



Introduction to FPP Modeling

Rob Bocchino
NASA Jet Propulsion Laboratory
October 17, 2022

Copyright © 2022 California Institute of Technology.
Government sponsorship acknowledged.



Jet Propulsion Laboratory
California Institute of Technology

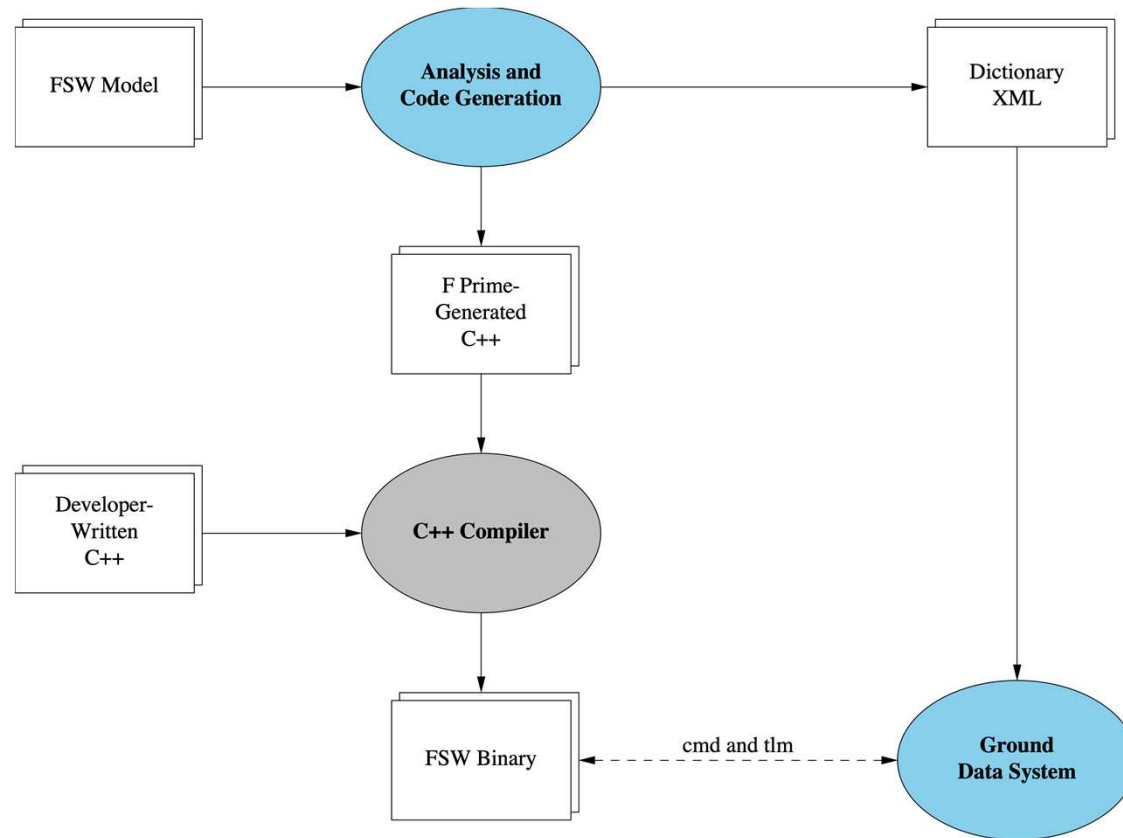


Software Modeling in F Prime

- Developers write models
 - Define components and ports
 - Specify connections in a topology
 - Define the flight-ground interface
- Tools generate code
 - C++ code for implementation and unit testing
 - XML for command and telemetry dictionaries
- Developers fill in the mission-specific details in C++



F Prime Modeling: Block Diagram



FSW Modeling: Benefits

- Clear statement of design intent
- Auto-generation of architecture diagrams
- Automatic checking of correctness properties
- Auto-generation of “boilerplate” implementation code
- Auto-generation of ground dictionaries
- Potential for integration with system modeling



FPP (F Prime Prime)

- A domain-specific modeling language for F Prime
 - Free and open source
 - Simple and easy to use
- Provides
 - A succinct and readable source representation
 - Robust error checking and reporting
 - Good integration between the model and the generated code
 - A tool for visualizing topology graphs
- Integrated with F Prime as of v3.0

<https://github.com/fprime-community/fpp>



Constants and Types

```
1 @ A constant
2 constant c = 5
3
4 @ An enum
5 enum E { X, Y }
6
7 @ An array type
8 array A = [3] U32
9
10 @ A struct type
11 struct S {
12     x: U32
13     y: string
14 } default { x = 1, y = "hello" }
```



Ports and Components

```
1 @ A port for carrying an F32 value
2 port F32Value(value: F32)
3
4 @ A component for adding F32 values
5 active component F32Adder {
6
7     @ Input: An array of two F32 values
8     async input port f32ValueIn: [2] F32Value
9
10    @ Output: A single F32 value
11    output port f32ValueOut: F32Value
12
13 }
```


Instances and Topologies

```
1 @ Command dispatcher instance
2 instance cmdDisp: Svc.CommandDispatcher base id 0x0500 \
3   queue size 20 \
4   stack size Default.stackSize \
5   priority 101
6
7 ...
8
9 @ An example topology
10 topology Example {
11
12   ...
13
14   @ Automatically insert all command connections
15   command connections instance cmdDisp
16
17   @ Command sequence connections
18   connections Sequencer {
19     cmdSeq.comCmdOut -> cmdDisp.seqCmdBuff
20     cmdDisp.seqCmdStatus -> cmdSeq.cmdResponseIn
21   }
22
23 }
```

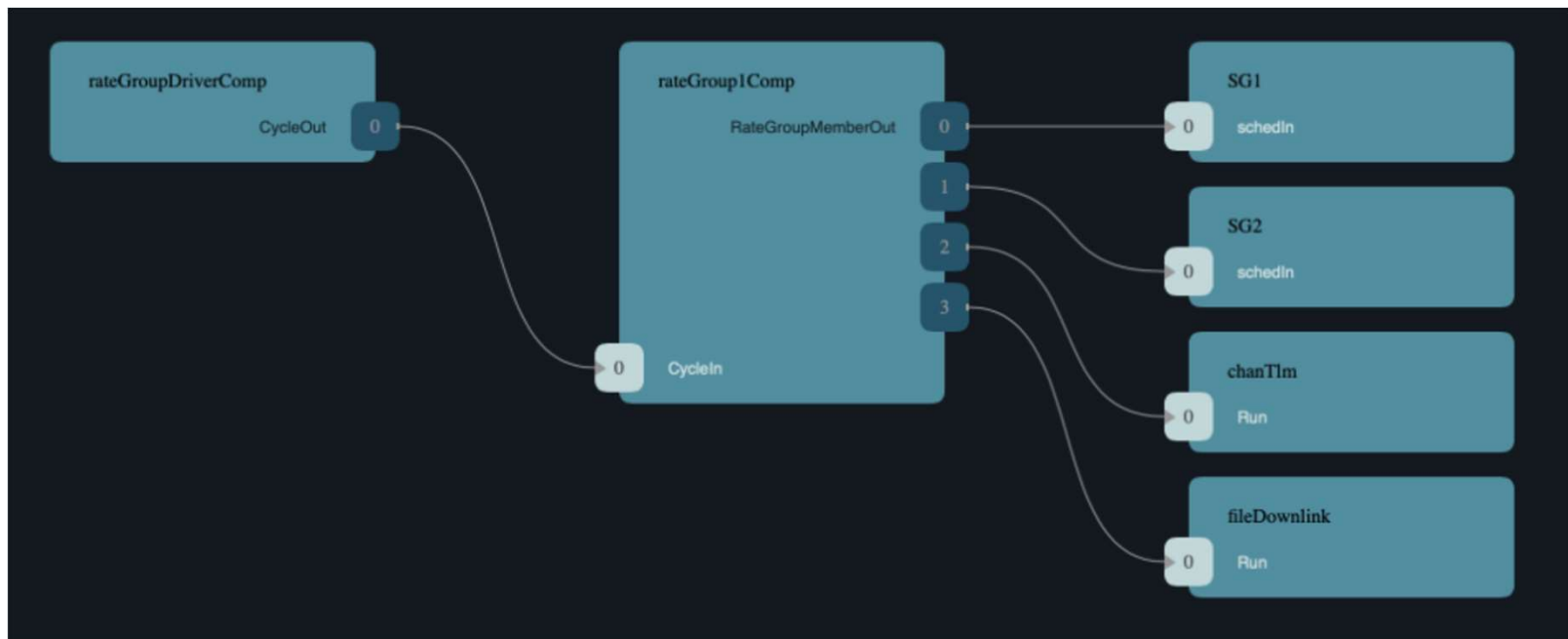



Ground Dictionaries

```
1 active component Dictionaries {
2
3   ...
4
5   @ An asynchronous command
6   async command START(a: F32, b: U32) opcode 0x10
7
8   @ An event report
9   event Event(
10    count: U32 @< The count
11  ) \
12    severity activity high \
13    id 0x10 \
14    format "The count is {}"
15
16  @ A telemetry channel
17  telemetry Channel: F64 id 0x10 update on change
18
19  @ A parameter (ground-configurable constant)
20  param Param: F64 default 2.0 id 0x10
21
22 }
```

Topology Visualization

- Visualization tool uses simple layout algorithm
- Uses named connection groups to generate subgraphs





Future Work

- Improved C++ code generation
- Improved visualization
- Advanced analysis
- Better integration with state machine modeling