



Software in System Engineering: Affects on Spacecraft Flight Software

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Charles (Bud) Hammons, PhD
Mary Ann Lapham
Nov 4, 2009



Agenda

Introduction

Issues Resulting from Software/Systems Relationship

- Early identification/mitigation of software engineering issues
- Traditional system engineering inappropriate to incremental development environments
- Inter-increment dependencies
- Unprecedented software integration complexity
- Software architectures

Summary

Contact Information



Introduction

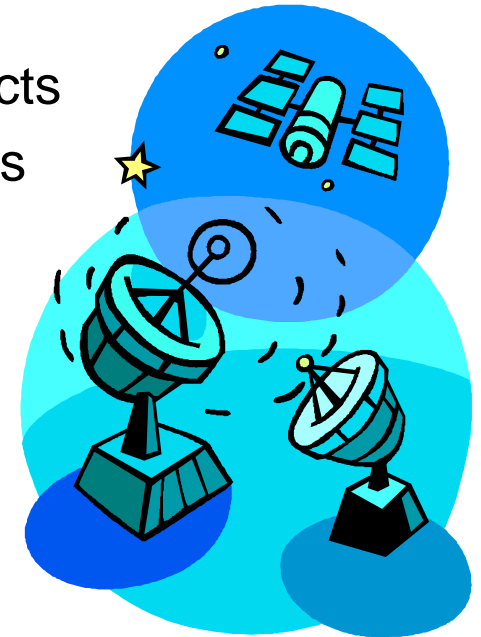
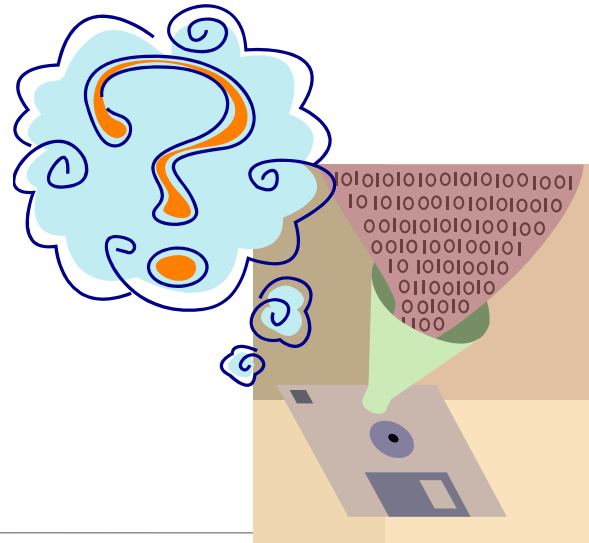


What is the relationship Software/System Engineering??

- Historically, software is considered an “implementation detail”
- Software considered late in system lifecycle
- An alternative view – software and system engineering need to work collaboratively from start of development

Not here to debate either premise but to look at impacts

Historic approach causes or exacerbates many issues



Early Identification/Mitigation of Issues₁



Software engineering issues appear or start early in programs

- Space programs typically have ground and spacecraft components/segments
- Most times segments treated separately

Miss system level software engineering issues or get misconceptions and disconnects between segments

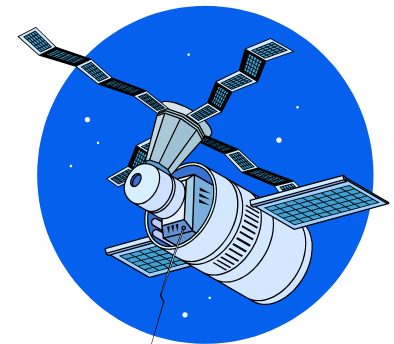
- Requirements
 - Interpretation is key
 - Segment versus system could be inconsistent resulting in non-interoperable designs and implementations



- Need system software team involved in technical oversight



= potential solution



Early Identification/Mitigation of Issues₂



Miss system level software engineering issues or get misconceptions and disconnects between segments (continued)

- Defining or declaring operational configuration
 - Different perspective from ground and spacecraft
 - Need to know static structure of system? Known defects?
 - Deployed, development, and maintenance configurations
 - Roll back information
 - Coordinated and interdependent high-level software design decisions
 - Define maintenance model early including explicit use cases to avoid inviting shortfalls in capability



Early Identification/Mitigation of Issues₃



Miss system level software engineering issues or get misconceptions and disconnects between segments (continued)

- Life-cycle costs

- Effect on software complexity, costs, etc vetted where
- Segments have different view than system
- Potential higher costs especially during operations/maintenance
- System wide role to integrate all the various software costs to ensure correctness, completeness, and consistency



- Testing

- System versus segment testing
- Spacecraft interested in edge-to-edge (within segment)
- System interested in end-to-end
- Include both positive and off-nominal paths
- Coordination of segment and system views and plans essential

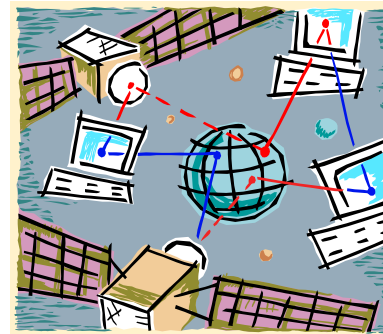


Traditional System Engineering with Incremental Environment



Merging traditional system engineering with incremental development

- Satellite constellations may be done in increments
- User expectations can change and grow between increments
- Requirement, configuration, and version interdependencies need to be well understood
- Allocation between hardware and software may change across increments
- Institute concurrent development approach
- Provide venue for cross-domain systems engineering issues



Inter-increment dependencies



Each increment considered entire system unto itself

Subsequent increments treat previous increments as legacy systems

Infers set of parallel or staggered development efforts

- Set of artifact needs
- Places unexpected strains on resources (shared or not) and development teams

Strategies vary for individual programs



Design each increment to be modifiable (solutions must be able to be modified and extended)




Unprecedented software integration complexity




Major integration risk for 10^7 or more equivalent lines of code


Risk will surface in later stage integration and test


- Usually when program vulnerable due to schedule and cost experience versus early promises

 Empanel vigorous systems integration team

- Beginning of program
- Charged to address downstream integration issues including software integration

 Institute cross-program configuration control at requirements level to aid in managing one of the primary drivers of complexity growth

 Institute test and integration teams charged with ensuring that dependencies are properly tested and existing operational capabilities not interrupted

 Test early and often



Software architectures



Multiple increments leads to potential different architectures for each increment or satellite in a constellation



Design variations may be needed to mitigate deltas – some in hardware but some in software

Coordination across prime items between segments required

- Messages exchanged across segments may have different results on various increments
- Configuration control difficult when CCB oversight spans development, sustainment, and multiple contracts

Leveraging COTS across increments may change architectures

- Variable nature of COTS releases
- Different functionality available, evolution in interfaces

  Integrate early and often

  Enable strong communication and collaboration



Summary



Early attention to software engineering is critical in reducing software risk throughout the program's lifecycle

Issues manifest in all phases of the lifecycle – requirements, development, test and integration

Solutions include:

- Need system software team involved in technical oversight from program inception
- Ensure high level software decisions are coordinated across segment and contractual boundaries
- Provide venue for cross-domain systems engineering issues, e.g. combined software and systems CCB
- Modifiable solutions
- Test and integrate early and often
- Enable strong communication and collaboration



Contact Information Slide Format

Mary Ann Lapham/Bud Hammons

Senior Member Technical Staff

Acquisition Support Program

Telephone: +1 412-268-5800

Email: info@sei.cmu.edu

World Wide Web:

www.sei.cmu.edu

www.sei.cmu.edu/contact.html

U.S. mail:

Software Engineering Institute

Customer Relations

4500 Fifth Avenue

Pittsburgh, PA 15213-2612

USA

Customer Relations

Email: customer-relations@sei.cmu.edu

Telephone: +1 412-268-5800

SEI Phone: +1 412-268-5800

SEI Fax: +1 412-268-6257

