Developing Reliable Software For A Rapid Deployment Product

ATK Advanced Weapons

Overview

Challenge:

Develop reliable software while minimizing risk for a rapid deployment product.

Approach/Goal:

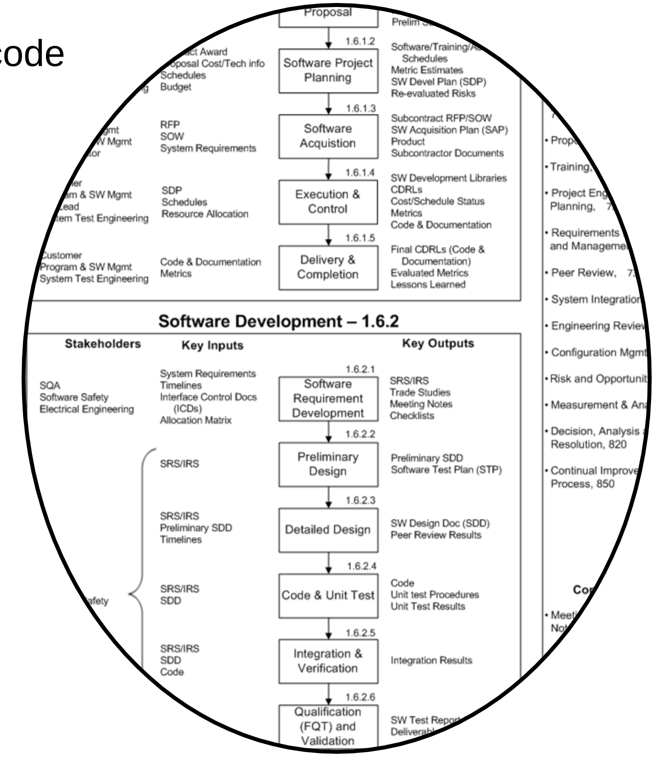
Apply simple strategies to the following standard software activities.

- **Process**
- **Design/Implementation**
- **Integration**
- **Field Test**

Process: Simplify

Use a consistent, rigorous software process that allows flexibility.

- **Simplifying process when risk is low expedites development.**
  - Generate thorough requirements without over specifying, simplify scope if possible
    - Missed requirements results in errors while excessive requirements add overhead in maintenance and testing
    - Challenge questionable requirements
  - Hold “appropriate” peer reviews
    - Broad review team for requirements, focused review for code
  - Allow parallel effort
  - Don’t apply process formality too early
  - Perform thorough integration test and analysis, focusing on most likely scenarios
    - Ensure implementation of requirements and handling of possible failure situations
- **Develop complex software incrementally**
  - Focusing on a single functionality makes it easier to specify, implement, debug, and integrate



Process: Configuration Management

Simple and effective Configuration Management is critical when development is progressing quickly. Incorrect software in a product build leads to disaster.

- **Manage software configurations without invoking a complex change control process too early**
- **Use diligent configuration management throughout software development whether software is prototype, test, intermediate distributed version, lab configuration, tactical, etc**
- **Provide unique <u>readable</u> version/build number and checksum in each software release**
- **What we do…**
  - Provide lab checkout software with unique ID but not released to CM
  - Release flight test software to CM system along with a version control document
  - Document all software version(s) in a TRR package prior to flight test
- **Change Control formality must increase as software matures**
  - Early: Fix anomalies and add functionality per test schedule / build plan, review, debug, and integrate. Limited approval required.
  - Later: Obtain CCB approval, implement and test

Process: Example

There can be several steps to the final software product while software process formality increases with software maturity.

- **High level software architectural design concurrent with software requirement development**
- **Early prototype software developed for interfacing with external components**
- **Test software developed for integration and checkout of external components**
- **Major functionality added incrementally**
  - 1: Perform pre-programmed controlled maneuvers
  - 2: Add Navigation/Guidance algorithms, program mission “manually”
    - 2a: Disable Navigation and Guidance, use for data recording purposes only
    - 2b: Enable Navigation and Guidance
  - 3: Program mission “tactically”
- **Unique version ID and CRC for each build, identified in TRR package and readable from the embedded software at test site**
- **Requirement change control initiated after requirements baseline, tactical code change control initiated after unit test complete.**

Process: Integrated Activities

Early and close involvement in all aspects of program design and development reduces risk.

- **Ensure well-justified decisions and obtain robust system understanding**
  - Participate in proposal and planning
  - Participate in processor selection
  - Interact closely with other engineering disciplines
    - Systems
    - Electrical
    - Simulation/algorithm
- **Perform early risk mitigation activities**
  - Perform trade studies for concept validation
  - Prototype software before requirements complete
- **Create environment for smooth software transitions**
  - Obtain all stakeholder input (e.g., Safety , Field Test, Production Test, Electrical)
  - Develop software on tactical breadboards
  - Involve simulation/algorithm team implementation and test of embedded software
  - Participate in system integration

Design: Software Re-use

Field tested and/or qualified software re-use is especially beneficial.

- **Benefits of continuing re-use**
  - Reuse avoids reinventing/redesign/learning time
  - Software quality increases with reuse
  - Repository of reusable software increases with design for re-use
- **Several types of re-usability**
  - Software design architecture
  - Tool and knowledge (when same family processor used)
  - External factors driving software: algorithm, electronics
  - Software: embedded, external test and maintenance, data reduction and analysis tools, subcontractor software
- **Modular design and functional decomposition promotes re-use**
  - Design for re-use

But, be careful. Drive for robustness in re-usable software without over complicating it.

Design: Re-use examples

- **Airburst derivatives**
  - Four similar programs
  - Same processor family
  - Common messaging and arming/detonation
  - Same test environment

30mm ABM Fuze

Design: Isolate safety critical software

Most weapons systems contain safety critical software for fuzing and arming functions. Isolate safety critical software into a small single purpose and well defined Software Item (SI).

- **Stanag 4404 and other safety related requirements are applied to fewer lines of code**
  - Safety related requirements result in additional lines of code and added complexity
- **Smaller SI usually means fewer future updates required**
  - Updates to safety critical software are sensitive, requiring more analysis and testing
- **Safety analysis for safety board approval is simplified**
- **Safety critical software is typically not re-programmable when fielded**
  - Software cannot be easily modified

Integration: Solid System Integration

Use a thorough set of lab integration tests prior to each field test to help ensure success. Variety of testing is a key.

- **Ensure all requirements are tested during development or integration**
- **Specifically test all updates to the release and all field test objectives**
- **Perform end-to-end functional testing**
  - Processor-in-the-loop test gives a high level of confidence that all components are integrated correctly
  - Hardware-in-the-loop testing visually demonstrates closed loop control
- **Duplicate subcontractor development/test environment for parallel integration and smooth transition**
- **Perform component test during product assembly, integrating and testing software with hardware**

While integration testing is comprehensive, it is not Formal Qualification Test (FQT). FQT must be performed prior to product delivery.

Integration Example

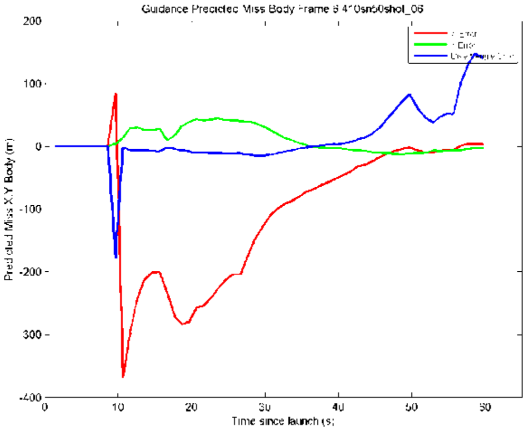
Three subcontractors provided software that required integration with ATK software. Integration continued through product build.

- **Well defined system timeline and interface specifications**
  - Software items integrated relatively smooth first time
- **Replicate development environments at all sites**
  - Facilitated working in parallel
  - Eliminated need for emulated subsystems
  - Allowed periodic integration as functionality added
  - Allowed timely delivery of software fixes as needed, key when debugging
- **Systematic detailed set of product build integration tests**
  - Ensured communication between subsystems during assembly

Field Test: Help Ensure Success

Effective field tests are critical to rapid deployment. Software results are necessary for analysis, whether test success or failure is declared.

- **Process control, design, and sufficient integration/test/analysis are keys to avoiding software induced test failure**
  - **Tests limited in number**
  - **Data is crucial**
- **Robust and detailed ground interface/telemetry/on-board recording is essential**
  - **Thorough self test and detailed reporting**
    - Key factor in go/no-go decisions for effective flight test
  - **Telemetry provides on-site real-time evaluation**
    - Key factor in go/no-go decisions for subsequent flight test
  - **On-board recorder (OBR)/Telemetry**
    - Provides invaluable 'real' flight data for system performance
    - Allows visibility into software, algorithms, and interface functions
    - Provides insight into software and/or system anomalies

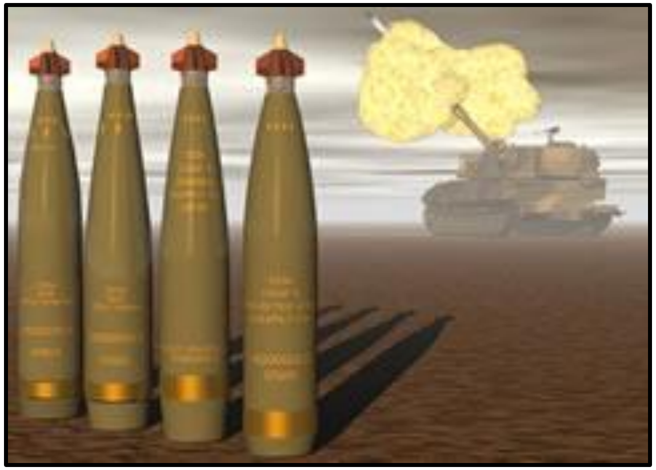


Conclusion

It is possible to develop reliable software in a rapid deployment environment.

Approach Summary:

- **Process – simplify/streamline**
- **Design/implementation – re-use, eliminate complexity**
- **Integration – don't shortchange**
- **Field Test – get the data**



PGK Field Test

Contact Information

Steve Gunderson

Software Design

ATK – Advanced Weapons Division

Plymouth, Minnesota

(763) 744-5106

Steven.Gunderson@atk.com

Mary Linman

Sr. Manager, Software Design

ATK – Advanced Weapons Division

Plymouth, Minnesota

(763) 744-5120

mary.linman@atk.com