



# Analyzing the Core Flight Software (CFS) with SAVE

Dharmalingam Ganesan ([dganesan@fc-md.umd.edu](mailto:dganesan@fc-md.umd.edu))

Dr. Mikael Lindvall ([mlindvall@fc-md.umd.edu](mailto:mlindvall@fc-md.umd.edu))

David McComas ([David.C.McComas@nasa.gov](mailto:David.C.McComas@nasa.gov))



# Dave's Flight Software Group

# Fraunhofer – a short intro

- Fraunhofer Center Maryland
  - Applied research and technology transfer
  - Not for profit
  - Affiliated with the University of Maryland
    - CEO also full professor in Computer Science, UMD
  - Sister institute in Kaiserslautern, Germany
- Business model
  - Conducts applied research in software architecture, verification & validation, process improvement and measurement
  - Contract research for industry and government clients
    - Clients/partners:
      - Bosch, Biofortis, DOD, FDA, JHU, JHU/APL, NASA.....
  - Receives NSF grants in software engineering

# Context of this Collaboration

- Fraunhofer CESE received a NASA IV&V SARP grant on software architecture evaluation
- SAVE technology is partly funded by the SARP grant
- One component is outreach to NASA projects
  - Apply to various kinds of software systems
  - Get feedback, improvement suggestions
    - Technology AND Project
  - Share, publish results

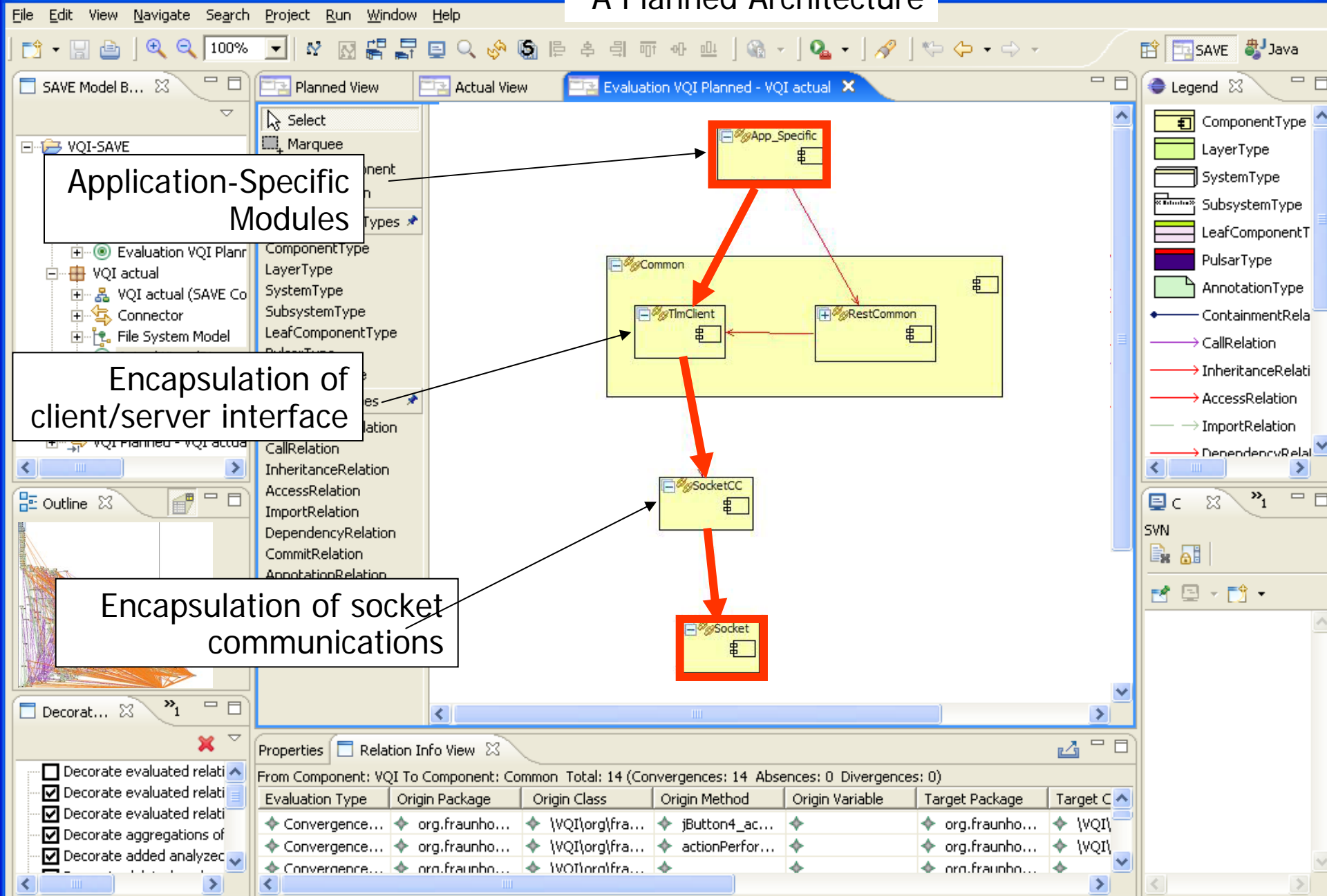
# CFS – Core Flight Software?

- CFS is project-independent flight software (FSW) that provides a runtime environment and a set of FSW applications
- Applications that comply with CFS API's can be reused for multiple missions
- CFS is designed for reuse using sound engineering principles, such as Layering, Modularity, Product Line
- **Challenge:** How to check whether CFS implementation and Applications follow the intended design rules to ensure “long-term” reuse

# The SAVE Tool

- Sample problem: How do you “understand” and “check” a larger software system?
  - Starting by looking at each line of code might not be feasible
- SAVE can automatically extract architectural views from the implementation (source code)
- SAVE can check the compliance of source code with the planned architecture (if any)
- Set of Eclipse plug-ins
- Supports C/C++, Java, Delphi, Simulink etc

# A Planned Architecture



File Edit View Navigate Search Project Run Window Help

SAVE Model B... Planned View Actual View Evaluation VQI Planned - VQI actual

Where's socket implemented?

Legend

- ComponentType
- LayerType
- SystemType
- SubsystemType
- LeafComponentType
- PulsarType
- AnnotationType
- ContainmentRelation
- CallRelation
- InheritanceRelation
- AccessRelation
- ImportRelation
- DependencyRelation

Outline

Decorat...

Properties Relation Info View

From Component: VQI To Component: Common Total: 14 (Convergences: 14 Absences: 0 Divergences: 0)

Evaluation Type	Origin Package	Origin Class	Origin Method	Origin Variable	Target Package	Target C
Convergence...	org.fraunho...	\QI\org\fra...	jButton4_ac...		org.fraunho...	\QI\
Convergence...	org.fraunho...	\QI\org\fra...	actionPerfor...		org.fraunho...	\QI\
Convergence...	org.fraunho...	\QI\org\fra...			org.fraunho...	\QI\



# The Actual Architecture vs. The Planned

SAVE Model B... | Planned View | Actual View | Evaluation VQI Planned - VQI actual

Legend

- ComponentType
- LayerType
- SystemType
- SubsystemType
- LeafComponentType
- PulsarType
- AnnotationType
- ContainmentRelation
- CallRelation
- InheritanceRelation
- AccessRelation
- ImportRelation
- DependencyRelation

Dependency in planned not in actual

Dependency in actual, not in planned

But, who does socket communicate with?

Properties | Relation Info View

From Component: VQI To Component: Common Total: 14 (Convergences: 14 Absences: 0 Divergences: 0)

Evaluation Type	Origin Package	Origin Class	Origin Method	Origin Variable	Target Package	Target C
Convergence...	org.fraunho...	\QI\org\fra...	jButton4_ac...		org.fraunho...	\QI\
Convergence...	org.fraunho...	\QI\org\fra...	actionPerfor...		org.fraunho...	\QI\
Convergence...	org.fraunho...	\QI\org\fra...			org.fraunho...	\QI\



**Fraunhofer USA, Inc**

Center for Experimental  
Software Engineering  
Maryland

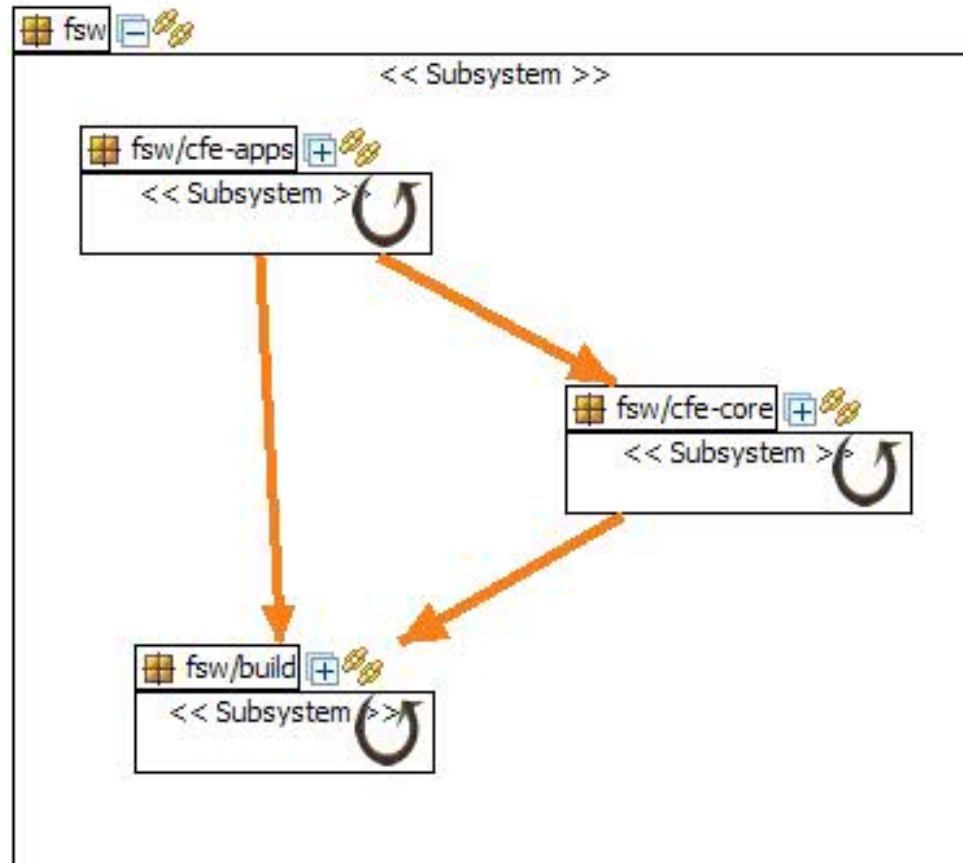
# Applying SAVE to CFS

## -A few example analyses

# Goals

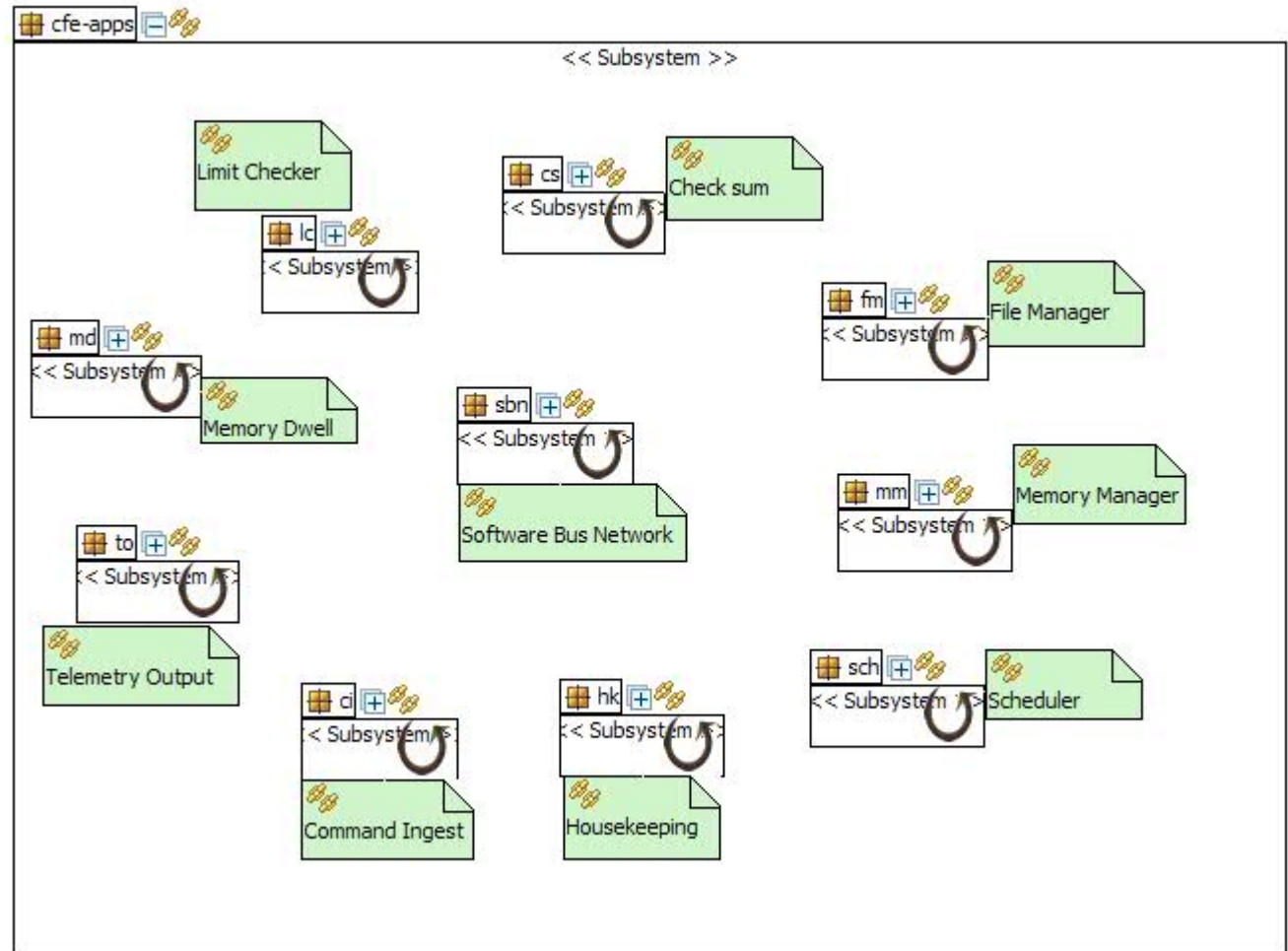
- Check if CFS implementation is consistent with design goals
- Evaluate and propose improvements of the CFS structure
- Check if all CFS applications have uniform look-and-feel
- Analyze variability potential of the CFS

# Implemented High-level View of CFS



This implemented view is consistent with the design guideline:  
*Cfe-app* should use *Cfe-core*, but not vice-versa

# Implemented View of Cfe-Apps

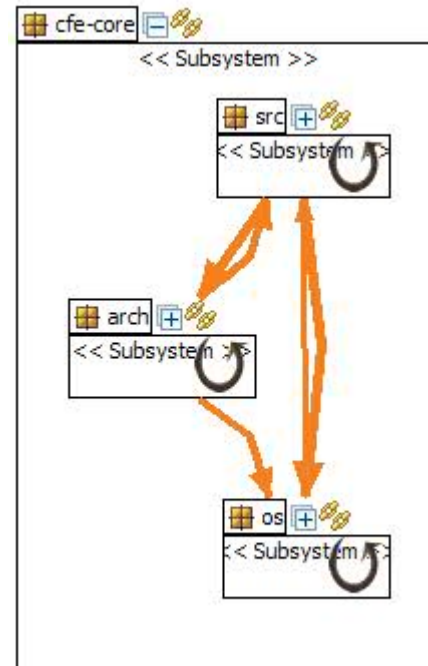


## Design Rule

No two applications are allowed to interact directly, and should instead use a bus to communicate

Yes. The code does follow the design rule

# Implemented View of CFE Core

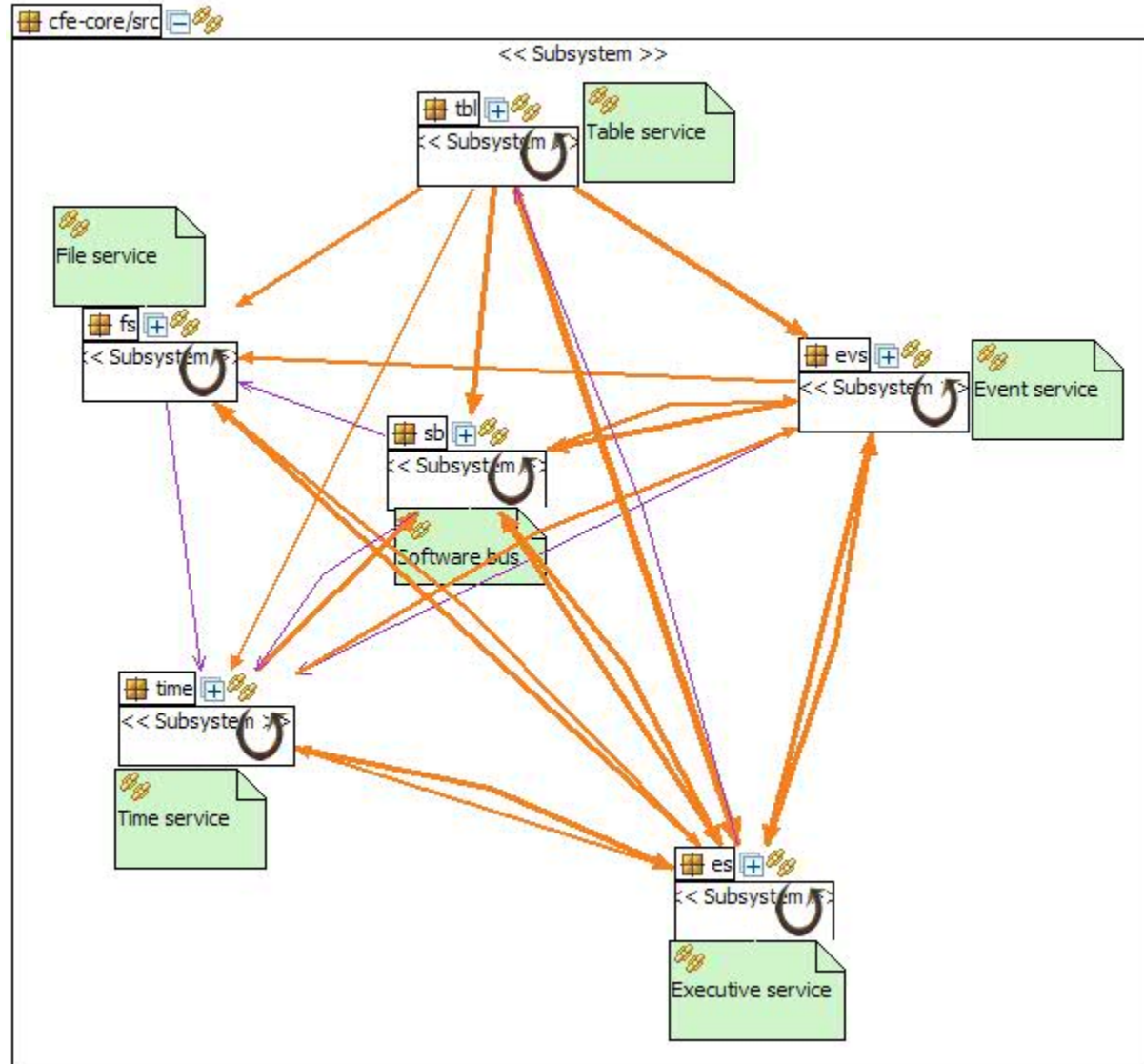


## Design Rule

Avoid cyclic dependencies (Basic design principle)

The dependency from os to src is avoidable by moving the "common\_types.h" from src to os.

# Implemented View of Cfe-core Services



## Comments

These dependencies are valid, and necessary according to the CFS team.

The SAVE analysis helped to validate the planned design

## Question:

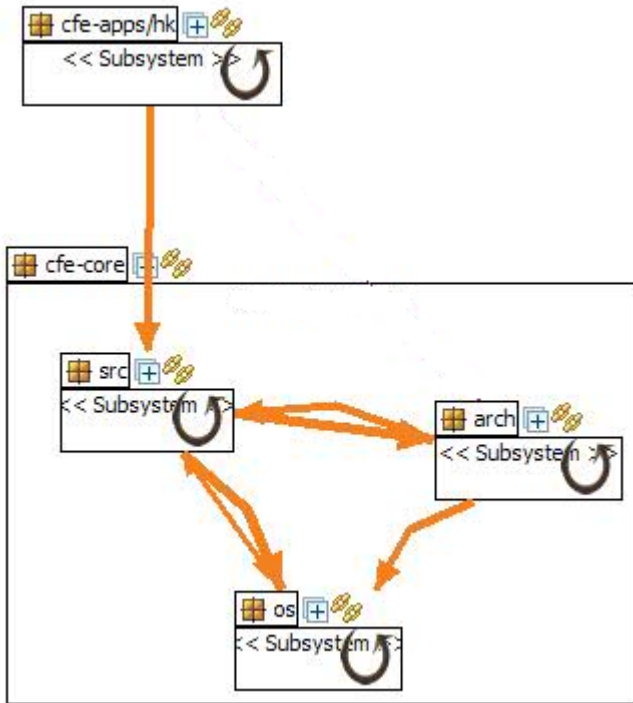
Is it possible to deliver Cfe Without table service?

# Analysis of CFS Applications



- SAVE was used to analyze dependencies from CFS apps to cfe core services
- The following applications were analyzed:
  1. HK – Housekeeping
  2. MD – Memory Dwell
  3. MM – Memory Manager
  4. CS – Checksum
  5. FM – File Manager
  6. LC – Limit Checker



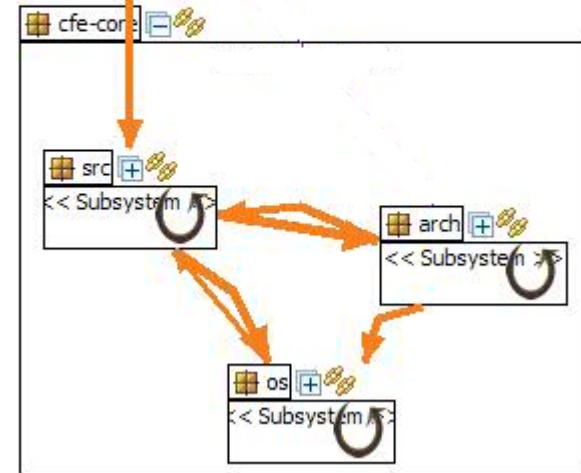
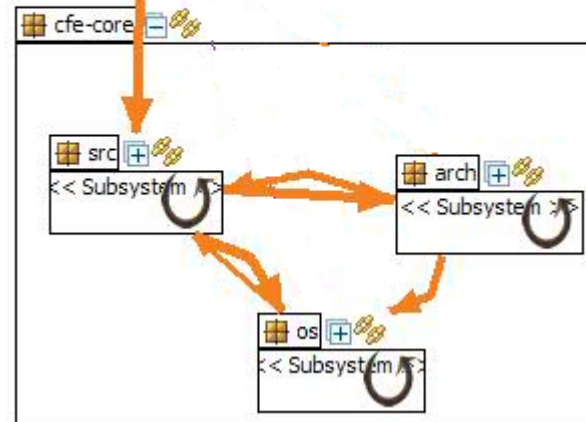
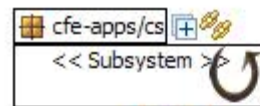
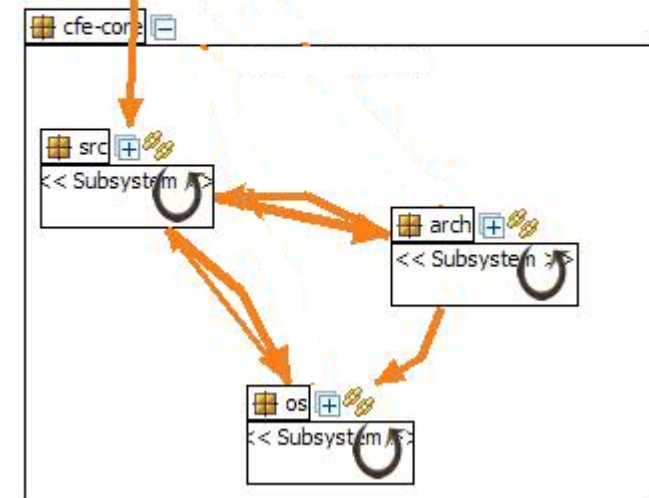
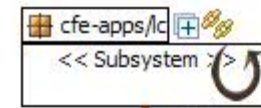
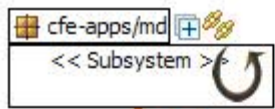
# Analysis of Applications to CFE Dependencies



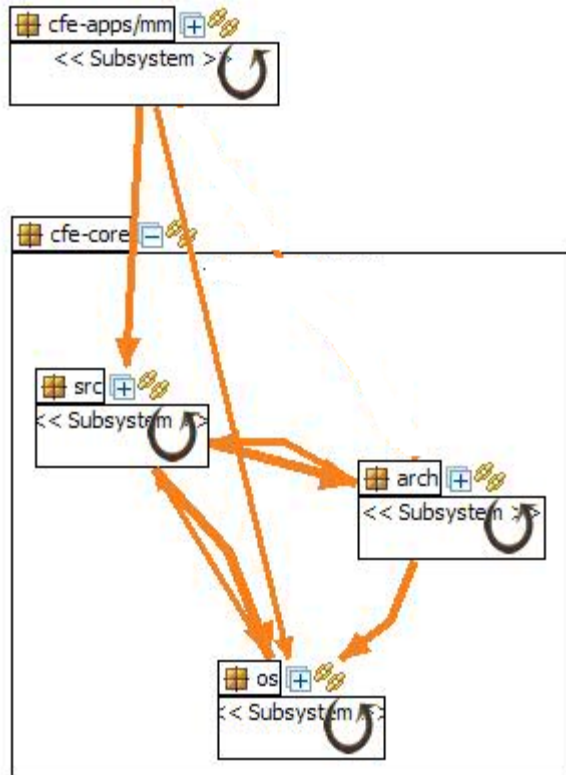
**CFS Design Rule:**  
 Applications should not directly use  
 arch and os

Origin Folder	Origin Component	Origin Routine	Target Folder	Target Component	Target Routine	Relation Type
 						
cfe-apps/hk/fsw/src	hk_utils.c	HK_TearDownOl...	cfe-core/src/evs	cfe_evs.c	CFE_EVS_SendE...	CALL
cfe-apps/hk/fsw/src	hk_utils.c	HK_SendCombin...	cfe-core/src/evs	cfe_evs.c	CFE_EVS_SendE...	CALL
cfe-apps/hk/fsw/src	hk_utils.c	HK_ProcessInco...	cfe-core/src/evs	cfe_evs.c	CFE_EVS_SendE...	CALL
cfe-apps/hk/fsw/src	hk_utils.c	HK_CheckStatus...	cfe-core/src/evs	cfe_evs.c	CFE_EVS_SendE...	CALL
cfe-apps/hk/fsw/src	hk_utils.c	HK_ProcessNew...	cfe-core/src/evs	cfe_evs.c	CFE_EVS_SendE...	CALL
cfe-apps/hk/fsw/src	hk_app.c	HK_VerifyCmdLe...	cfe-core/src/evs	cfe_evs.c	CFE_EVS_SendE...	CALL

# Analysis of Applications to CFE Dependencies ...



# Analysis of MM to CFE Dependencies



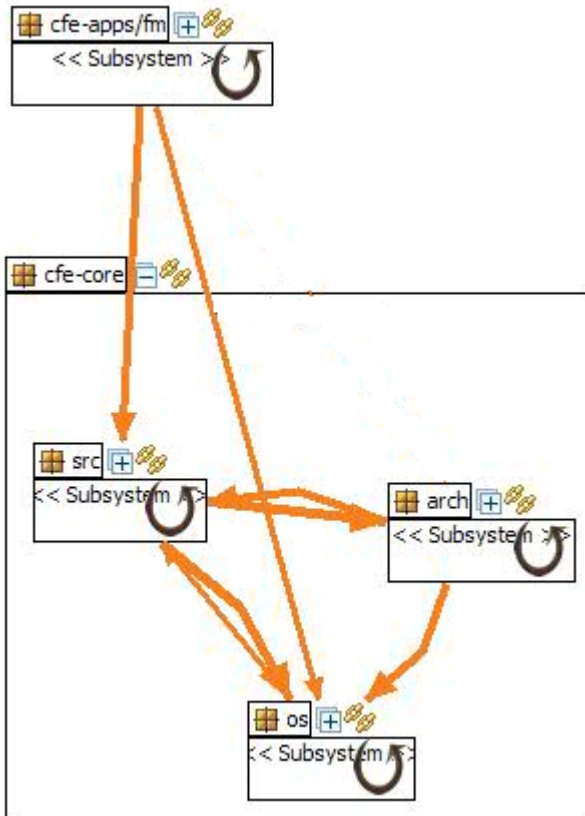
## Problem:

mm\_load.h directly uses os by directly including "osapi-os-filesystems.h"

## Solution:

Just remove that include statement. mm\_load.h already includes cfe.h which includes "osapi.."

# Analysis of FM to CFE Dependencies



## Problem:

`fm_cmds.c` directly uses `os` by directly including `"osapi-os-filesystem.h"`

## Solution:

Just remove that include statement. `fm_cmds.h` already includes `cfe.h` which includes `"osapi.."`

# Analysis of Applications to CFE Dependencies

	Executive Service (ES)	Event Service (EVS)	Software Bus (SB)	Table Service (Tbl)	File Service (FS)	Time Service
House Keeping (HK)	X	X	X	X		
Memory Dwell (MD)	X	X	X	X		
Memory Manager (MM)	X	X	X		X	
Check Sum (CS)	X	X	X	X		
File Manager (FM)	X	X	X	X	X	X
Limit Checker (LC)	X	X	X	X		X

- All applications are directly using:
  - ❖ Executive service to initialize
  - ❖ Event service for communication
  - ❖ Software bus to send/receive messages
- However, we still need all cfe services because Es, Evs, and SB depend on Table, File and Time Service
- More analysis is needed to validate and introduce appropriate Variability management technique

# Conclusion and Future Work

- CFS implementation does follow its planned design
  - There are some deviations from the design which needs further analysis
- By SAVE analysis, the distance between design and code can be significantly reduced!
- Future Work:
  - Dynamic dependencies among applications will be extracted using runtime execution and analysis of logs
  - Ordering of messages among applications will have to be analyzed
  - Timing information will be collected to check and resolve bottlenecks due to the interaction through message bus