Satellite Systems Software

Col. John Keesee
MIT Dept. of Aero/Astro
19 October 2003

Outline

- Definitions
- Software design process
- Design rules
- Software costing

Space System Software

Spacecraft computer systems and their software provide unprecedented capability on orbit, but drive system cost and complexity

Computer System Definitions

- Embedded System
  - Built-in processor providing real time control
- Real-Time processing
  - Handling or processing data at the time events occur
- Hard Real-Time
  - Precise timing required to avoid severe consequences

Computer System Definitions

- Soft Real-Time
  - Tasks must be completed in a timely manner, but missing a time boundary has minor consequences
- Operating System Software
  - Manages the computer’s resources (e.g. I/O, memory)
- Application Software
  - Mission-specific software related to the user instead of the support of the computer

Types Of Software

Application Software:

- Higher level functions that are provided to meet mission requirements.
  - Communications
  - Attitude and Orbit Determination and Control
  - Navigation
  - Autonomy
  - Fault Detection
  - Mission Management
  - Payload Management
- Continues to increase as requirements “creep” and problems are encountered.

Operating System Software:

- Low level functions that bridge application software to processing hardware.
  - Executive or run-time control
  - Kernel functions
  - Input/Output (I/O) device handlers
  - Built in Test (BIT)
  - Math Utilities
- Usually will not increase after CDR.

Computer Resource Estimation

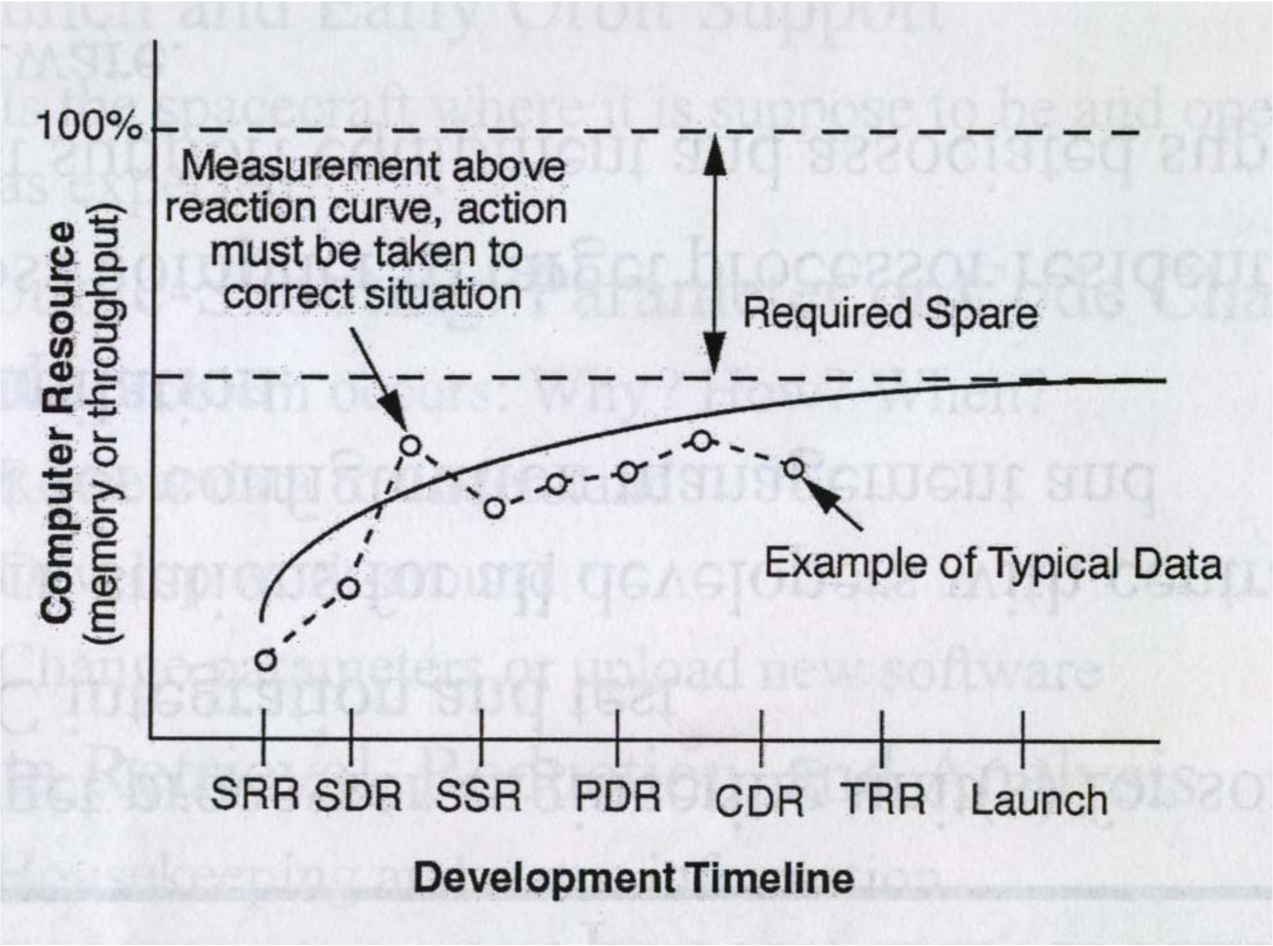
- **Define** processing tasks
  - Application software

PDR
  - Operating system functions
- **Estimate** software size and throughput
- **Establish** a cost for spacecraft software
- **Evaluate** development and test requirements
- **Create** funding profile for development and test computers and equipment.

- **Identify** life-cycle support costs
  - On-orbit operations and support

CDR
  - Upgrades, enhancements, modifications, sequence verification
- **Combine** resource/asset estimations to include all costs

Computer Resource Estimation



Functional Partitioning

- Between Space and Ground
  - Timing -Bandwidth
  - Autonomy -Human Interaction
- Between Hardware and Software
  - Performance -Complexity
  - Well-Defined Requirements -No Changes
- Between Bus and Payload
  - Duration -Quantity
  - Performance
- Alone Organizational Lines
  - Large Organizations -Small Team
  - Complexity -Accountability

Computer Systems Development Process

- Define requirements
  - Perform functional partitioning
- Allocate top-level computer requirements
  - Candidate Architechtures
  - Functional flow analysis
- Define computer system requirements
  - Define tasks
  - Establish size and throughput estimates
- Define development and support environment
- Document and iterate

Software Engineering Tasks

- Analysis
- Requirements
- Design
- Coding
- Testing and integration

- Installation and delivery
- Documentation
- Maintenance
- Quality Assurance
- Training

Software Development Environment

- Target processor engineering unit(s) for software CSC integration and test.
- Work stations for all developers with centralized host for configuration management and compilation.
- Cross compiler to target processor resident on host.
- Test support equipment and associated support software.
- Realistic I/O, associated device drivers and functional simulations, resident on host.

Software Test

- Usually begins with the lowest Computer Project Configuration Item and works up
  - Test scenarios built on the same pattern
- Increasing confidence in system performance
- Testing is a very complex activity
  - Often requires up to one half the development cost and a significant portion of the support cost

Software Integration

- Ensures all modules/classes work together
- Ensures the program works with other software packages
- Three types of integration
  - Big Bang (COTS)
  - Bottom-up (Drivers)
  - Top-down (Functional stubs)

Software Maintenance

- Critical to maintain a highly effective team for the life of the space system
- Software updates on orbit possible and likely
- Ground-based, high fidelity simulator is important for critical space systems

Life-cycle Support

- Launch and Early Orbit Support
  - Is the spacecraft where it is supposed to be and operating as expected?
- Trouble-Shooting: Parameter or Code Changes
  - If a problem occurs: Why? How? When?
  - Review data to understand
  - Develop work-around
  - Change parameters or upload new software
- Data Retrieval, Reduction, and Analysis
  - Housekeeping and status information
  - Payload and/or science data

Memory and Throughput Margin Requirements

- At SRR, size hardware to be four times the estimated requirement
- Reserve a full 100% margin (i.e., twice the delivered size and speed) at launch

Software Reliability Measures

- Watchdog timers ensure system restarts automatically if the processor stalls
- Operating system manages “Stuck” memory bits and Single Event Upsets
- High reliability systems maintain backup software images

Sixteen Critical Software Practices

- **Project Integrity**
  1. Adopt continuous program risk assessment
  2. Estimate cost and schedule empirically
  3. Use metrics to manage
  4. Track earned value
  5. Track defects against quality targets
  6. Treat people as the most important resource

Sixteen Critical Software Practices

- **Construction Integrity**

7. Adopt life cycle configuration management
8. Manage and trace requirements
9. Use system-based software design
10. Ensure data and database interoperability
11. Define and control interfaces
12. Design twice, code once
13. Assess reuse risks and costs

Sixteen Critical Software Practices

- **Product Stability and Integrity**

14. Inspect requirements and design
15. Manage testing as a continuous process
16. Compile and smoke test frequently

Spacecraft Software Costing

SMAD Chapter 20

- Costs - - (FY00$)
  - Flight Software
    $435 * lines of code
  - Ground Software
    $220 * lines of code
- Fee not included
- Language dependent (but should be tailored based on personnel experience and reuse)
  - Ada as baseline

Engineering Estimates

- Costs - - estimate hours
  -Flight Software (QA)
  6 hours / line of code
  -Ground Software
  3 hours / line of code
- Dollar estimates calculated from hours
- Amount of testing and Quality Assurance support influence costs

FIRESAT Example: Overview

- Three axis stabilized vehicle
- MIL-STD-1750A processor / Ada
- Spacecraft Bus Functions:
  - **ACS: 5200 SLOCs - -** Earth sensor, Sun Sensor, rate gyros, reaction-control thrusters, kinematic integration, ephemeris propagation, error determination.
  - **C & DH: 725 SLOCs - -** Command and telemetry processing
  - **Other: 500 SLOCs - -** Power management, thermal control
- Fire Detection using remote IR sensor
- Payload Functions:
  - Fire Detection: 666 SLOCs - - sensor processing
  - Fire Reporting: 205 SLOCs - - data reduction and transmission

FIRESAT Example: Cost Elements

SOFTWARE DEVELOPMENT

- Nearly 8,000 SLOC for on-board software
- Estimate at least 24,000 SLOC in avionics simulation software
- Estimate at least 8,000 SLOC in data collection and reduction software
- Ground station interface and simulation software

SUPPORT EQUIPMENT

- Estimate 4 Engineering Units (MIL-STD-1750)
- MIL-STD-1750 compiler and run-time kernel
- Main Frame Computer and workstations (10)
- Compiler(s), GUI builder and CM Tools
- I/O cards, card cage, and drives

FIRESAT Example: Costing

- On-Board Software Development $3.48M
  - 8,000 SLOC * $435/SLOC
- Support Software Development $7.04M
  - 32,000 SLOC * $220/SLOC
- Software Support Equipment $1.00M
  - Main Frame ($200K) / Workstations and S/W (10 * $20K)
  - Engineering Units (4 * $100K)
  - Cross Compiler ($150K) / Run-Time Kernel ($40K)
- Test Support Equipment $0.35M
  - Compilers (2 * $50K) / GUI ($50K) / CM ($50K)
  - I/O Cage (2 * $15K) / Cards (6 * $15K) / Drivers (3 * $10K)
- TOTAL COSTS: $11, 870,000.00

Application Software Size Estimates

- SMAD Table 16-13, page 665, is based on 16-bit words, a 1750A class instruction set architecture, and a higher order language
  - Communications
  - Attitude Sensor Processing
  - Attitude determination and control
  - Autonomy
  - Fault Detection
  - Other functions (power management, thermal control)

Operating System Software Sizing Estimates

- SMAD table 16-15, page 667, is based on similar systems—16-bit words and a 1750A class instruction set architecture
  - Executive
  - Run-time kernel
  - I/O device handlers
  - Built-in test and diagnostics
  - Math utilities

COCOMO

- Developed by B. Boehm (Software Engineering Economics, Prentice Hall, 1981)
- Computes the amount of effort and time to complete a software project
  - Breaks project into WBS elements
  - Requires estimate of the new lines of code required to complete each requirement

Basic COCOMO Formula

$E = a_b * K * \exp(b_b)$ (person-months of effort)

$D = c_b * E * \exp(d_b)$ (duration in months)

Where K is the total number of lines of code.

Software Project Type	a_b	b_b	c_b	d_b
Small project, experienced team, flexible requirements ("organic")	2.4	1.05	2.5	0.38
Hard real-time requirements and strict interoperability ("embedded")	3.6	1.2	2.5	0.32
A mixture of the two other types of projects ("intermediate")	3.0	1.12	2.5	0.35

http://www.jsc.nasa.gov/bu2

http://sunset.usc.edu/research/COCOMOII/index.html

Conclusion

References

- Wertz, James R. and Wiley J. Larson, Space Mission Analysis and Design, Microcosm Press, Torrance CA, 1999
- Leach, Ronald J., Introduction to Software Engineering, CRC Press, New York NY, 2000