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
Problem Overview

- Bomber Integration
  - Add Targeting system
  - Add different Radar

![Diagram showing integration of Fighter Controller, Radar A, Altimeter, Radio, and Bomber Controller with Radar B, Altimeter, Radio, and Target System.](image-url)
Problem Overview

- Transport Integration
- Add different Altimeter
Overview

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

- Platform decided at runtime
- Platform decided at compile time
- Separate software baselines

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
  • High initial cost with long term low cost
  • High initial risk with long term low risk
  • 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**
  • 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
  • Medium initial cost with long term medium cost
  • Medium initial risk with long term medium risk
  • 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
Software Methods—Separate Software

- **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
  • Low initial cost with long term high cost
  • Low initial risk with long term high risk
  • 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
• Platform decisions at run time, while initially more risky and more expensive have a more favorable long term outcome

• Platform decisions at compile time, while initially less risky and less expensive do not offer long term advantages

• Separate software is a tempting short term solution, but will be 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
Program Methods

- Problem Report Tracking Database
  - One change request (CR) for each problem report on each platform
  - CR is closed when software is tested for a platform
  - Sibling CRs track a common issue on multiple platforms

![Diagram showing the process flow from Issue Found to Tested with Fighter, Bomber, and Issue Fixed stages]
• CR Tracking Database Considerations
  • Users of tracking system
  • Data/metrics tracked
    • Platform found, Applicable platforms
  • Traceability from test to issue documentation
• 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

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>Fighter</th>
<th>Bomber</th>
<th>Transport</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>The system shall...</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>The system shall...</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>The system shall...</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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
Summary

• 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