Modularizing a DO178-Compliant Helicopter Simulink System

Stephen Scott

Table of Contents

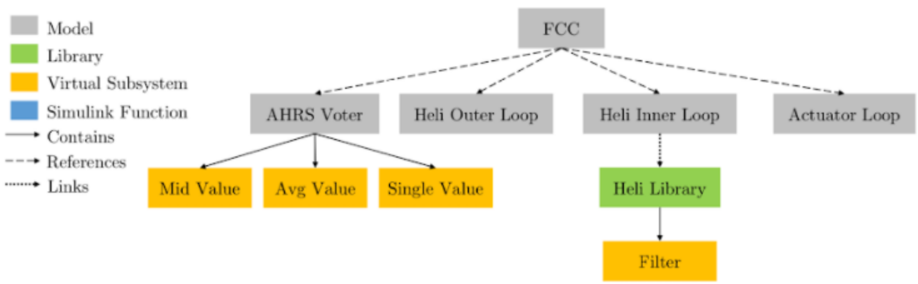
1. Summary of Changes

1.1 Notes

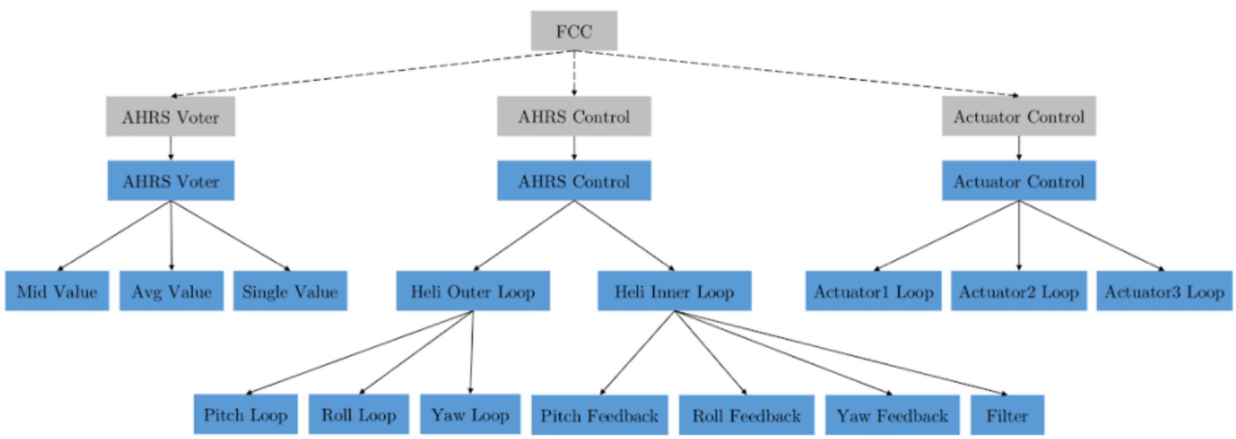
- Filter Simulink Function was called 3 times which is why Function Callers is 2 greater than Simulink Functions
- 2 of the model references (HOLC/HILC) were merged into 1 model reference (AHRS Control)
- The Actuator Loop model was referenced 3 times in the original system, so there were 4 *unique* model references, and 6 model references in total
- The ‘SOF’ matrix gain in the Heli inner loop was split into 3 separate gains
- Unit-delay in the roll-off filter was moved outside of the subsystem to allow for function reuse

Block	**Original**	**New**	**Difference**
Model References	6	3	-3
Library Links	3	0	-3
Simulink Functions	0	18	+18
Function Callers	0	+20	+20

1.2 System Structure

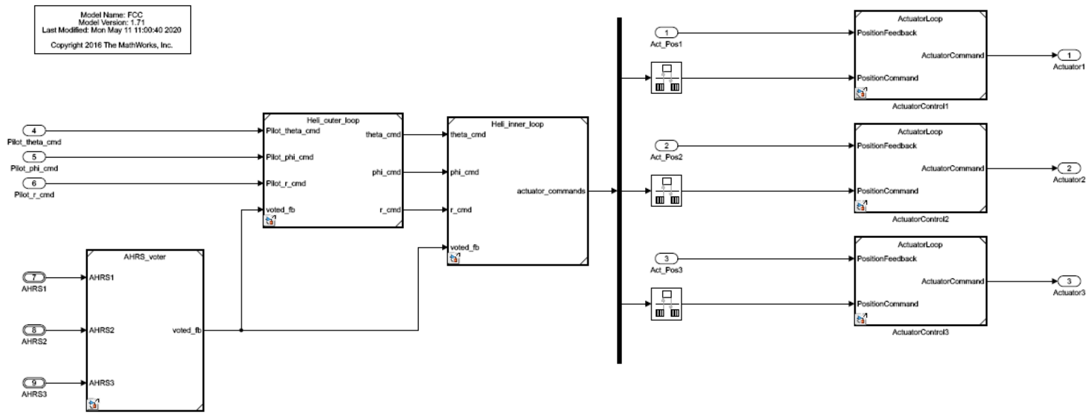


(a) Structure of the original FCC decomposition.

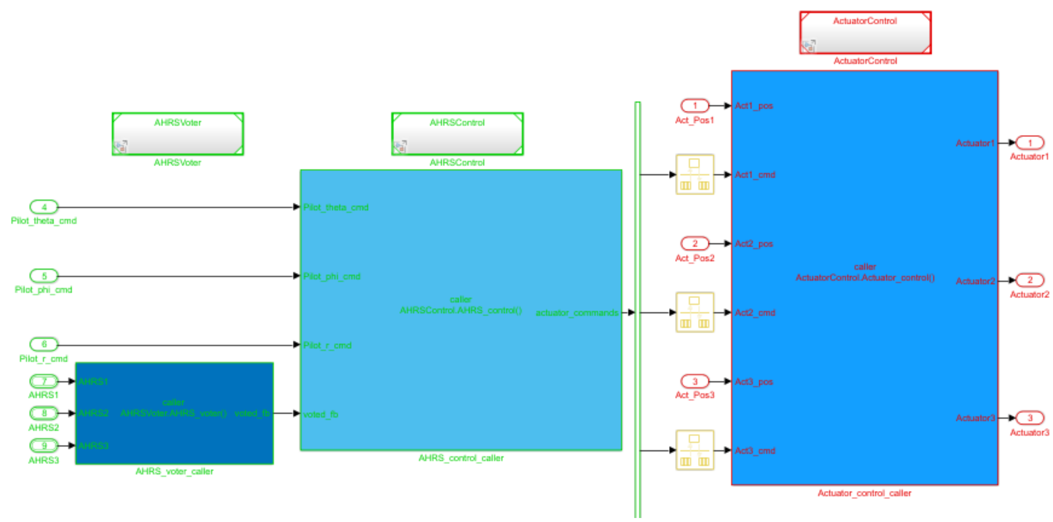


(b) Structure of the new FCC decomposition.

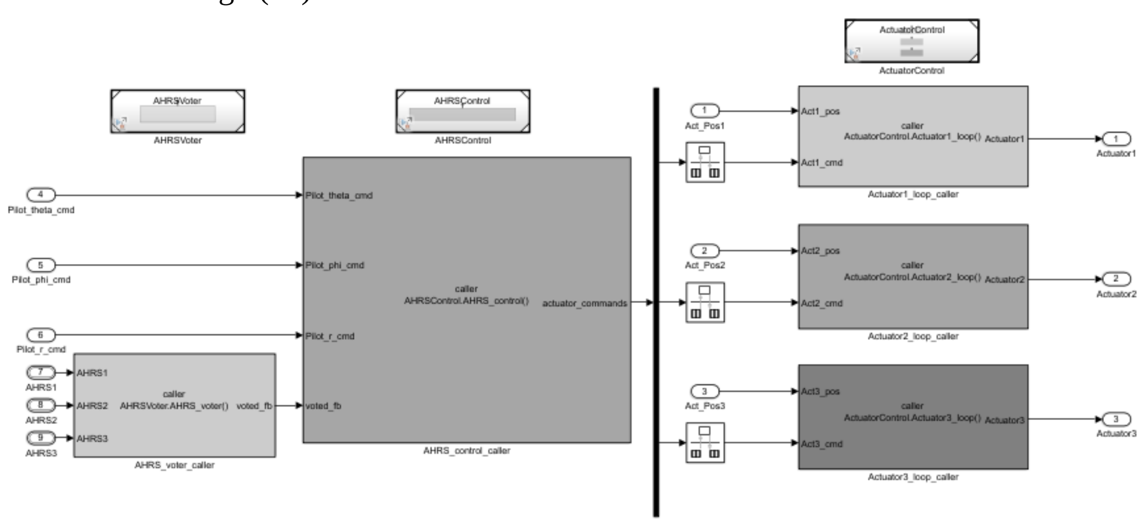
1.2.1 Original Design



1.2.2 New Design



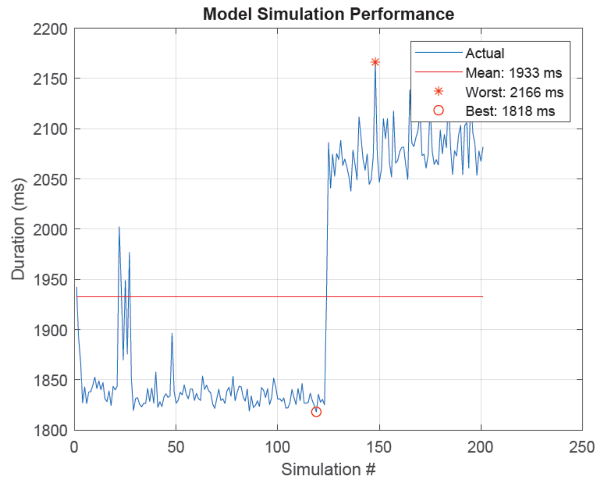
1.2.3 New Design (v2)



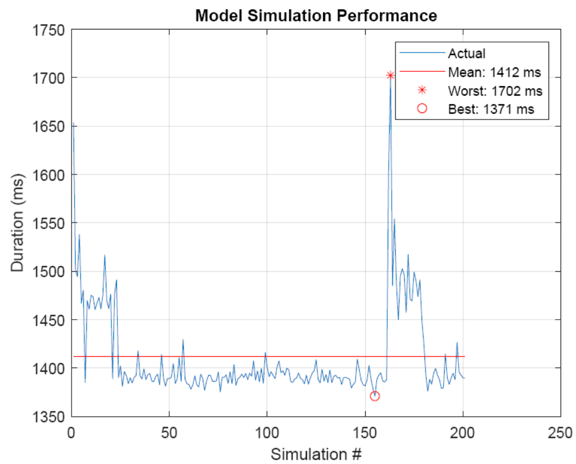
2. Analysis

2.1 Model Execution Time Comparison

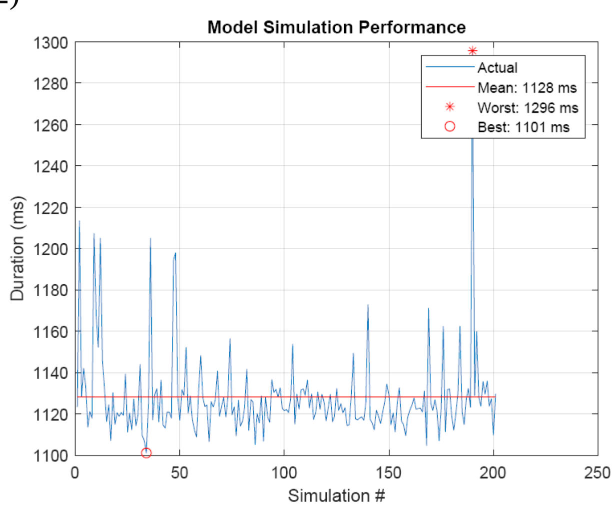
2.1.1 Original FCC



2.1.2 New FCC

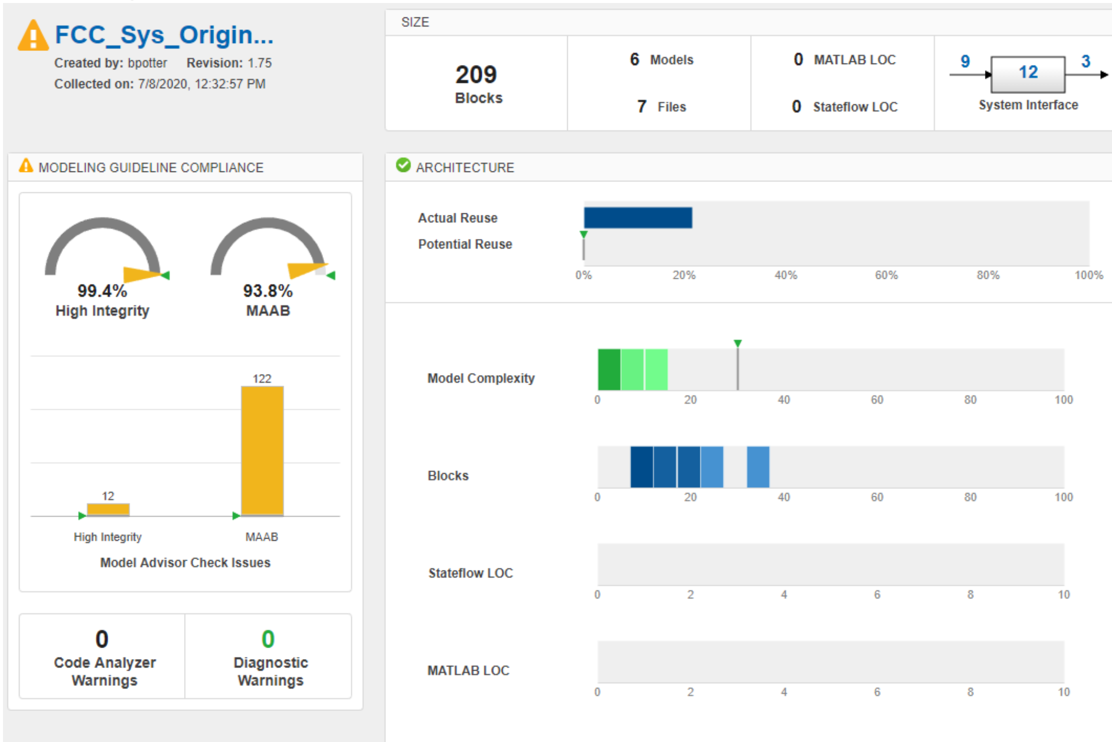


2.1.3 New FCC (v2)

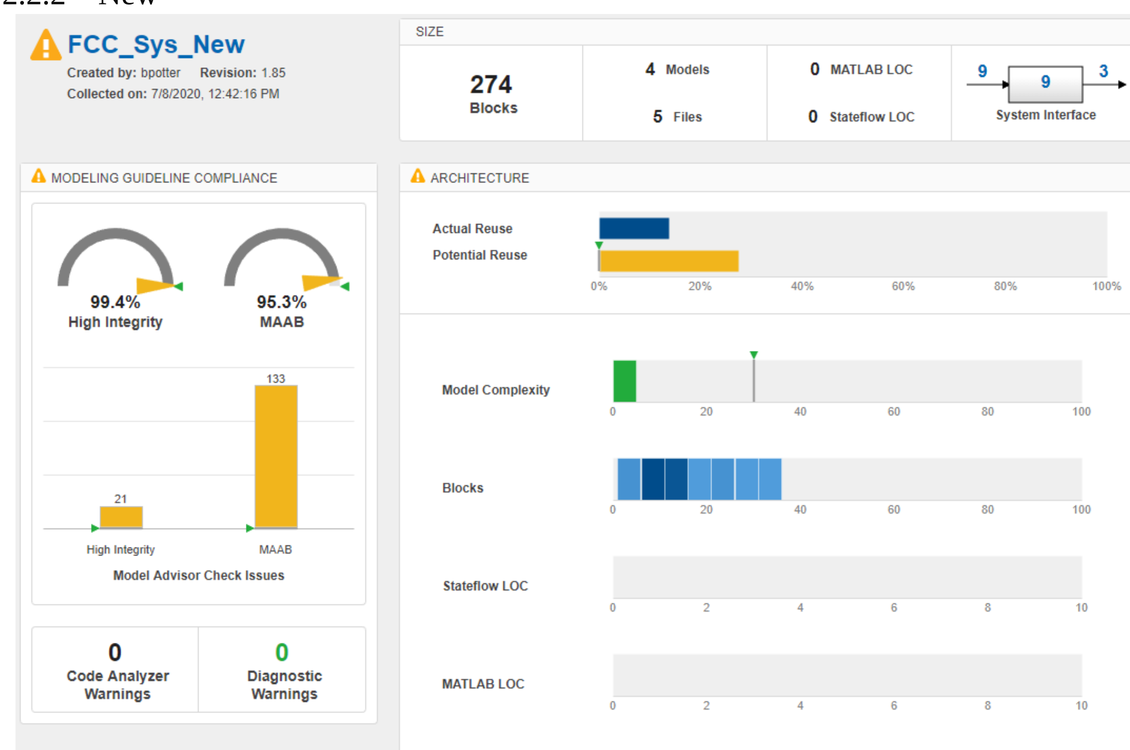


2.2 Metrics Dashboard

2.2.1 Original



2.2.2 New



3. Module Guide

3.1 New FCC

- The principle of information hiding guides the decomposition of a system into modules
- Each module should hide a likely change, or “secret”
- Secrets can pertain to the hardware-hiding, behaviour-hiding, and software design decisions
- **Bold** – exported function, *Italic* – local function

Module	Secret	Module Type
AHRS Voter	Sensor voting algorithm	Behaviour-hiding, Hardware-hiding
AHRS_voter	-	-
Mid_Value	Algorithm condition	Behaviour-hiding
Avg_Value	Algorithm condition	Behaviour-hiding
Single_Value	Algorithm condition	Behaviour-hiding
AHRS Control	Controller algorithm	Behaviour-hiding, Hardware-hiding
AHRS_control	-	-
Heli_outer_loop	Pilot control algorithm, scaling, saturation limits	Behaviour-hiding, Hardware-hiding
Pitch_loop	Pilot theta controller	Software Design Decision
Roll_loop	Pilot phi controller	Software Design Decision
Yaw_loop	Pilot r controller	Software Design Decision
Heli_inner_loop	Command control algorithm	Behaviour-hiding
Filter	Derivative noise filter	Behaviour-hiding
pitch_feedback	theta command controller	Behaviour-hiding
roll_feedback	phi command controller	Behaviour-hiding
yaw_feedback	r command controller	Behaviour-hiding
Actuator Control	Actuator control algorithm, scaling, saturation limits	Behaviour-hiding, Hardware-hiding
Actuator_control	-	-
Actuator1_loop	Actuator 1 controller	Behaviour-hiding
Actuator2_loop	Actuator 2 controller	Behaviour-hiding
Actuator3_loop	Actuator 3 controller	Behaviour-hiding

3.2 New FCC (v2)

Module	Secret	Module Type
AHRS Voter	Sensor voting algorithm	Behaviour-hiding, Hardware-hiding
AHRS_voter	-	-
AHRS Control	Controller algorithm	Behaviour-hiding, Hardware-hiding
AHRS_control	-	-
Filter	Derivative noise filter	Behaviour-hiding
Actuator Control	Actuator control algorithms	Behaviour-hiding, Hardware-hiding
Actuator1_loop	Actuator 1 controller	-
Actuator2_loop	Actuator 2 controller	-
Actuator3_loop	Actuator 3 controller	-

4. Simulated Changes

4.1 AHRS Optimal Controller

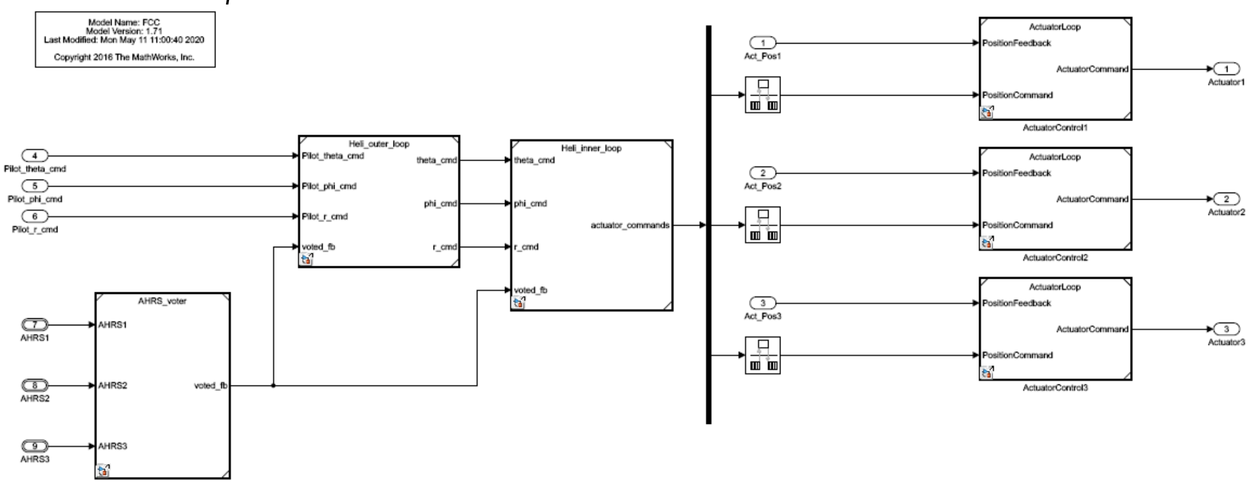
- Changing AHRS controller from a decoupled PID controller to an optimal state-space controller
- The chosen state-space controller is a Linear-Quadratic Regulator (LQR) with reference tracking and a Kalman filter observer to estimate the full state of the helicopter given the AHRS input

4.1.1 Original System

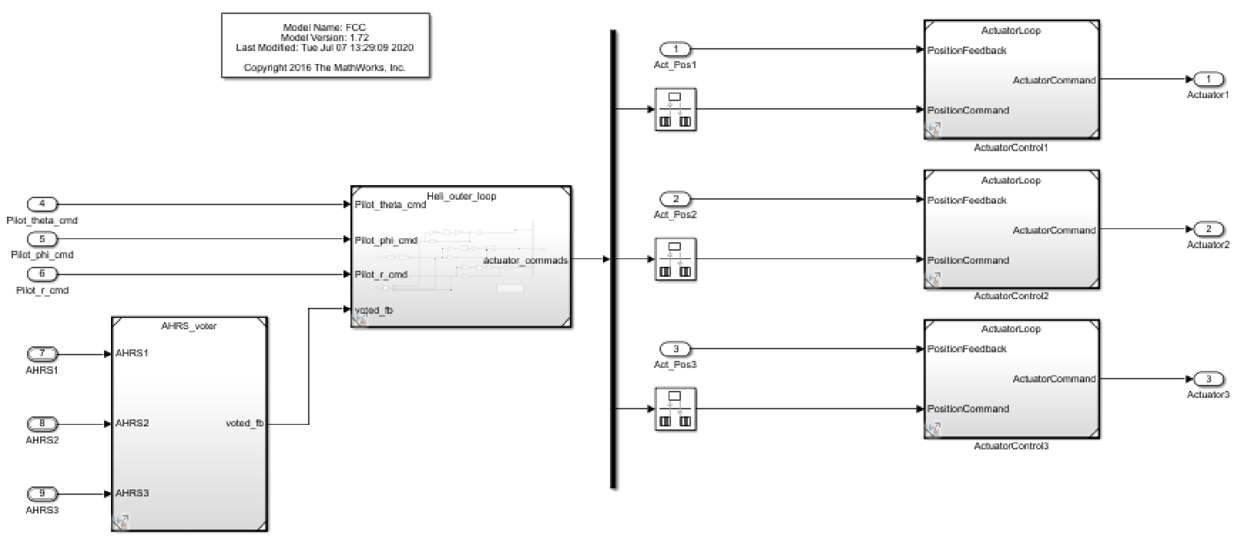
4.1.1.1 Notes

- 3 modules need to be modified, and the interface of the Heli_outer_loop module is altered
  - Remove Heli_inner_loop and its model reference in the FCC model
  - Replace controller algorithm in Heli_outer_loop module
  - Change output on the Heli_outer_loop interface
  - Rename ‘Heli_outer_loop’ to ‘Heli_loop’
    - This is because there is no longer any concept of an outer and inner loop

4.1.1.2 Before



4.1.1.3 After



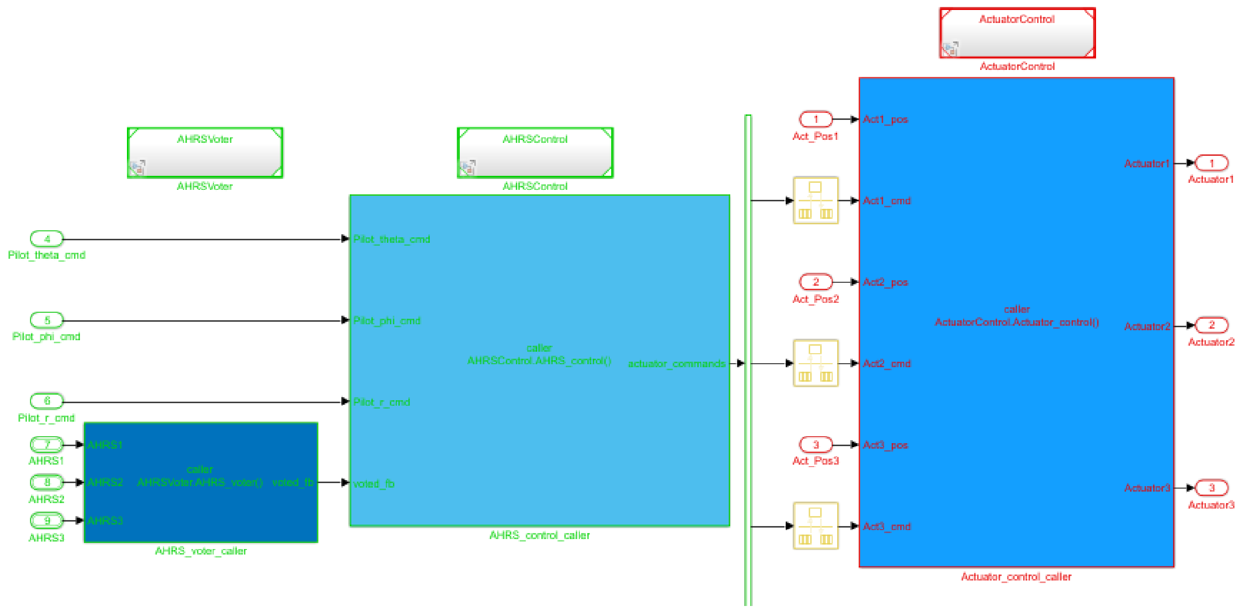
4.1.2 New System

4.1.2.1 Notes

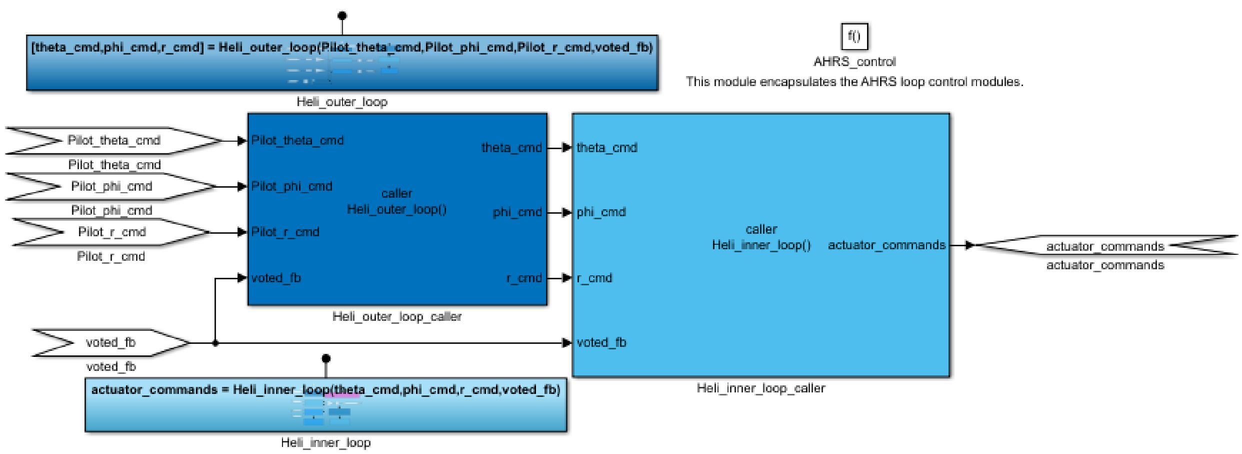
- Replace controller algorithm in AHRS Control module
- The change is contained in the AHRS Control module
    - No changes to the FCC are needed, and no interfaces are altered

4.1.2.2 Before

4.1.2.2.1 FCC_New.slx

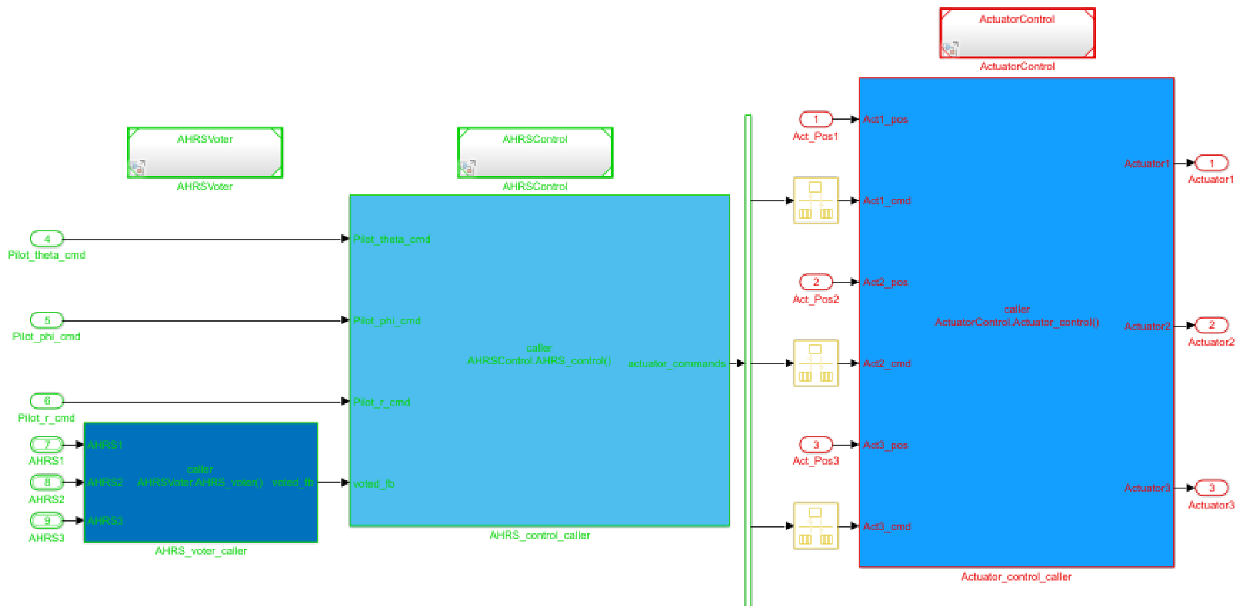


4.1.2.2.2 FCC_New.slx/AHRSControl.slx

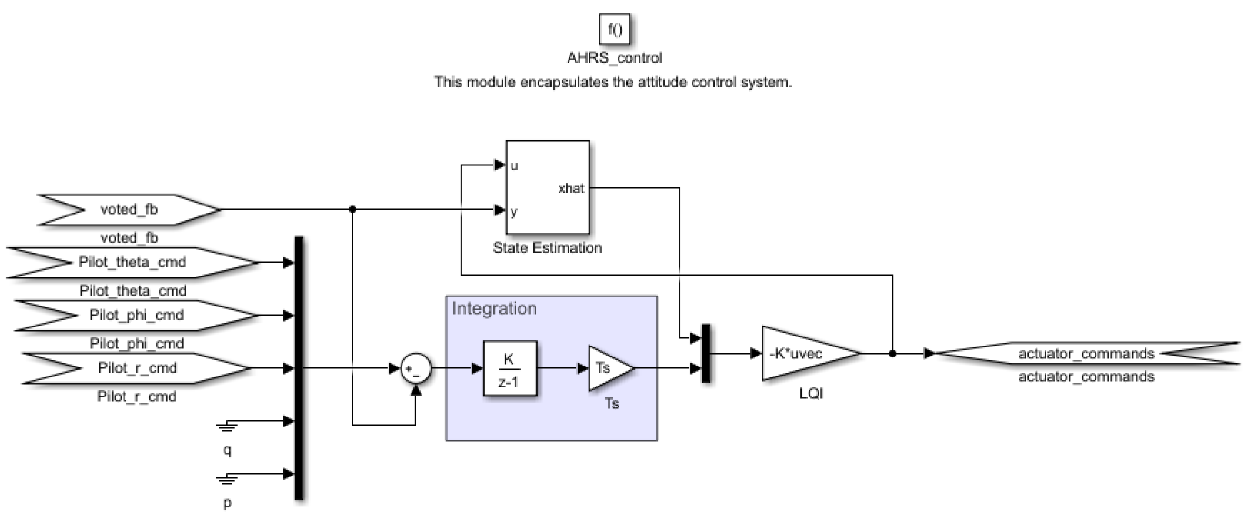


4.1.2.3 After

4.1.2.3.1 FCC_New.slx



4.1.2.3.2 FCC_New.slx/AHRSControl.slx

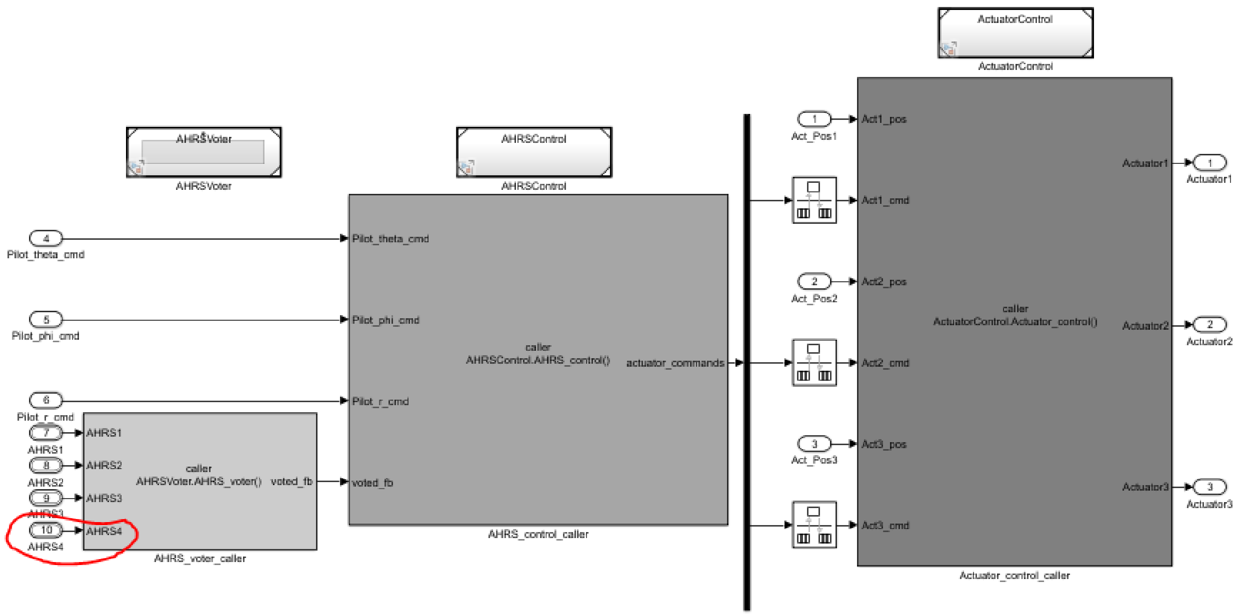


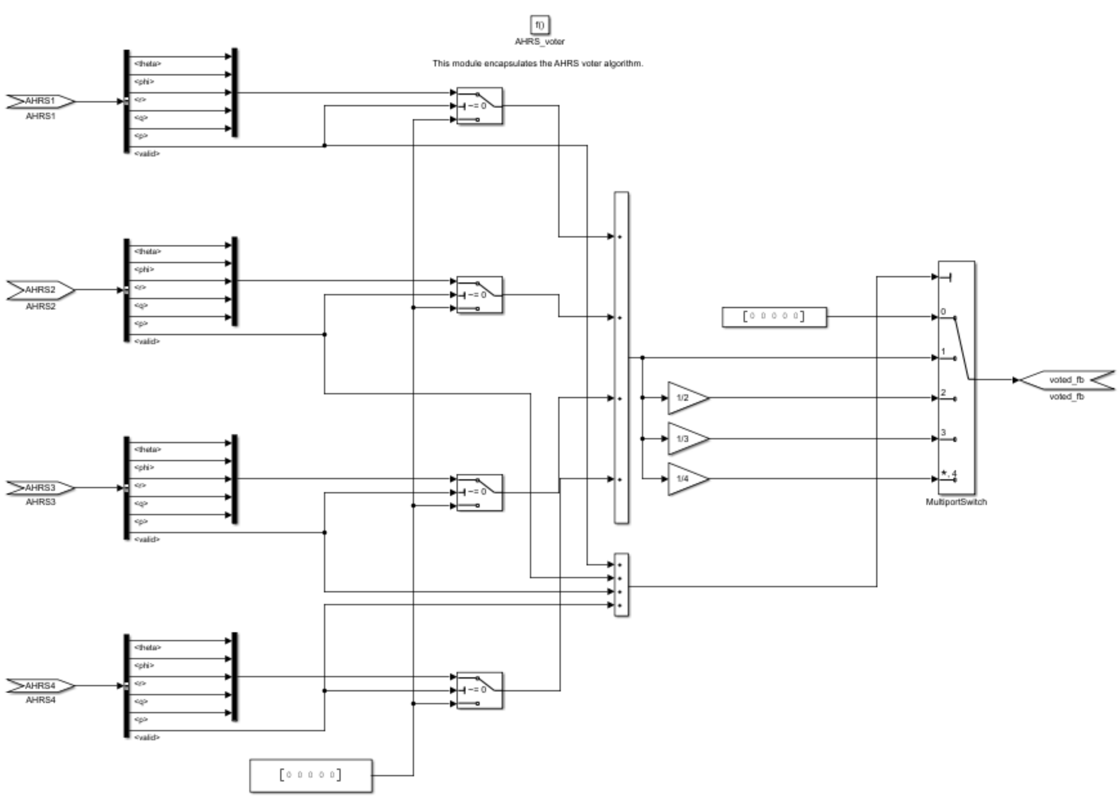
4.2 Additional AHRS Sensors

- Adding additional AHRS sensors to the system

4.2.1 Results

- Original system requires an additional input on the AHRS voter module interface
- New system exports the same functionality – but the function requires an additional input
- Both versions require modification to the internal functionality of the modules





4.3 Summary

To evaluate the effectiveness of the decomposition approach, we simulated various likely changes to the original and new systems to determine how the changes would propagate through the design structures. We first examine the impact of changing the flight control algorithm. The purpose of the controller is to track the pilot input for the pitch/roll angles, and yaw rate. In the original system, the attitude, heading, and reference control system is implemented as a decoupled PID controller. The original design incorporates the use of two model references that divide the algorithm into an outer and inner loop. The outer loop provides setpoint tracking for the pilot inputs, and the inner loop provides stability augmentation.[1]

The simulated change assumes that a PID controller is no longer required, and that the controls engineer would like to implement a state-space controller with state estimation and feedback. To implement this requirement, we chose to use a Linear-Quadratic-Gaussian (LQG) controller[2] with integral action for setpoint tracking, which consists of a linear-quadratic optimal gain to provide stability, a Kalman filter observer to provide state estimation, as well as integration for reference tracking. There is only one feedback loop in the new algorithm, and thus only one of the model references from the outer/inner loop controller was needed. We arbitrarily chose to remove the inner loop model reference and added the functionality of the optimal state-space controller to the outer loop model reference. A modification to the FCC model was needed to remove the inner loop model reference and connect the outer loop model reference to the actuator control model reference. The inner loop model was then deleted, as well as the helicopter library model that was only used by the inner loop model. Finally, a modification to the outer loop model was needed to implement the optimal controller functionality, which also required a modification to the outputs of the outer loop model. In summary, 2 models were deleted, 2 models were modified, and the interface of 1 model was changed.

The new system was decomposed in such a way that the concept of an inner and outer loop controller is hidden by the new AHRS control module. The module outputs one function that is responsible for the entirety of the control algorithm. Thus, to change the controller implementation in the new system, a modification is needed in the AHRS model to swap the old controller functionality with the new controller functionality. This change has no impact on the AHRS model's interface and requires no changes to the FCC model itself. In summary, only 1 model is modified, namely the AHRS control model.

The results of this simulated change indicate that the development effort required to change the flight control algorithm in the new decomposition of the system is less than the effort required in the original system. If each model were assigned to a developer, to implement the change in the original system, two developers would need to collaborate to modify their respective models, whereas in the new system, only one developer would be required to modify their designated model.

4.3.1 Extra Thoughts

This is assuming that development effort is measured by the number of coupling links in the system that must be modified to satisfy the change requirement. In other words, if changes are needed in many of the coupling links that make up a system, then the effort is high, which is undesired. The goal is to isolate changes to individual coupling links, such that modules can be assigned to members of a development team. This ensures that each developer can implement the required functionality of their designated module without knowledge of the underlying functionality of the other modules that encompass the system.

[1] https://www.mathworks.com/help/control/ug/tuning-control-systems-with-control-system-tuner.html

[2] https://www.mathworks.com/help/control/getstart/linear-quadratic-gaussian-lqg-design.html

The second simulated change pertains to adding an additional AHRS sensor to the system for redundancy. In both the original and new systems, this change requires adding an additional bus input to the FCC model, that must be routed to the AHRS voting model.

In the original system, an additional bus input must also be added in the AHRS voting model and a modification is required to implement the new voting logic. The new bus input on the FCC interface is routed directly into the interface of the AHRS voting model reference. The AHRS model interface changes from 3 inports and 1 outport to 4 inports and 1 outport.

In the new system, an additional argument must be added to AHRS voting function and a modification to the functionality is required. The new bus input on the FCC interface is routed to the AHRS function-call in the FCC model. The AHRS model interface still has 0 inports and outports, and 1 exported function; however, that function now requires 4 input arguments rather than 3 input arguments.

The result of this simulated change indicates that there is an indifference towards making the change in the original system versus making the change in the new system. In both the original and new systems, 2 models must be modified to implement the change. If each model were assigned to a developer, 2 developers would need to collaborate to make the required change, regardless of whether the original or new system is used.

4.3.2 Motivation Behind Version 2 Decomposition

At the conclusion of developing the first version of the new decomposition, an evaluation was performed to analyze the impact of several metrics including cyclomatic complexity, decision and execution coverage, as well as software-in-the-loop (SiL) performance. Due to the introduction of plentiful Simulink functions in the new decomposition, the system experienced a significant decrease in performance (higher execution time). This was because each Simulink function introduced the cyclomatic complexity of the system by one unit, and the cyclomatic complexity of a system directly affects the performance of that system. Specifically, the original system had a cyclomatic complexity of 41, whereas the new system had a complexity of 61, an increase of nearly 50% which resulted in a 25% increase in average SiL execution time. To limit the impact of the decomposition approach on the performance of the system, a second version of the decomposition was developed with an emphasis on minimizing the use of Simulink functions. I set a goal to limit the increase of cyclomatic complexity at 10% of the cyclomatic complexity of the system without any decomposition constructs. 10% of the cyclomatic complexity of 41 in the original system limits the decomposition to a maximum of 4 Simulink functions. In my best efforts, I was able to limit the use of Simulink functions to 6 functions total versus 18 in the first version of the decomposition. The cyclomatic complexity of the second version of the new decomposition is 46, which is only a 12% increase from the original system. (Need to compare SiL).

In the second version of the decomposition, each model still exports its respective functionality: the AHRS voter/control models export 1 function each, and the actuator control model exports 3 functions. The 6th function was a private function for the filter in the AHRS control model that was originally implemented as a link to a helicopter library model. I decided to use a Simulink function for the filter because it was functionality that was reused in the model in three different instances and thus provided the ideal opportunity for function reuse. This version of the system has the same 3 modules with the same exported functionality; however, each module has less internal branching decomposition and little to no private functionality.