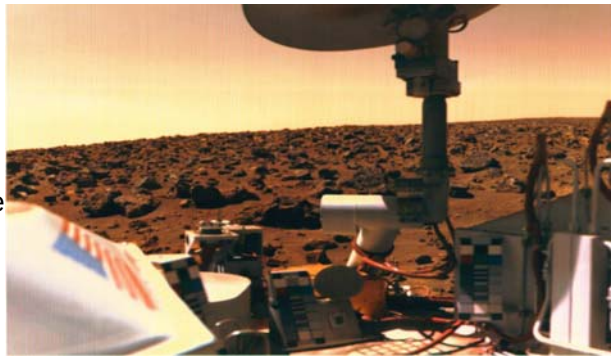


Example: Mars Rovers (Pathfinder, MER)

- Requirements:
  - Land Safely
  - Take a picture
  - Deploy the Rover
  - Analyze dirt and rocks
  - Send more pictures and analysis data
- Analysis
  - "Land Safely" means that airbags can't pop, so we have to limit mass, so we can only have one processor and one memory board, so… add requirement on limiting all FSW to space available

Example (continued)

- Requirements:
  - Land Safely using only one processor
  - Take a picture
  - Deploy the Rover
  - Analyze dirt and rocks
  - Send more pictures and analysis
- Analysis
  - With mass limiting memory available, can only take so many pictures before we have to stop and downlink data to Earth
- Design
  - This part of memory for scratch
  - This part for science data
  - Take color picture and process data for 2 seconds
  - Downlink to Earth for 1 hour



Our model:
Viking 1 1976

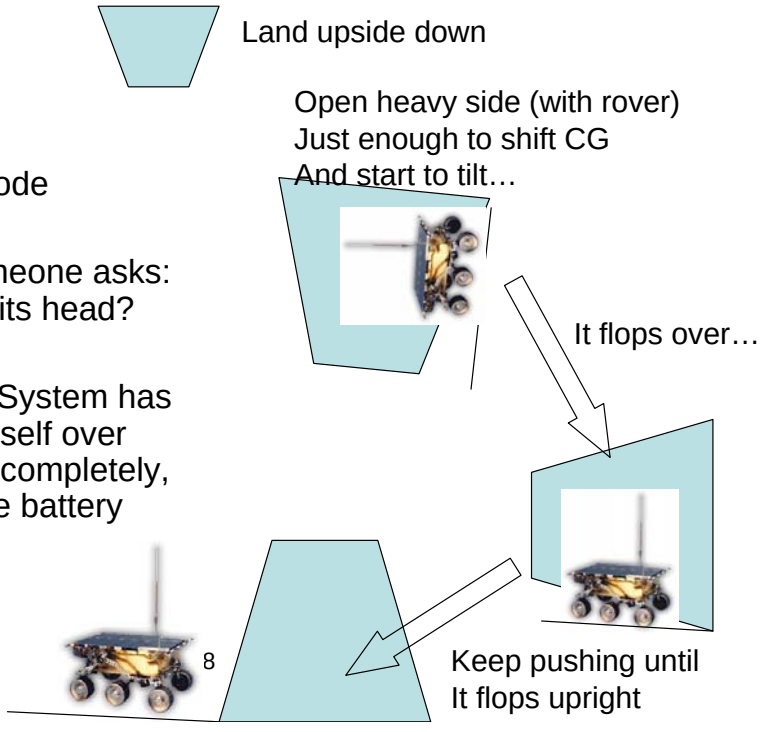
BUT we only get 10 minutes of downlink time that first day, so we can only send back a B&W photo…

Example (cont)

- Requirements:
- Analysis
- Design
- Code code code code

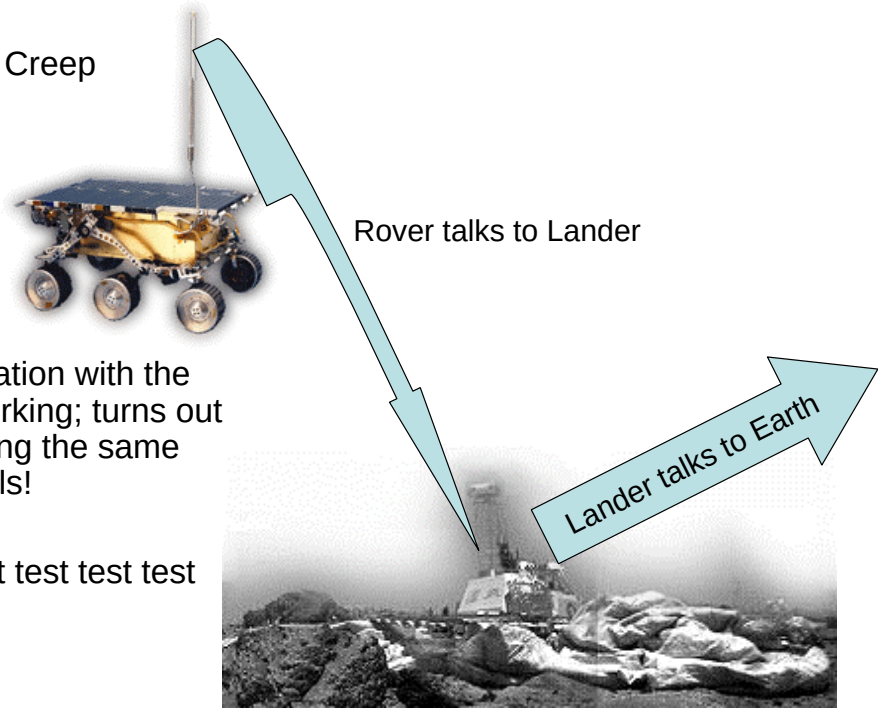
In system review, someone asks:
What if it lands on its head?

- New requirement: System has to be able to turn itself over before it opens up completely, and do it before the battery power runs out



Example (cont)

- Requirements Creep
- Analyze
- Design
- Code
- Test
- Fail



Rover communication with the lander isn't working; turns out they aren't using the same comm protocols!

Analyze code test test test test

Example (cont)

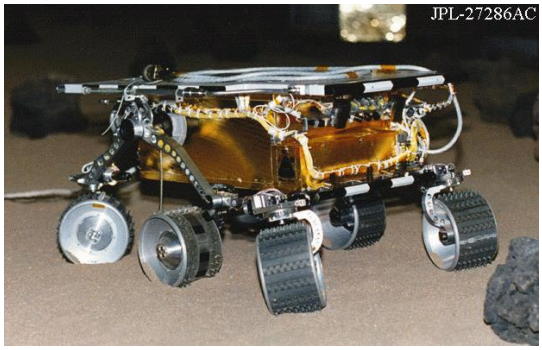
- Test test code test
- Deliver

ROVER still has never stood up and crawled off the Lander on its own!

- Test test test test
- Rebuild hardware test test test test…

Example (cont)

- OUT OF TIME
  - Planets only line up briefly every 2 years, so we have to launch….
- LAUNCH
- Test test test test
- Uplink
- Test test test test



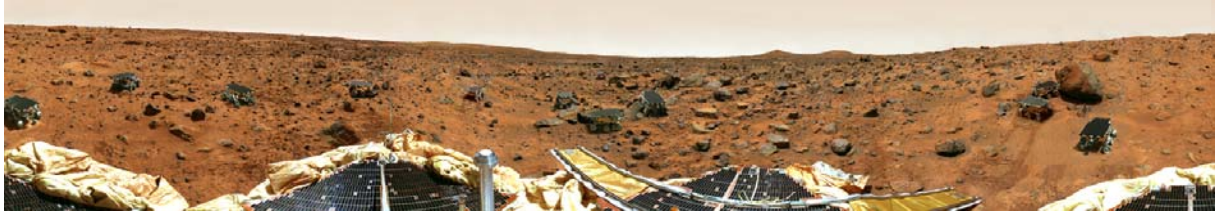
Non-flight rover was used to test out new software in a Mars simulator dubbed the "sandbox"

SUCCESS!

So many rocks, so little time….

Science team comes up with an idea to test soil adhesion by locking 4 wheels and spinning one so we're back to…

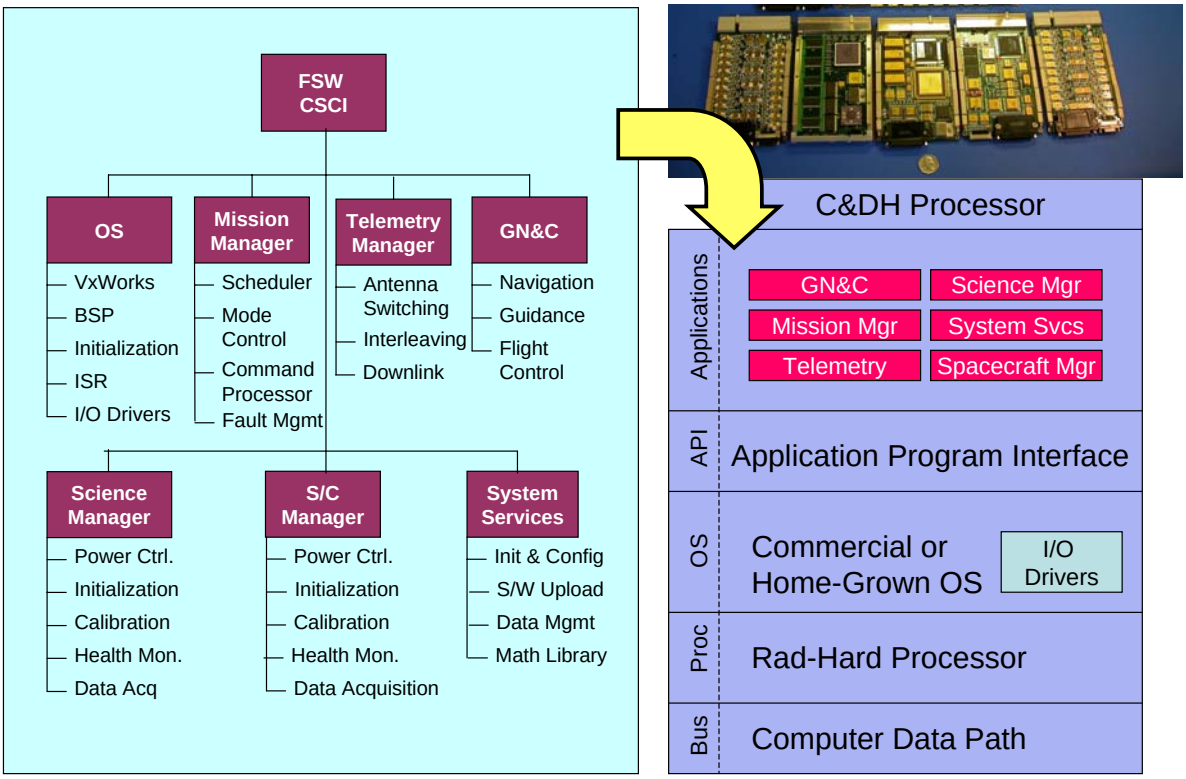
Requirements
Design
Analysis
Code
Test



A look at software complexity

- What software does it take to implement a mission to Mars?
  - Analysis SW
  - Simulation SW
  - Spacecraft SW
  - Ground SW

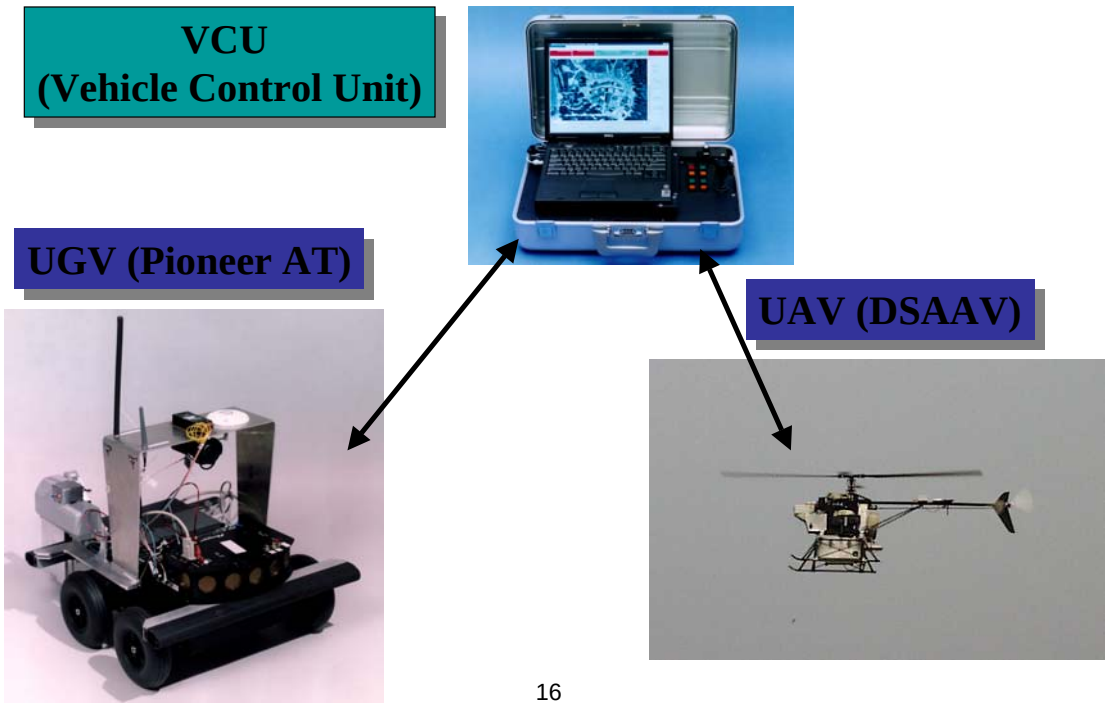
Sample Spacecraft Flight Software Components



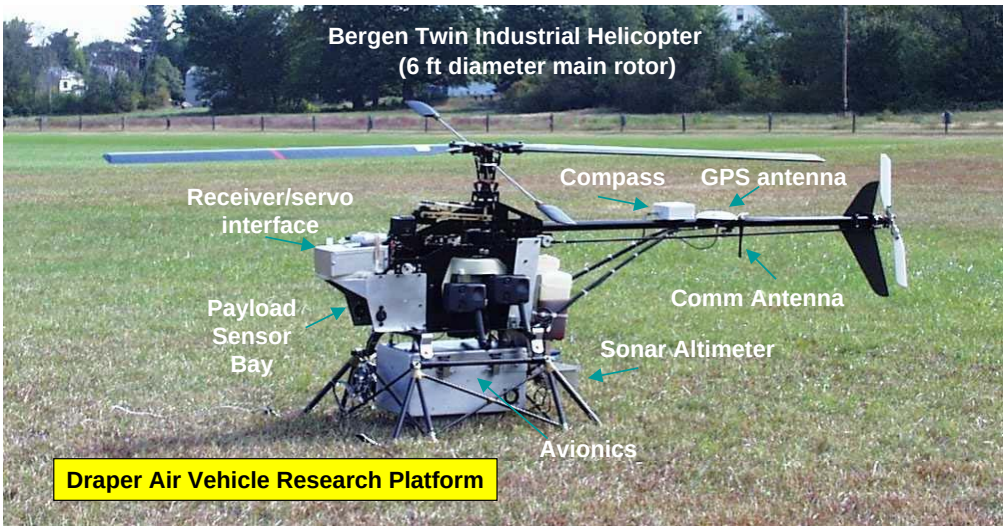
Source Lines of Code

- Small Scale: < 15K SLOC
  - Autonomous Helicopter Software
- Medium Scale: 15K-100KSLOC
  - Satellite Mission Software
  - Submarine GN&C Software
  - Spacecraft Flight Software
- Large Scale: >100 KSLOC
  - Collaborative Unmanned Aircraft Systems (J-UCAS)
  - Complex Avionics Systems (F-22)

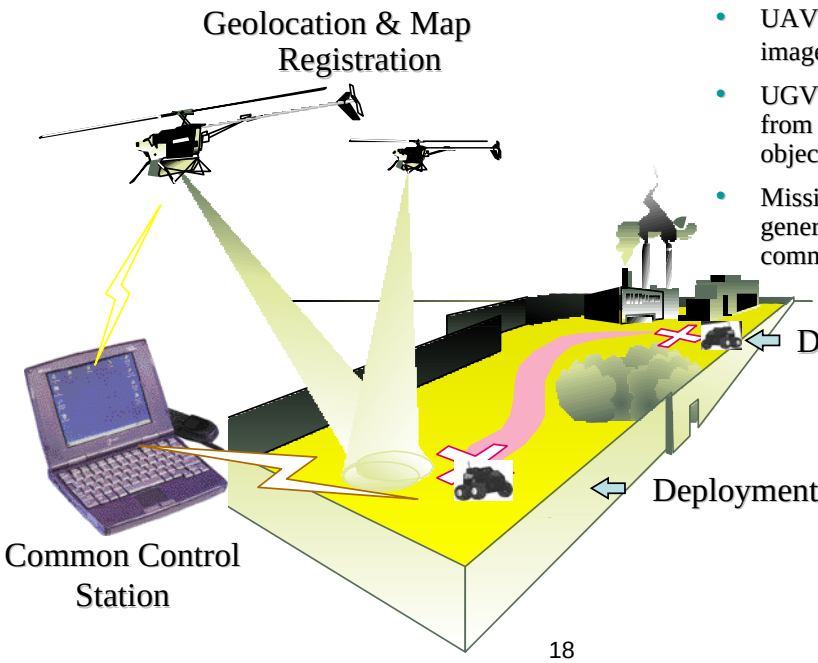
Small Scale: Autonomous Helicopter



Unmanned Air Vehicle [UAV]

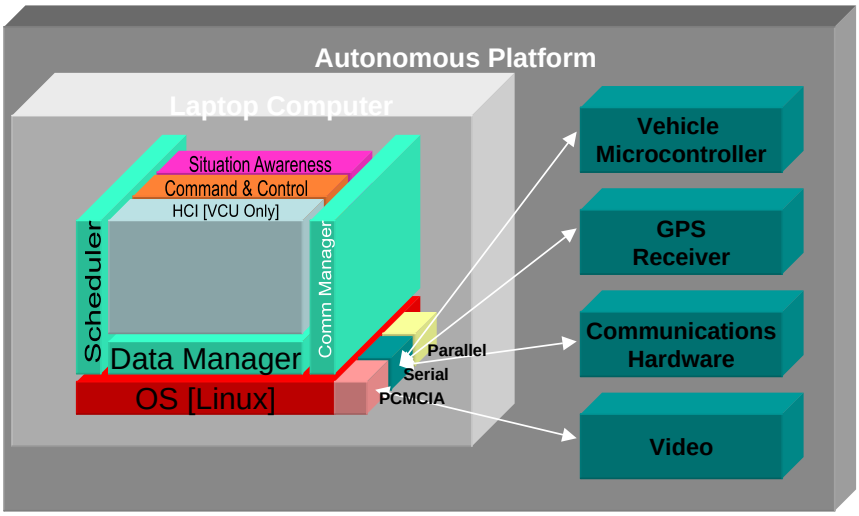


System Concept: Cooperative UAV/UGV Reconnaissance



- UAV collects & geolocates images for UGV deployment
- UGV autonomously transits from deployment to destination objective
- Mission plan for both vehicles generated & uploaded from common control station

Software Architecture



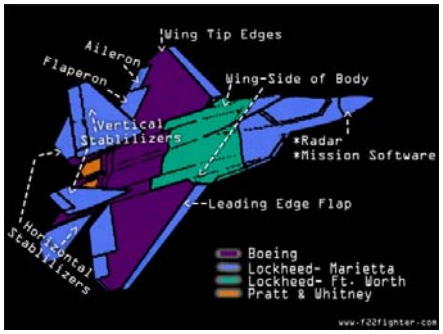
- A common, modular, reusable, component-based architecture has been developed
- Interfaces between modules are standardized so that changes can easily be made

Large Scale Software – F22

• The software that provides the avionics system's full functionality is composed of approximately 1.7 million lines of code.

• Ninety percent of the software is written in Ada, the Department of Defense's common computer language.

• Exceptions to the Ada requirement are granted only for special processing or maintenance requirements.



Data courtesy of Lockheed Martin

Large Scale Software – F22

The avionics software is to be integrated in blocks, each building on the capability of the previous block.

Block 1
• Primarily radar capability
• Contains more than 50 percent of the avionics suite's full functionality source lines of code (SLOC)
• Provides end-to-end capability for the sensor-to-pilot data flow.

Block 2
• Start of sensor fusion.
• Adds radio frequency coordination, reconfiguration, and some electronic warfare functions.

Block 3
• Encompasses full sensor fusion
• Includes embedded training capability
• Provides for electronic counter-counter measures (ECCM).

Data courtesy of Lockheed Martin

Large Scale Software – F22

Block 3.1
• Adds full GBU-32 Joint Direct Attack Munition (JDAM) launch capability
• Adds Joint Tactical Information Distribution System (JTIDS) receive-only

Block 4 software (as proposed)
• Scheduled to be integrated on the Initial Operational Capability (IOC) F-22s
• Likely to include…
  • helmet-mounted cueing
  • AIM-9X integration
  • JTIDS-send capability.

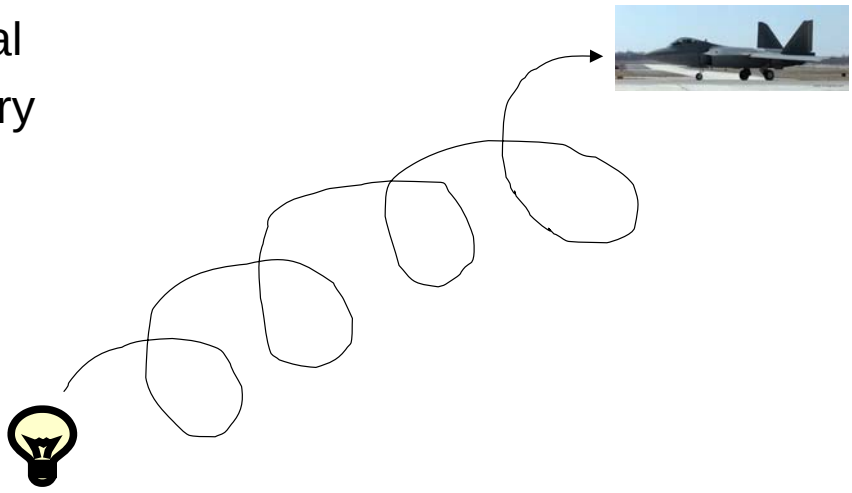
Data courtesy of Lockheed Martin

How are these applications similar?

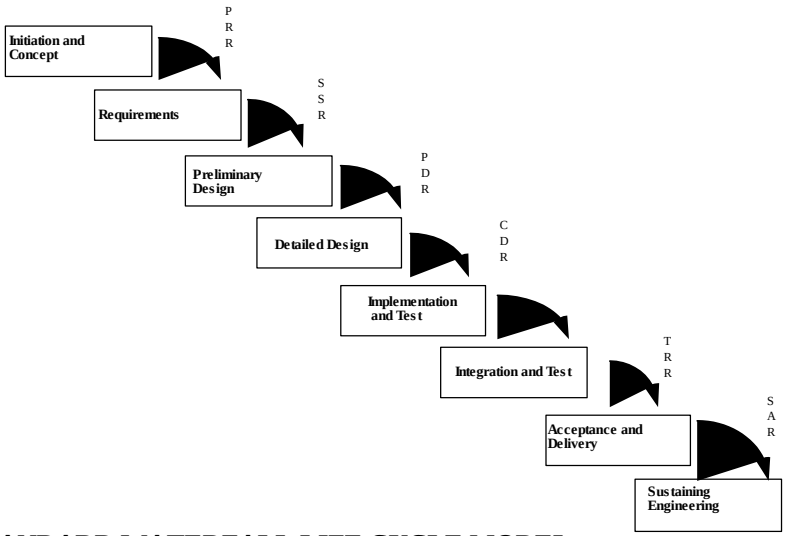
- Each requires robust, real-time software
- Development is carefully planned
- Test is an essential part of the development process

The Software Life Cycle

- Waterfall
- Incremental
- Evolutionary
- Spiral

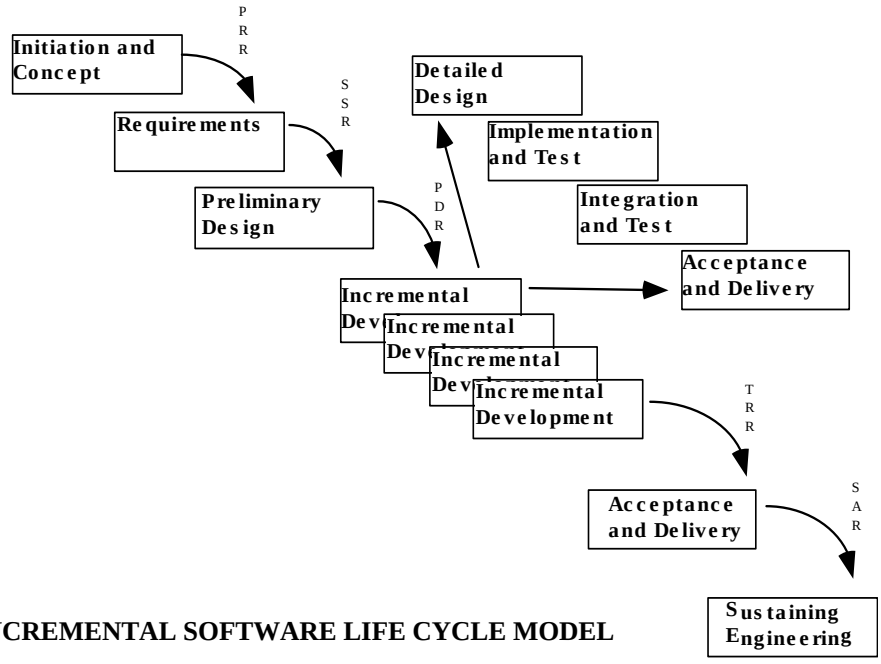


Life Cycle Choices -- Waterfall



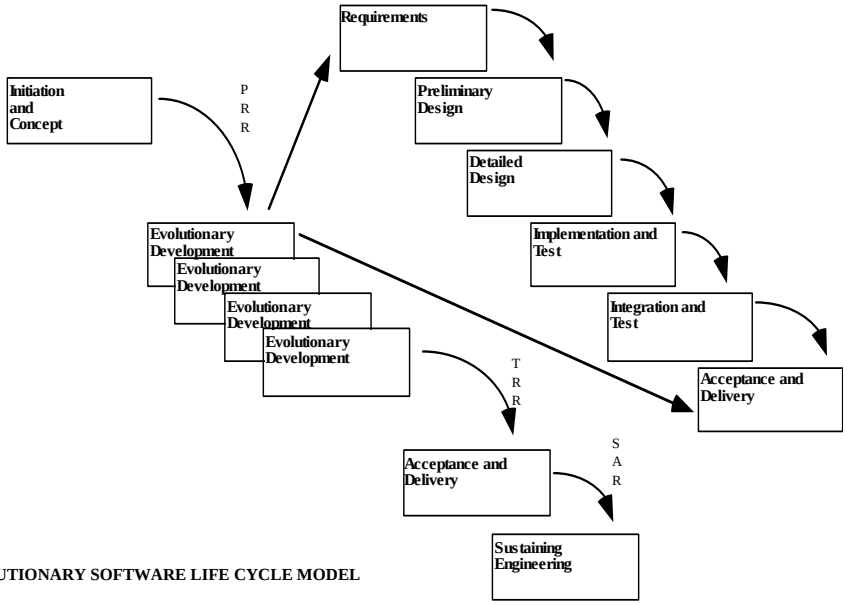
STANDARD WATERFALL LIFE CYCLE MODEL

Life Cycle Choices -- Incremental



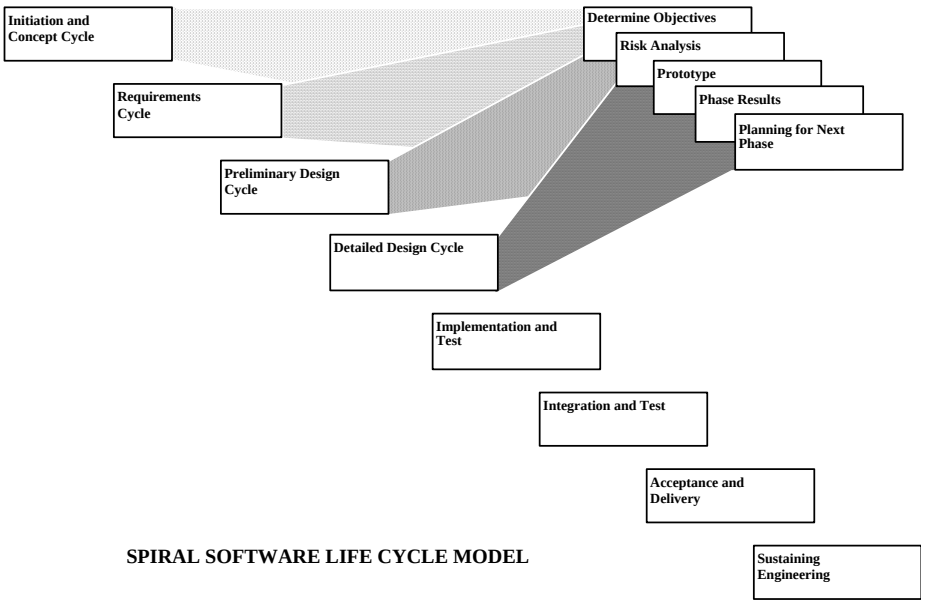
INCREMENTAL SOFTWARE LIFE CYCLE MODEL

Life Cycle Choices -- Evolutionary



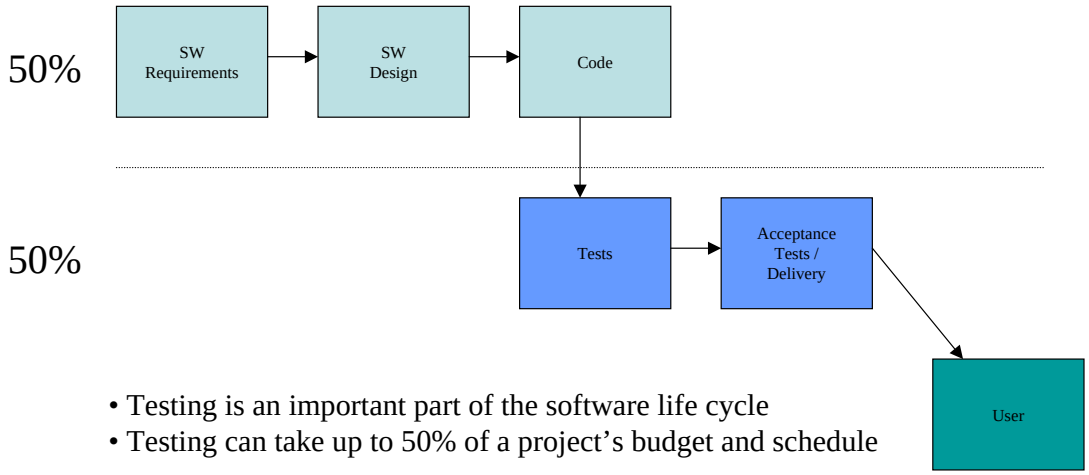
EVOLUTIONARY SOFTWARE LIFE CYCLE MODEL

Life Cycle Choices -- Spiral



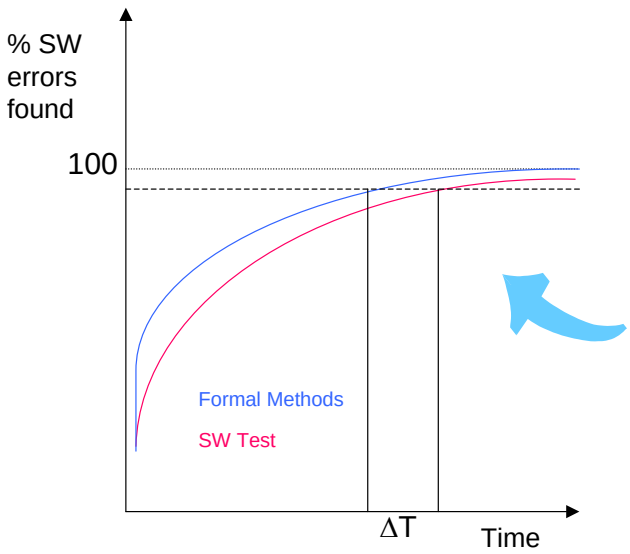
SPIRAL SOFTWARE LIFE CYCLE MODEL

Real Time System Testing



- Testing is an important part of the software life cycle
- Testing can take up to 50% of a project's budget and schedule

Why is there so much Test Test Test?



- Real Time software must be evaluated in its intended environment
- Aerospace applications are often unforgiving of failure
- More needs to be done to model real time system behavior in the design phase

Closing Remarks

- Well-developed robust software is critical to aerospace mission success
  - Robust: Mars Rover (Flash overrun issue, Priority Inversion, etc.)
  - Well Developed: A controlled software process for large scale software development is absolutely essential
- Aerospace software is fun
  - Real “toys”
  - Real world interfaces to sensor suites, etc.
  - Opportunity to collaborate with end-users to solve aerospace problems (pilots, scientists, astronauts, etc.)
- Aerospace Software Engineers are critical to systems integration and eventual mission success