Flight Software Development: A Manager’s Perspective

Robyn L. Haleski
Software Acquisition and Process Department

Computers and Software Division/
Software Engineering Subdivision
December 18, 2014
Overview

• Flight Software is Hard to Develop
• Managing the Development of Flight Software is Difficult
• Lessons Learned to Help Manage the Development
• Conclusion
Flight Software (FSW)

**Characteristics**

- Embedded
- Hard Real-time
- Mission Critical
- Invisible

Functioning flight software is typically required early in I&T to support integration and test of the first electrical element and the ground system.

*and*

Because of dependencies on hardware subsystem requirements, flight software is the last flight element to have mature requirements.

*and*

Flight software may change up until launch.

*Flight software is mission critical but is finalized late in the development of the flight system*
Flight Software

Why is it hard to develop?

• The flight software controls and monitors many hardware subsystems including Electrical, Power, Guidance, Navigation, and Control, Science Instruments

• The subsystems and science instruments are developed independently
  – Understand the design of subsystems and instruments
  – Participate in trade studies
  – Perform risk assessments
  – Finalize FSW requirements after all subsystems have been defined
  – Rely upon specialized hardware and software for testing

• Instrumentation is usually required to debug and test flight software

• Flight systems usually have concurrency and distributed functionality which increases complexity
Flight Software Management

Why is it difficult to manage development?

- It is invisible
- Low level details of the subsystems increase the complexity of the flight software driving the cost and schedule
- It isn’t clear when flight software is done
  - *What is a sufficient level of testing?*
  - *What types of testing are necessary?*
- Flight software engineers tend to be perfectionists
- There are competing demands for the team. The FSW team:
  - *Builds the software*
  - *Responds to questions*
  - *Supports I&T*
Flight Software Management
Lessons Learned (1 of 2)

• Use engineering rigor
  – A defined development process improves quality of product
  – Be careful, the development process can become an end in itself

• Apply KISS to process, design, and development

• Assign software to owners to improve quality

• Conduct walkthroughs to find errors

• Include systems engineers to add value to software design and code reviews

• Improve productivity by using development and coding standards

• Treat rule-based autonomy systems as flight software

• Ease system integration and future modifications with clearly defined and controlled software interfaces

• Expedites full integration with early integration of software interfaces
Flight Software Management

Lessons Learned (2 of 2)

• Use a variety of management tools
• Plan the development; i.e., create a detailed schedule
  – Have a plan and execute the plan
  – Identify and track dependencies
  – Assume problems will occur
• Define the scope and manage changes to requirements
• Identify and mitigate risks
  – Limit use of new or unfamiliar technologies
  – Create teams with an appropriate mix of experienced and less experienced personnel
• Analyze defect metrics
• Manage by walking around
• Learn from history
Flight Software Management

**Conclusion**

- There is no silver bullet
- Common sense is essential

Strong Engineering Discipline Coupled With Appropriate Processes

Reliable, Robust, Correct Flight Software
Thank you