Introduction to FPP Modeling

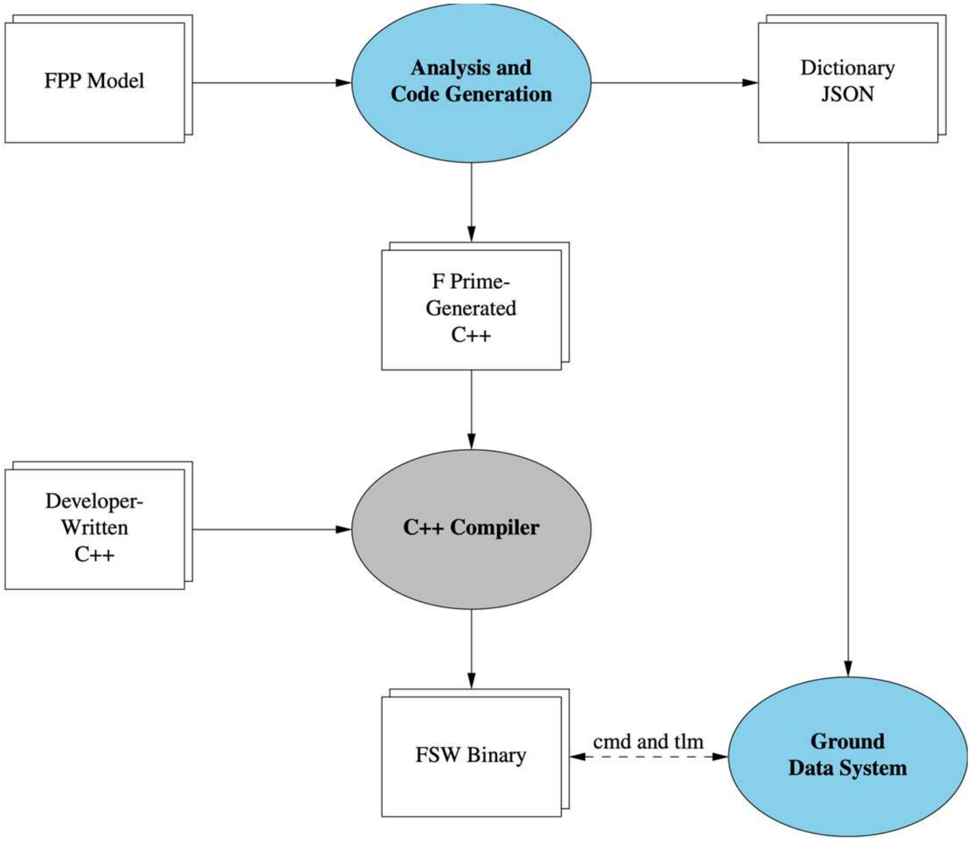
Rob Bocchino
NASA Jet Propulsion Laboratory
October 21, 2024

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
  - JSON for command and telemetry dictionaries
- Developers fill in the mission-specific details in C++

F Prime Modeling: Block Diagram



FSW Modeling: Benefits

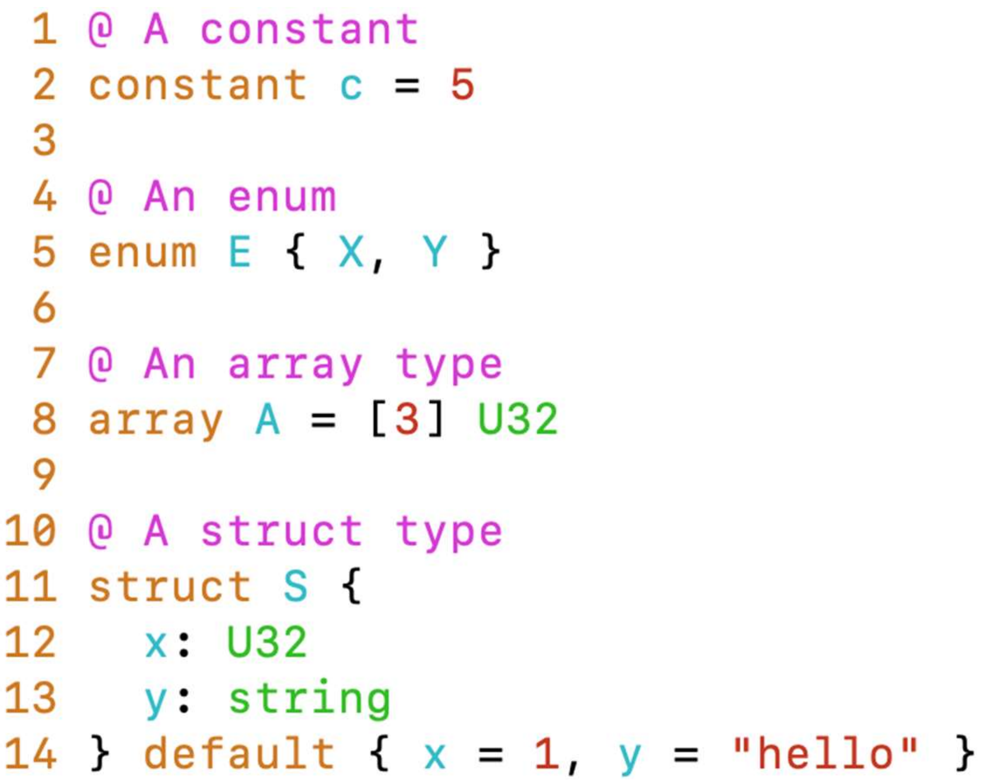
- Clear statement of design intent
- Auto-generation of architecture diagrams
- Automatic checking of correctness properties
- Auto-generation of “boilerplate” implementation code
- Auto-generation of ground dictionaries
- Potential for integration with system modeling

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 the F Prime build system

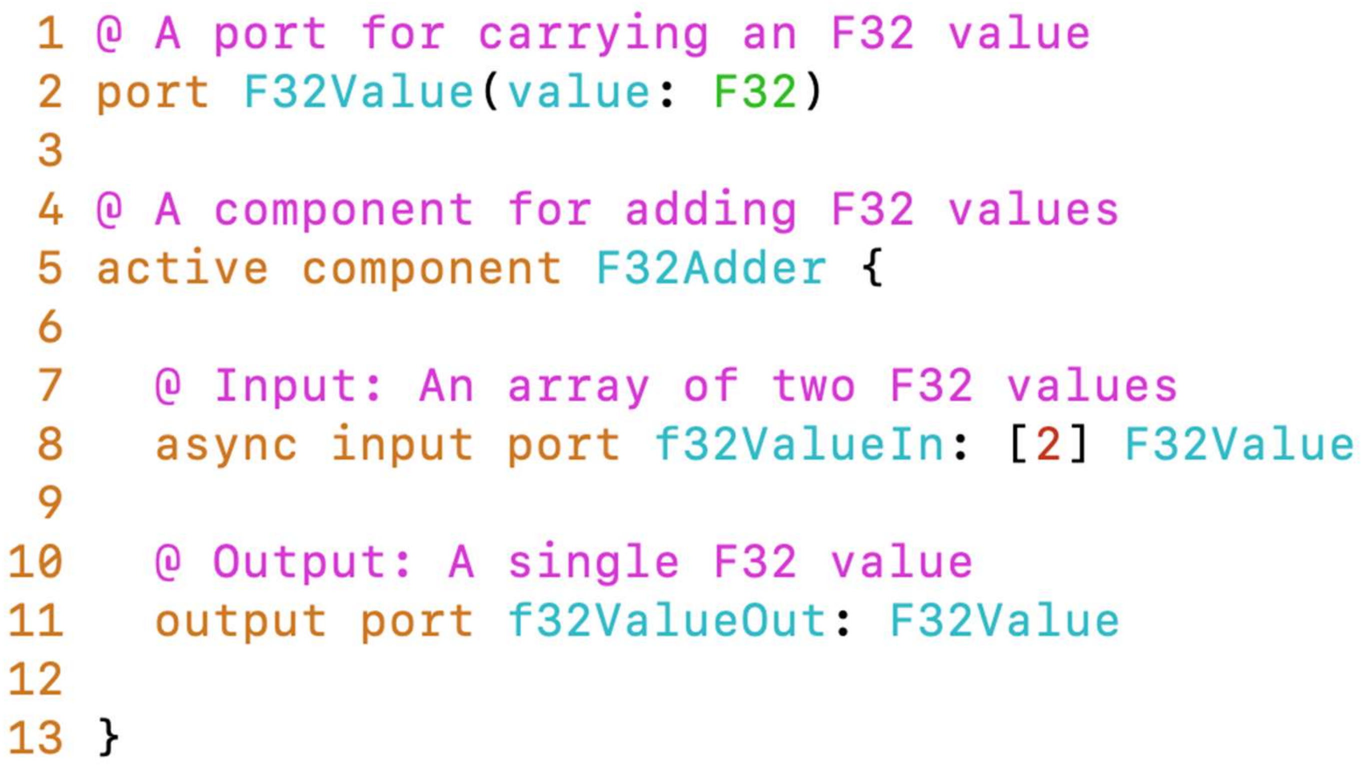
https://github.com/fprime-community/fpp

Constants and Types



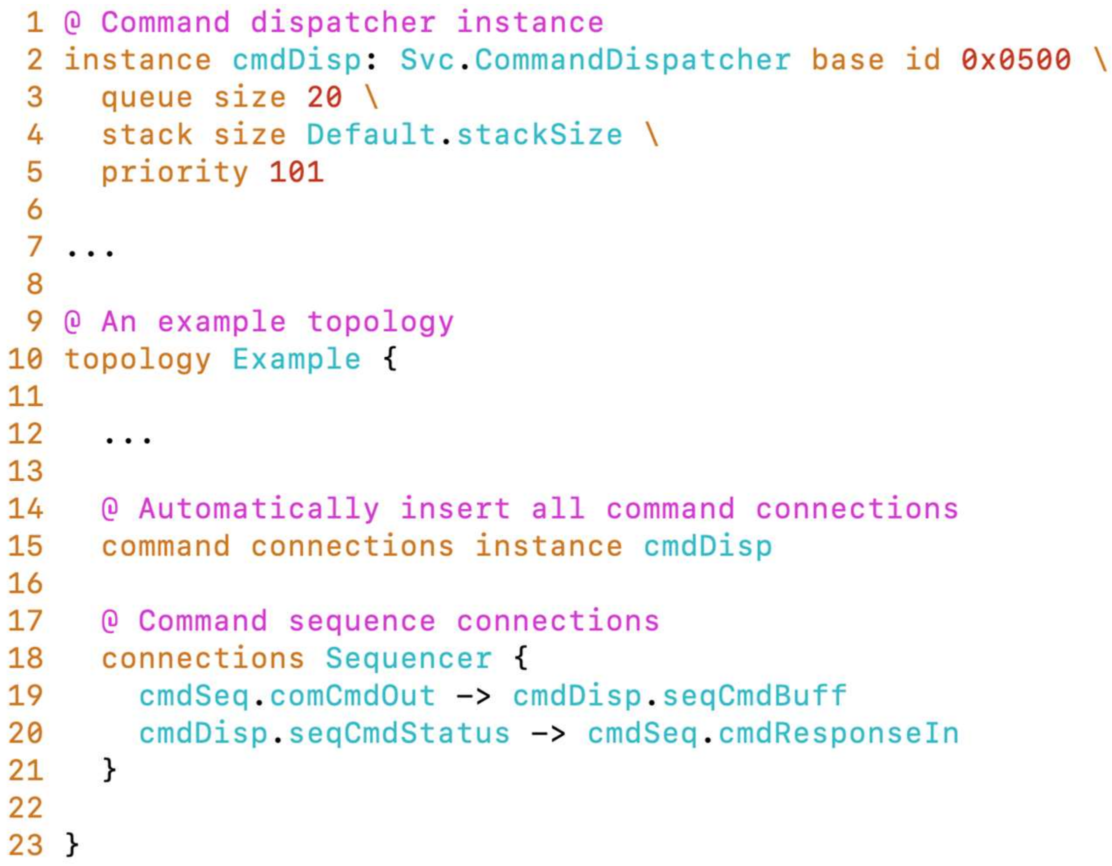
```
@ A constant
constant c = 5

@ An enum
enum E { X, Y }

@ An array type
array A = [3] U32

@ A struct type
struct S {
  x: U32
  y: string
} default { x = 1, y = "hello" }
```

Ports and Components



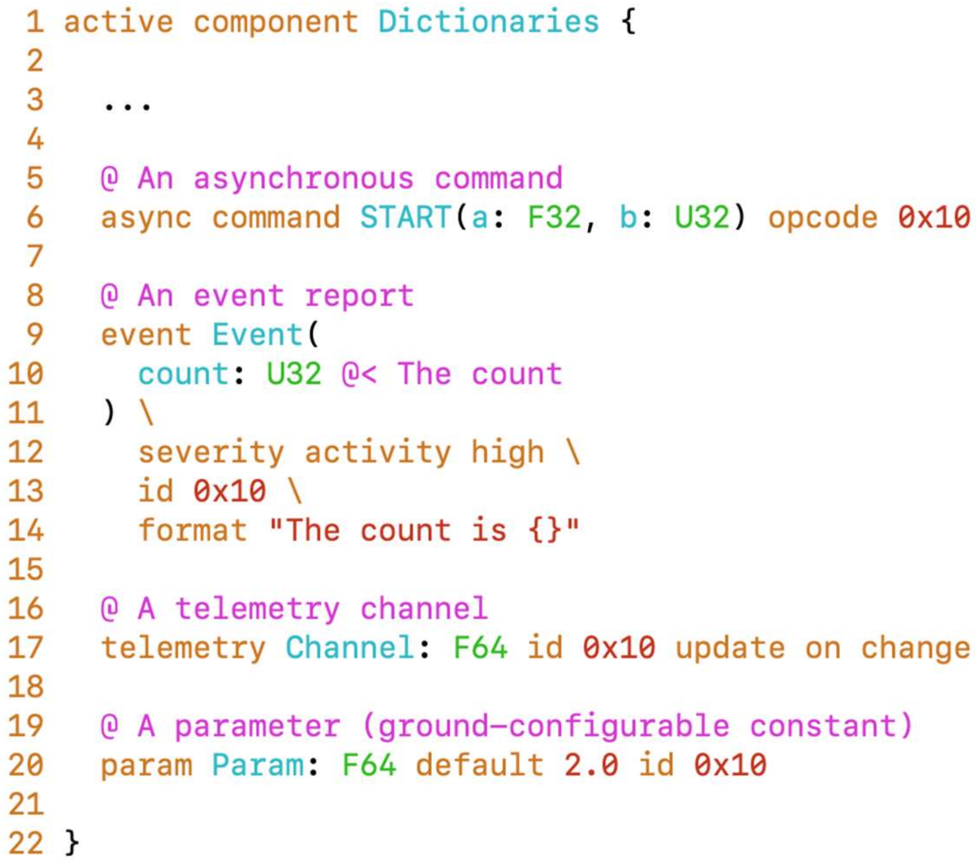
```
@ A port for carrying an F32 value
port F32Value(value: F32)

@ A component for adding F32 values
active component F32Adder {

  @ Input: An array of two F32 values
  async input port f32ValueIn: [2] F32Value

  @ Output: A single F32 value
  output port f32ValueOut: F32Value

}
```

Instances and Topologies



```
@ Command dispatcher instance
instance cmdDisp: Svc.CommandDispatcher base id 0x0500 \
  queue size 20 \
  stack size Default.stackSize \
  priority 101

...

@ An example topology
topology Example {

  ...

  @ Automatically insert all command connections
  command connections instance cmdDisp

  @ Command sequence connections
  connections Sequencer {
    cmdSeq.comCmdOut -> cmdDisp.seqCmdBuff
    cmdDisp.seqCmdStatus -> cmdSeq.cmdResponseIn
  }

}
```

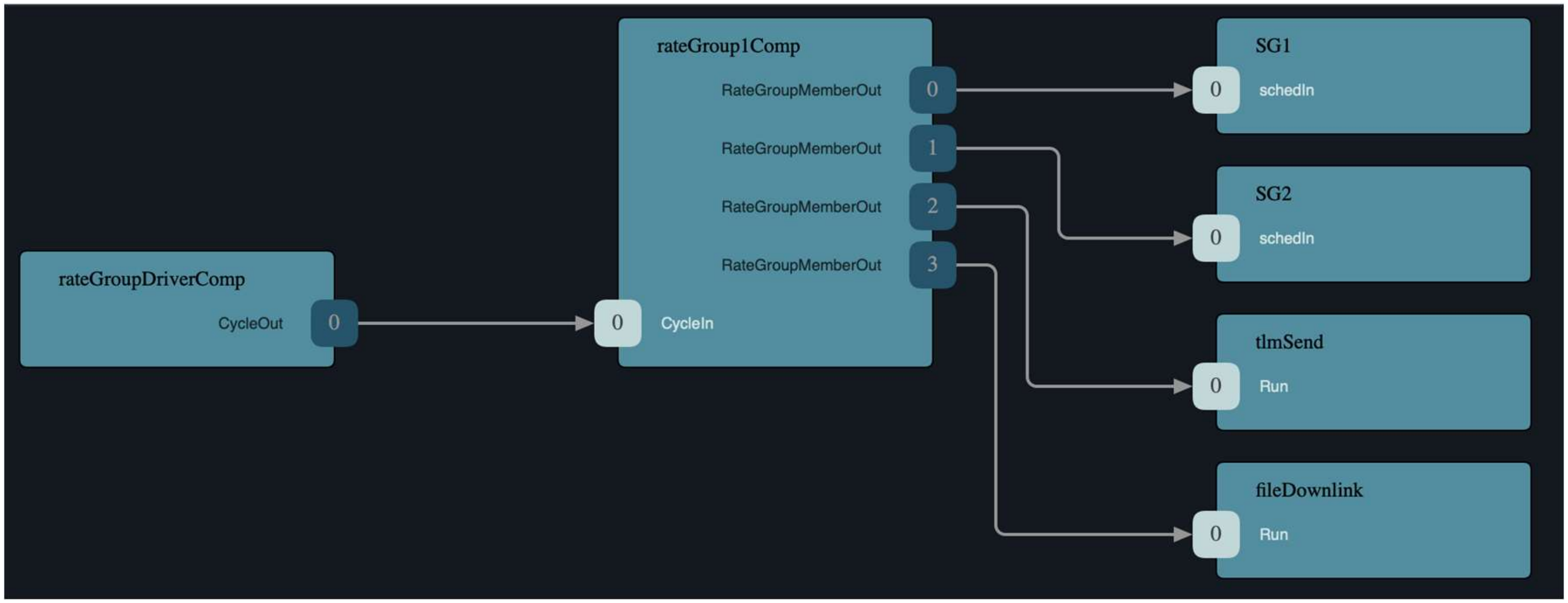
Ground Dictionaries



```
active component Dictionaries {

  ...

  @ An asynchronous command
  async command START(a: F32, b: U32) opcode 0x10

  @ An event report
  event Event(
    count: U32 @< The count
  ) \
    severity activity high \
    id 0x10 \
    format "The count is {}"

  @ A telemetry channel
  telemetry Channel: F64 id 0x10 update on change

  @ A parameter (ground-configurable constant)
  param Param: F64 default 2.0 id 0x10

}
```

Topology Visualization

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



Future Work

- Improve the visualizer
- Add new language features
  - State machine modeling
  - Type aliases and improved integration with GDS
  - Command argument validation
  - Improved topology modeling
- Improve support for system analysis
  - Advanced analysis of FSW properties, e.g., queue sizes, memory usage
  - Integration with system models