



Adaptable System Integration on Multiple Platforms

System of Systems Engineering
Collaborators Information Exchange

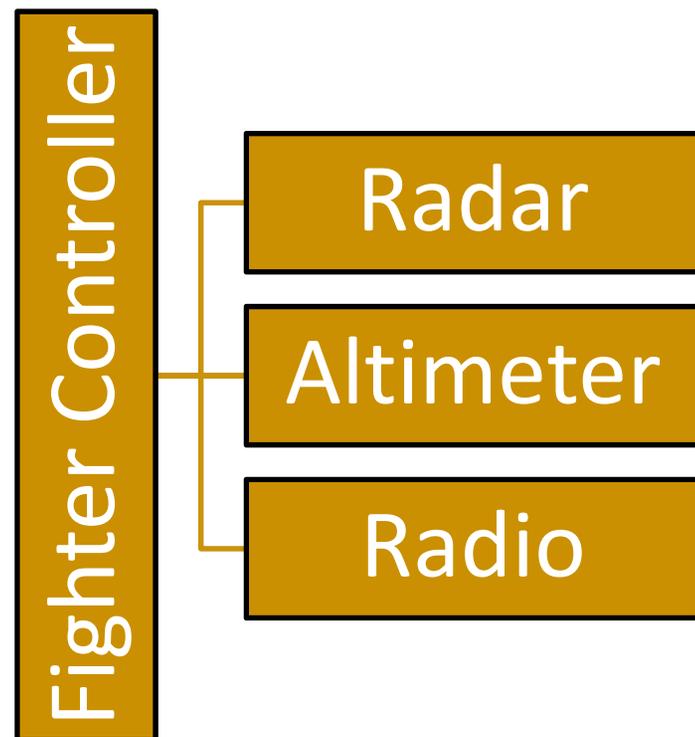
July 31, 2012

Tim Palmer
Sam Swafford

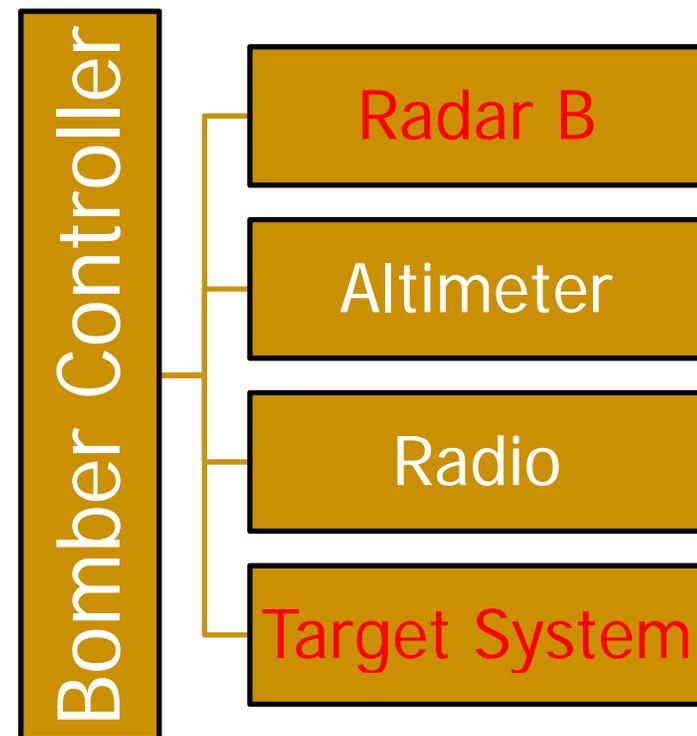
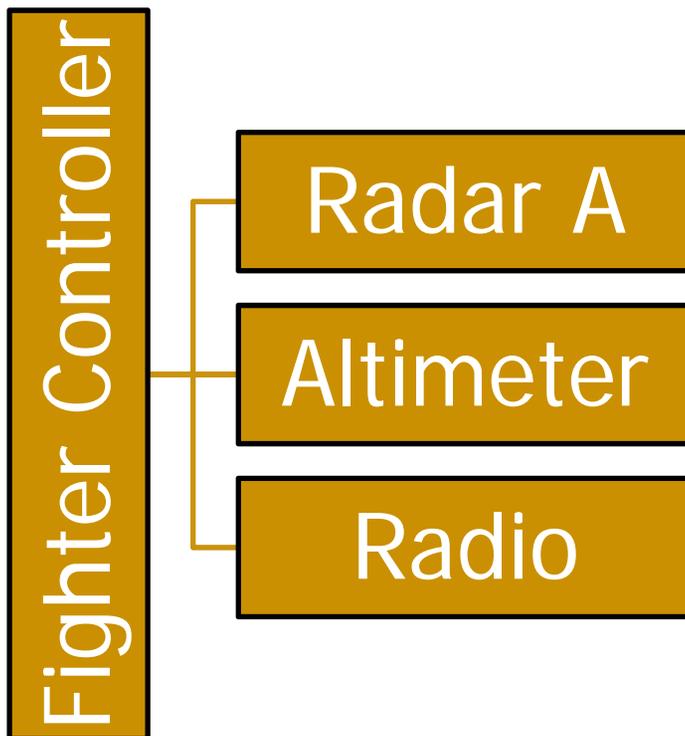
1. Problem Overview
2. Software Methods
3. Program Methods
4. Test Methods
5. Conclusion

- 1. Problem Overview**
2. Software Methods
3. Program Methods
4. Test Methods
5. Issue Tracking

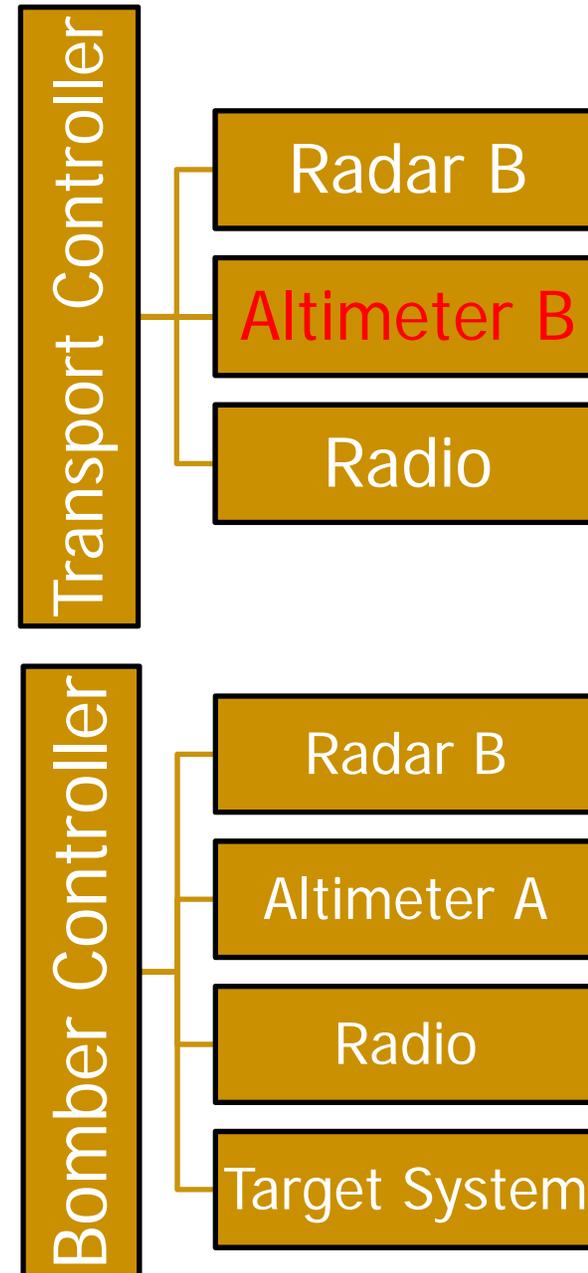
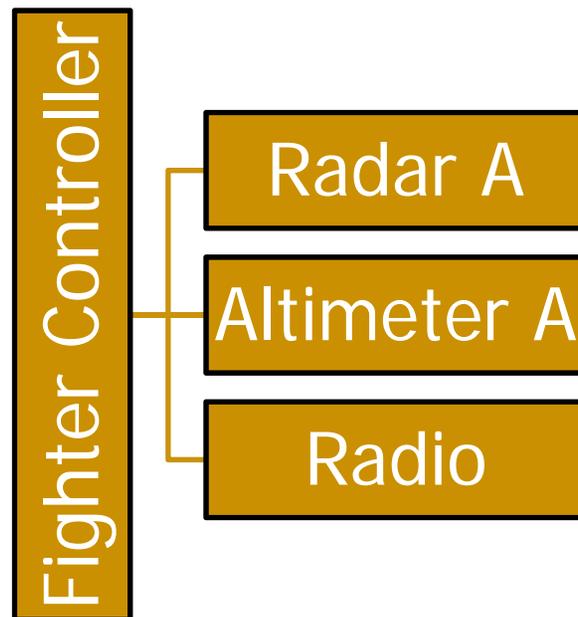
- Controller Product
 - Controls and integrates multiple systems
 - Used on a Fighter



- Bomber Integration
 - Add Targeting system
 - Add different Radar



- Transport Integration
 - Add different Altimeter

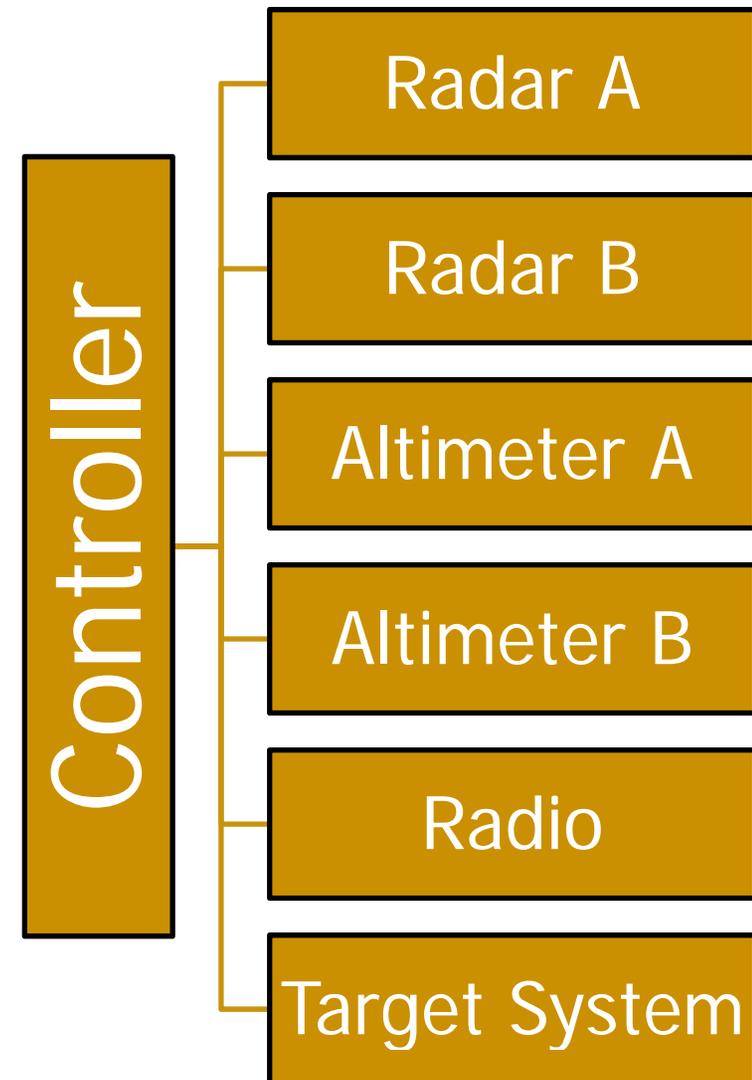


- Each new platform increases complexity and the size of the program
- How do we reduce risk, effort, and costs?
- Can the different platforms leverage capability from one another?
- Software, System, and Test methods must be considered as a whole to achieve goals

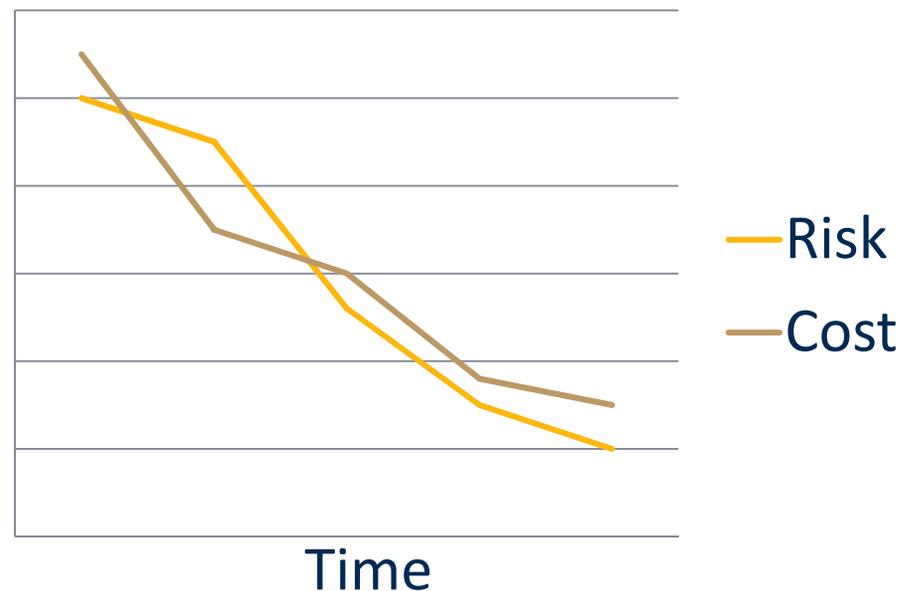
1. Problem Overview
2. **Software Methods**
3. Program Methods
4. Test Methods
5. Conclusion

- Software Practices
 - Platform checked at runtime
 - Separate software baselines
 - Platform decided at compile time
- Helpful Software Architectures

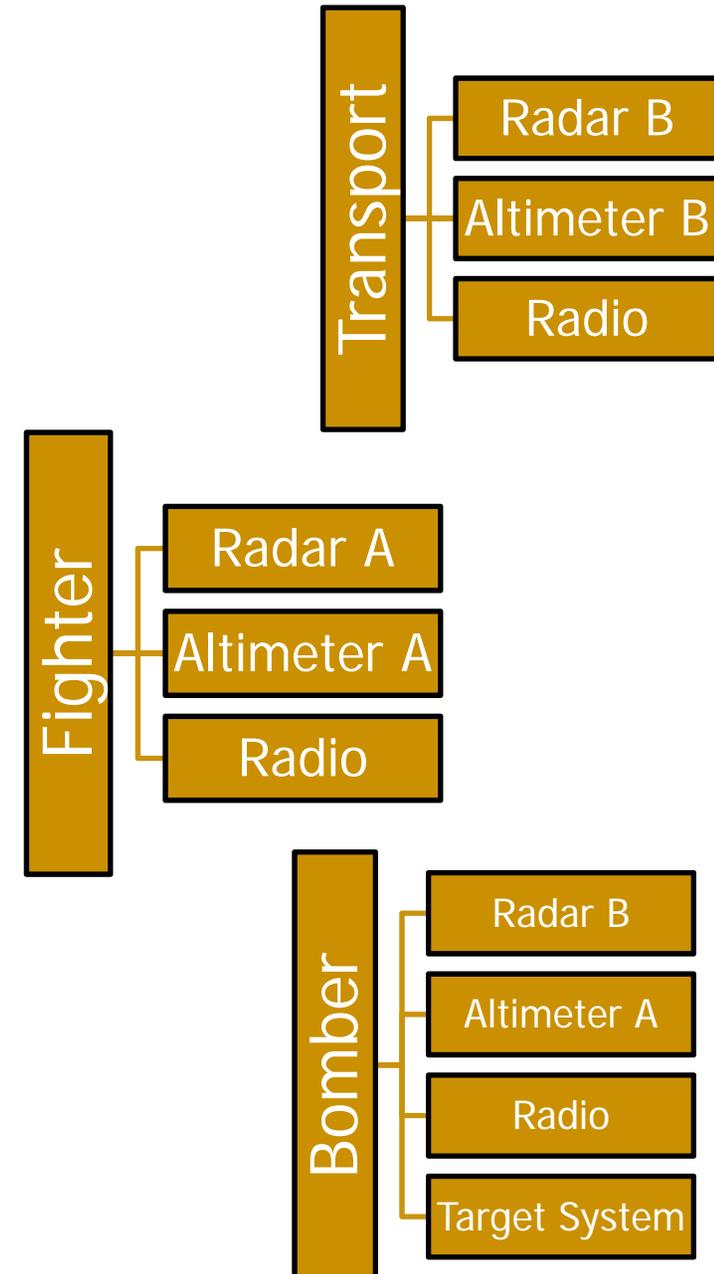
- Pros
 - Very adaptable to new platforms
 - Less software in the field
 - Changes benefit all platforms
 - Bugs will be fixed once
- Cons
 - More Complex Software
 - More processing and memory needed



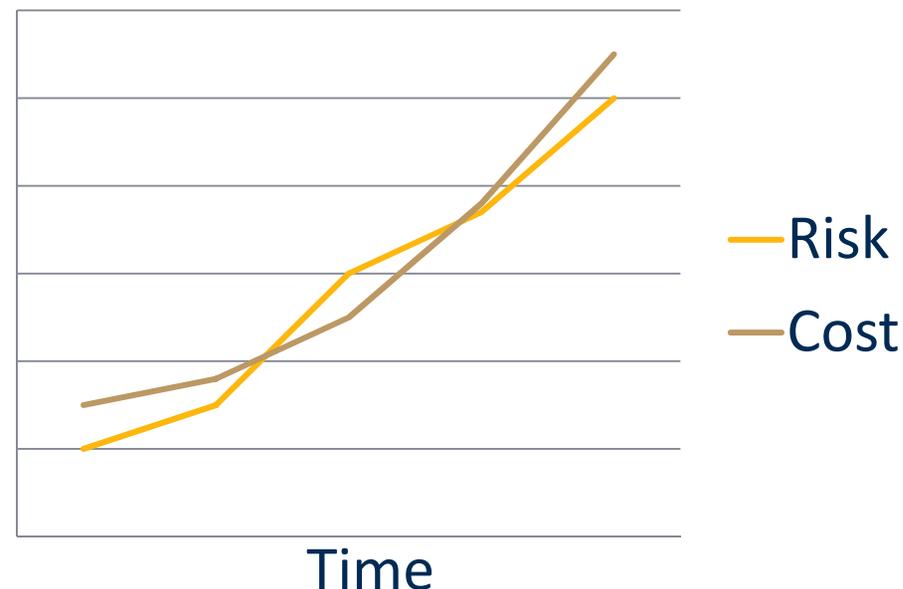
- Program Effects
 - Depot software load is operational for any platform
 - Low effort in issue tracking
- Test Effects
 - Unit level testing can be used for any platform
 - Similar tests can be created to execute for each platform



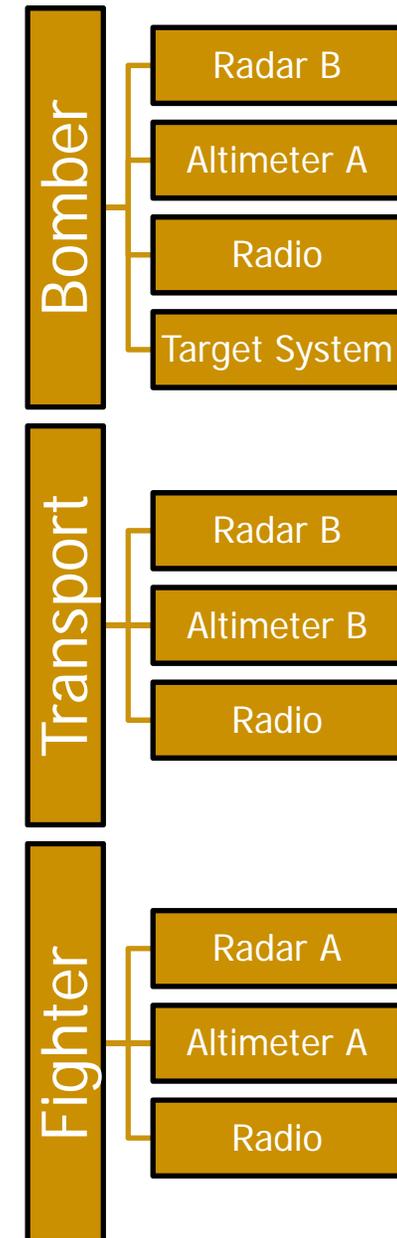
- Pros
 - Clean Code
 - Minimal memory usage
 - Minimal Processing
- Cons
 - Changes only benefit one platform
 - A bug will need to be fixed for each platform
 - Not adaptable to new platforms
 - Code could diverge into multiple designs



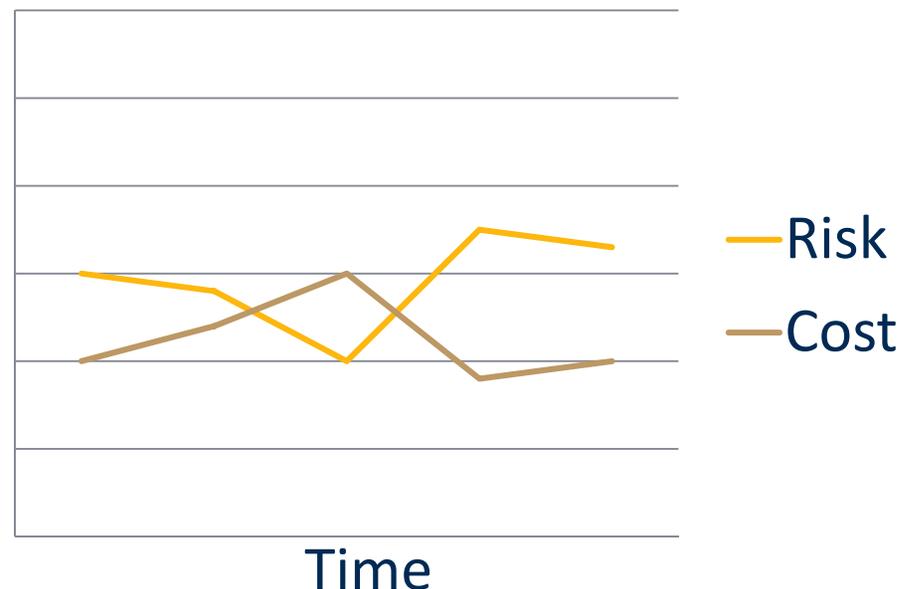
- Program Effects
 - Software must be loaded in the field
 - High effort in issue tracking
- Test Effects
 - Code inspections quicker with less code to review
 - Unique tests must be created for each platform



- Pros
 - Adaptable to new platforms
 - Changes may benefit all platforms
 - Less memory usage
- Cons
 - Multiple releases in the field
 - Changes may only benefit one platform
 - May need to fix one bug multiple times

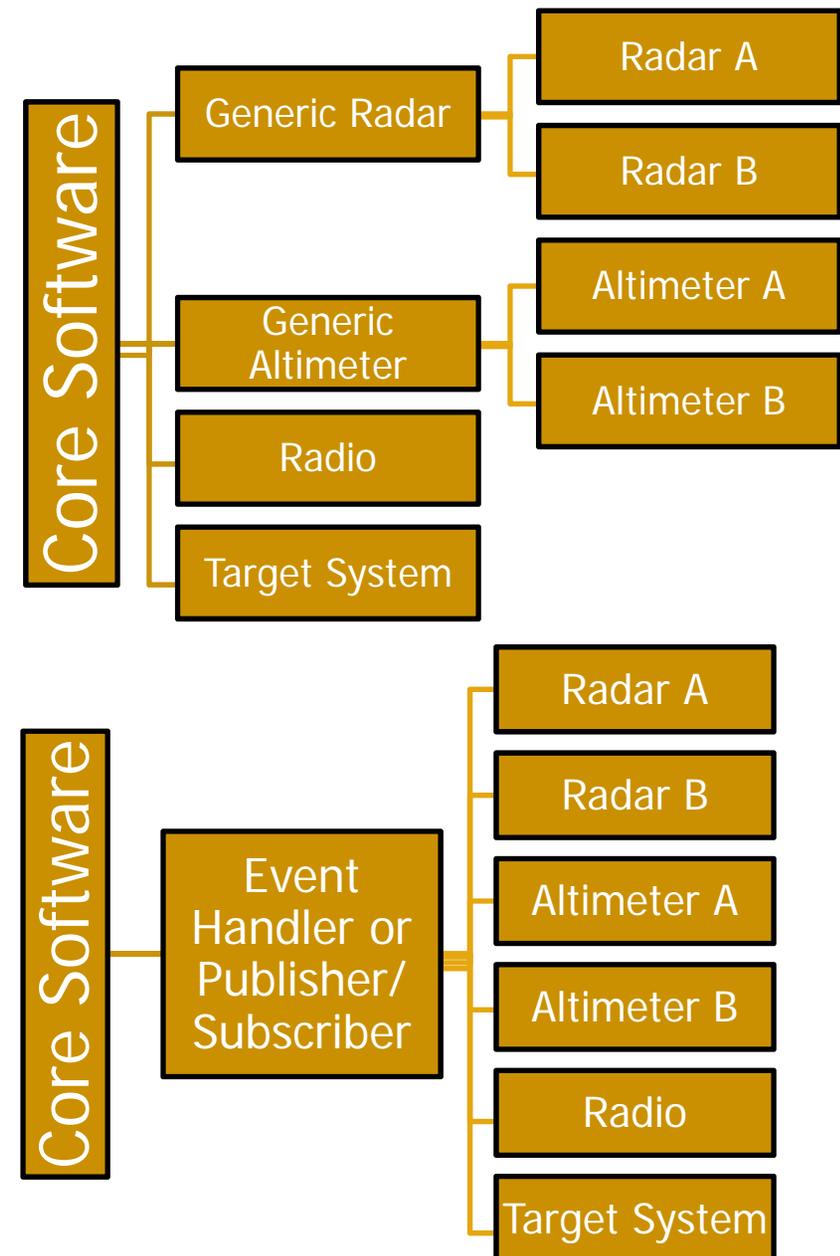


- Program Effects
 - Software must be loaded in the field
 - Medium effort in issue tracking
- Test Effects
 - Unit level testing may be used for any platform
 - Similar tests can be created to execute for each platform



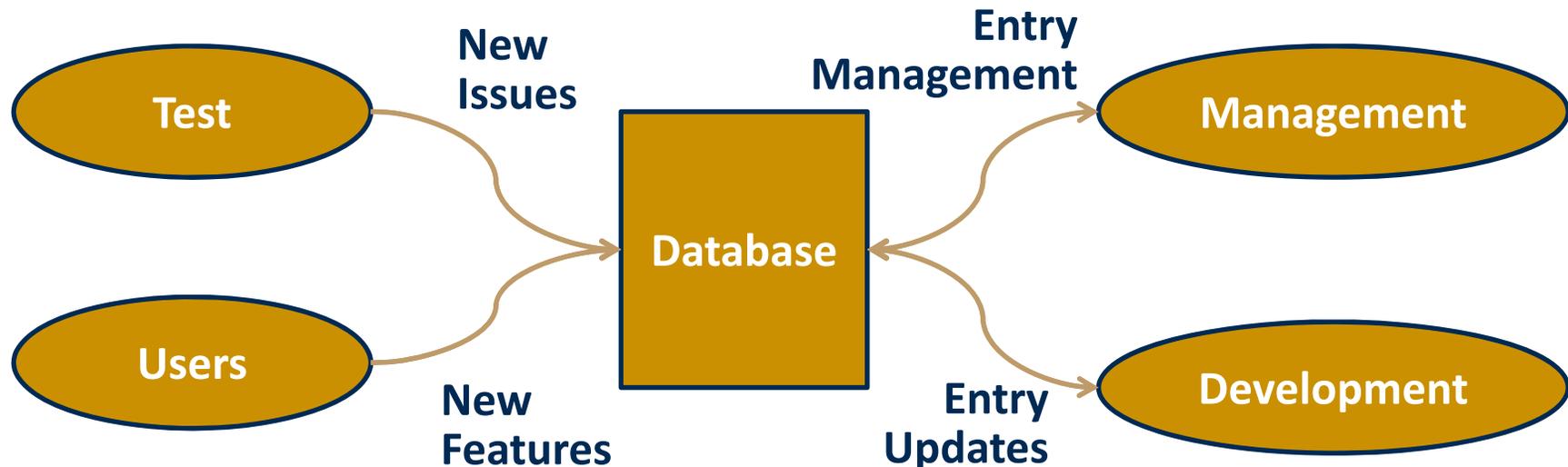
- Runtime platform decisions, while initially more risky and more expensive, have a more favorable long term outcome
- Compilation platform decisions, while initially less risky and less expensive, do not offer long term advantages
- Separate software is a tempting short term solution but is the most risky and costly method over the long term

- Separate inputs from core software
- Minimize subsystem impacts on core software
- Allows core software to easily add new systems or adapt to subsystem updates

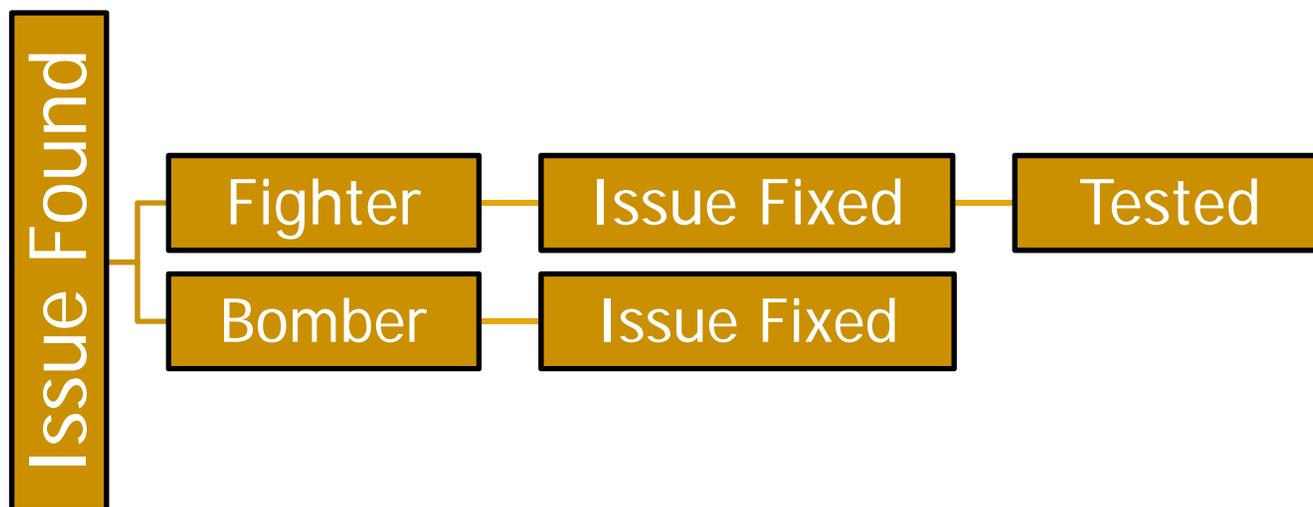


1. Problem Overview
2. Software Methods
- 3. Program Methods**
4. Test Methods
5. Conclusion

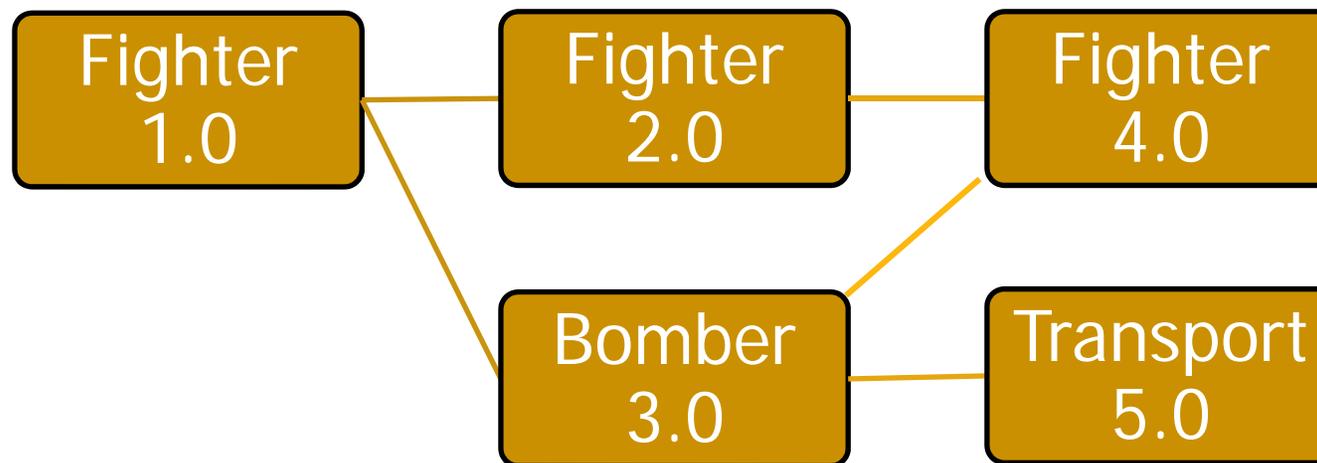
- Change Request (CR) Tracking Database
 - Users of tracking system
 - Data/metrics tracked
 - Platform found, Applicable platforms
 - Traceability from test to issue documentation



- CR Tracking Database Entries
 - One entry for each CR on each platform
 - CR is closed when software is tested for a platform, even if the fix applies to multiple platforms
 - Sibling CRs track a common change on multiple platforms



- Configuration Management
 - Development paths allow different platform development efforts to occur in parallel
 - Merge development paths to reduce the amount of variants
 - **Only for Runtime or Compile time software**
 - Configuration Management tools help manage multiple development paths



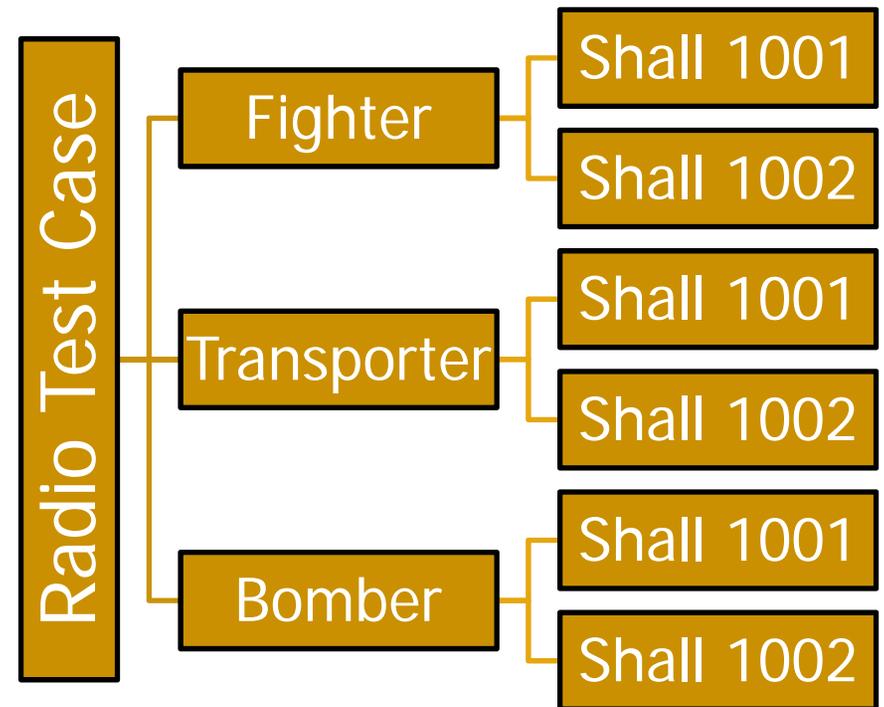
- Requirements Management
 - Use a requirements management tool such as DOORS
 - Each requirement is assigned to one or multiple platforms
 - Filters allow for one platform's requirements to be viewed
 - Used for System and CSCI level requirements

#	Description	Fighter	Bomber	Transport
10001	The system shall...	Yes	Yes	No
10002	The system shall...	No	Yes	No
10003	The system shall...	Yes	Yes	Yes

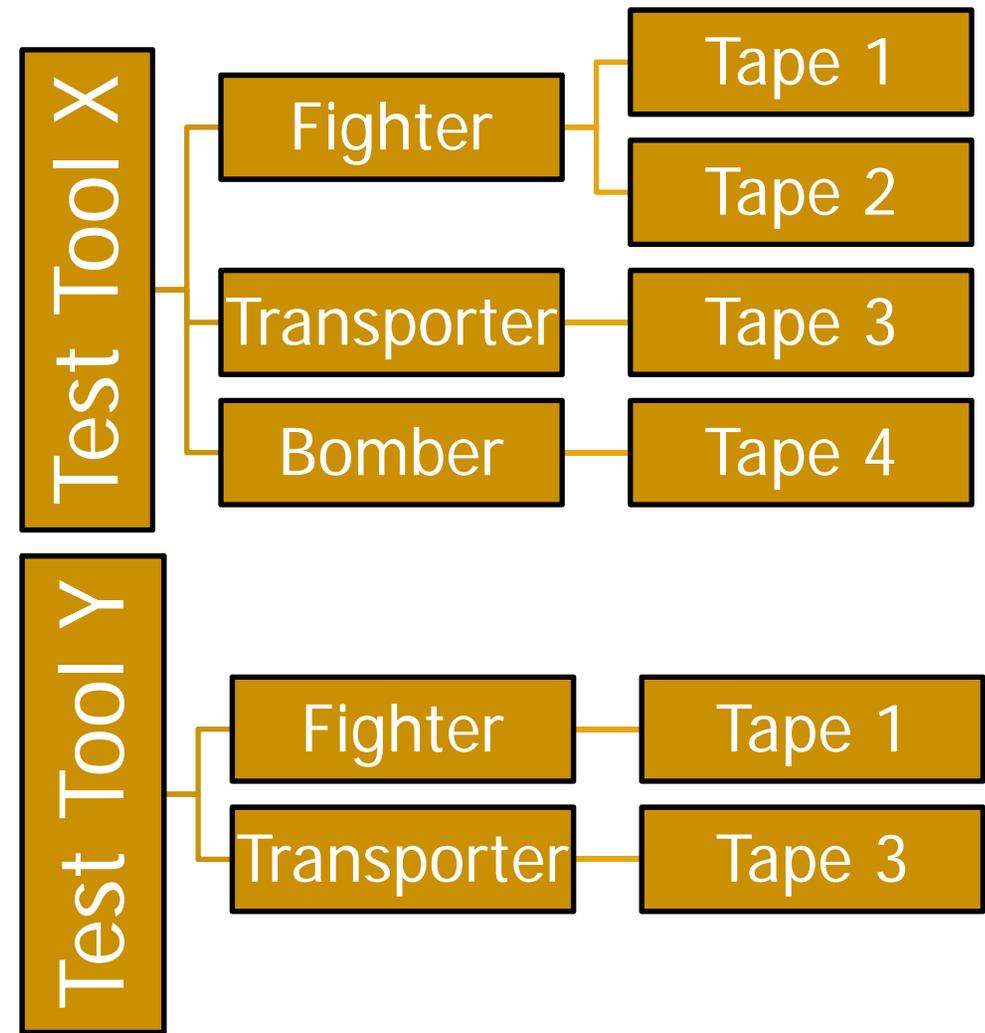
1. Problem Overview
2. Software Methods
3. Program Methods
4. **Test Methods**
5. Conclusion

- Common Functionality Test
 - Generic test cases with customizable fields to quickly transition test from one platform to another
- Test File Creation and Maintenance
 - Careful test case planning can assist in reducing effort level when converting existing test files for one platform to test another platform

- Creation of single test set to verify multiple platforms
- Requirements mapped to test cases only once



- Configuration manage files by platform, test tool type, and software version
- Utilize commonalities across platforms for test file creation



- Software, System and Test of an integrated system are interrelated components that must be considered as a whole.
- Supporting multiple platforms or configurations to leverage existing technology can be cost effective and improve common capability.



Questions?

Tim Palmer : Tim.Palmer@GTRI.gatech.edu (404) 407-7701

Sam Swafford : Sam.Swafford@GTRI.gatech.edu (404) 407-6473

