

Using Automatic Code
Generation
in the
Attitude Control Flight Software
Engineering Process

David McComas

Stephen Andrews

James O'Donnell, Jr., PhD

12/3/98



Agenda

- Background
 - Microwave Anisotropy Probe (MAP)
 - Attitude Control Subsystem (ACS)
 - MATRIXx tool set
- Software Development Process
 - Analysis and Design
 - Implementation
 - Testing
- Lessons Learned



What is the Microwave Anisotropy Probe (MAP) ?

- Spacecraft to measure properties of the cosmic background radiation over the full sky
- Measurements will determine
 - “Big Bang” parameters
 - How and when galactic structures formed
- Maintain a halo orbit about the Sun-Earth Lagrange point (L_2) 1.5 million km from Earth
- Maintain a 0.464 rpm spin rate for science data collection

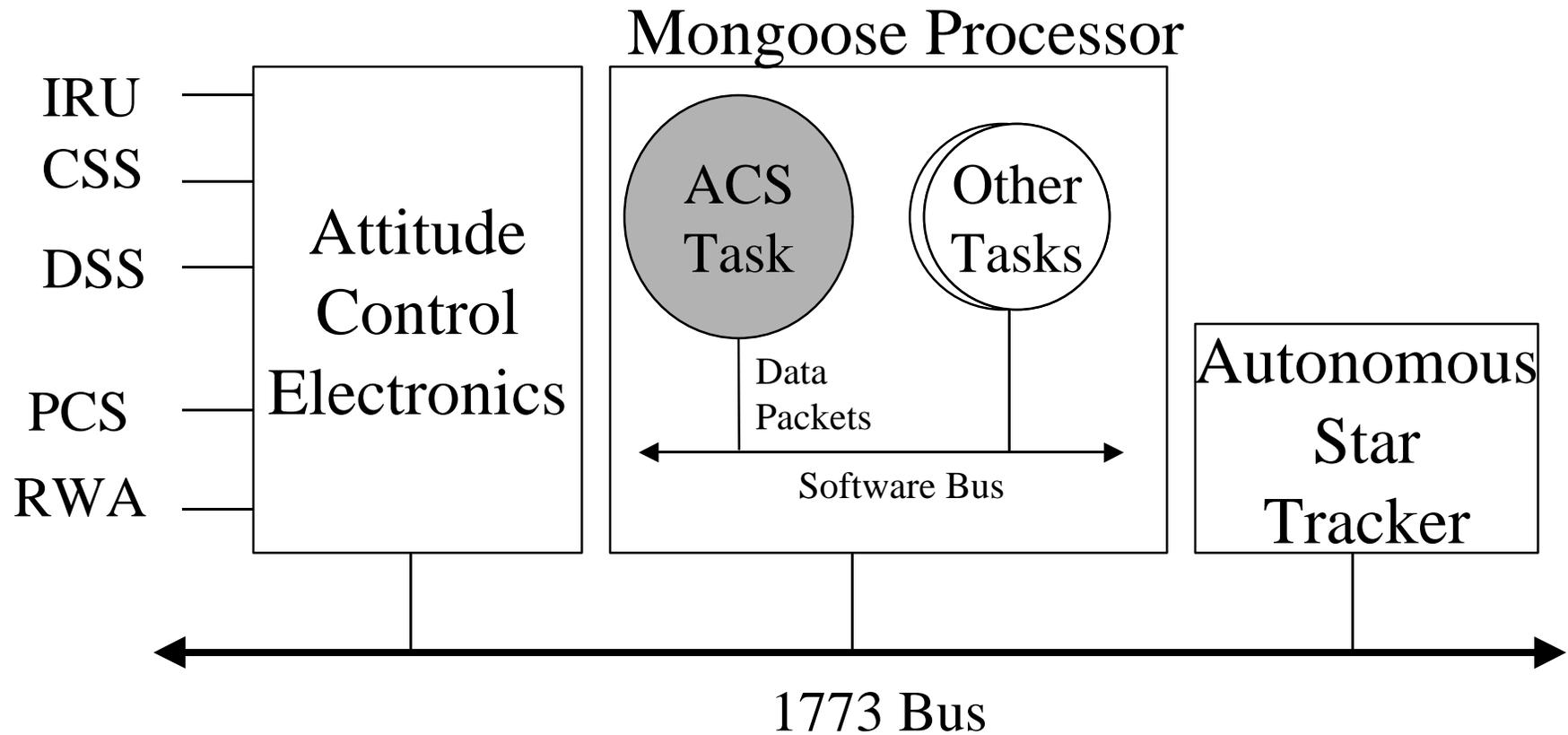


What is the MAP Attitude Control Subsystem (ACS) ?

- Onboard hardware and software responsible for
 - Attitude Determination
 - Attitude Control
 - Failure Detection and Correction
- MAP ACS manages
 - Control following separation from the launch vehicle
 - Orbit maneuvers to get to L_2 and to maintain L_2 orbit
 - Control of the 0.464 rpm scan
 - Momentum unloading



MAP ACS Flight Architecture





MAP ACS Sensor and Actuators

Inertial Reference Units (IRU)	Measure changes in MAP's angular position. Spacecraft body rates are derived from the incremental angular measurements.
Digital Sun Sensor (DSS)	Provides accurate measurements ($< 0.01^\circ$) of the sun's position within a 64 degree square field of view.
Coarse Sun Sensors (CSS)	Provide coarse measurements ($< 10^\circ$) of the sun's position. The CSSs are mounted to provide complete sky coverage.
Autonomous Star Tracker (AST)	Provides an estimated attitude derived from star measurements.
Propulsion Control System (PCS)	Provides external force and torque to the spacecraft via hydrazine-fueled thrusters.
Reaction Wheel Assembly (RWA)	Provides spacecraft attitude control via three reaction wheels.

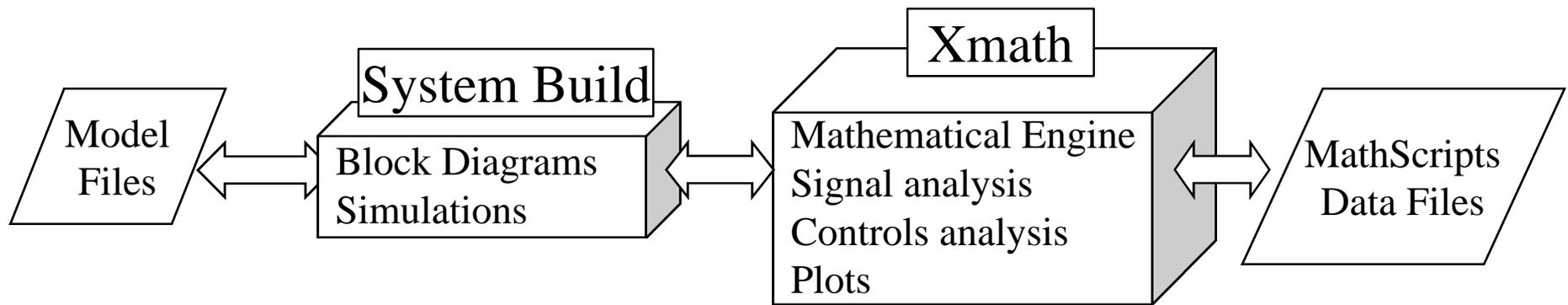


MAP ACS Operational Modes

Operational Mode	Description
Sun Acquisition (SA)	Uses IRUs, CSSs, and the RWA to acquire a sun-pointing, power and thermally-safe attitude within 20 minutes from any initial attitude.
Inertial (IN)	Uses IRUs, DSS, ST, and the RWA to acquire and hold a fixed commanded attitude.
Observing (OB)	Uses IRUs, DSS, ST, and the RWA to perform a scanning pattern. Observing is the only mode used for collecting science data.
Delta-V (DV)	Uses IRUs and the PCS to perform spacecraft maneuvers. Delta-V is used for trajectory management to get to the Sun-Earth L ₂ point approximately 1.5 million km from the Earth (away from the sun) and for L ₂ station-keeping.
Delta-H (DH)	Uses IRUs and the PCS to perform momentum unloading.



MATRIXx Tool Set



- System Parameters (Xmath %VARs) are defined in Xmath and exported to System Build. These are used for FSW tables and commands.



SystemBuild Environment

- SuperBlocks are hierarchical objects
 - Provide logical structure
 - Contain other blocks
- SuperBlock timing attributes
 - *Discrete*, Continuous, *Procedure*, Triggered
- Procedure SuperBlock types
 - *Standard*, *inline*, macro, interrupt, background, or startup

Italicized SuperBlock attributes are being used by MAP for AutoCode

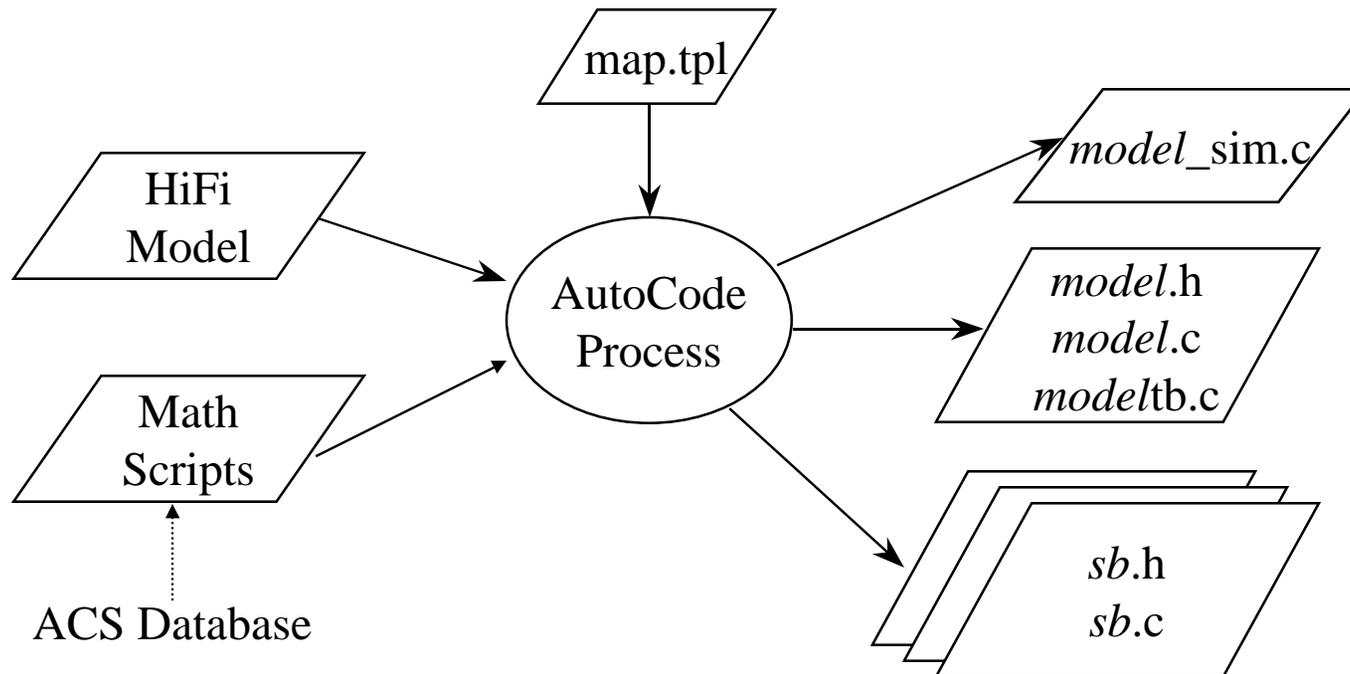


SystemBuild Environment

- Common functional blocks
 - Trig functions, algebraic functions, and dynamic systems functional blocks
- “Open” blocks include
 - Algebraic Blocks
 - Define outputs as the result of algebraic functions of the inputs and block parameters
 - BlockScript
 - Limited structural programming environment using FORTRAN-like language
 - User Code Blocks (UCB)
 - Import user written code



Code Generation Process



- *model* is the model name supplied to AutoCode. For map “achifi” is used.
- *sb* is the SuperBlock name defined in the HiFi.
- *model_sim.c* contains any non-FSW code. E.G. UCB wrappers.
- Process
 - Load HiFi model
 - Execute MathScript files to define Xmath variables
 - Run AutoCode to generate the code

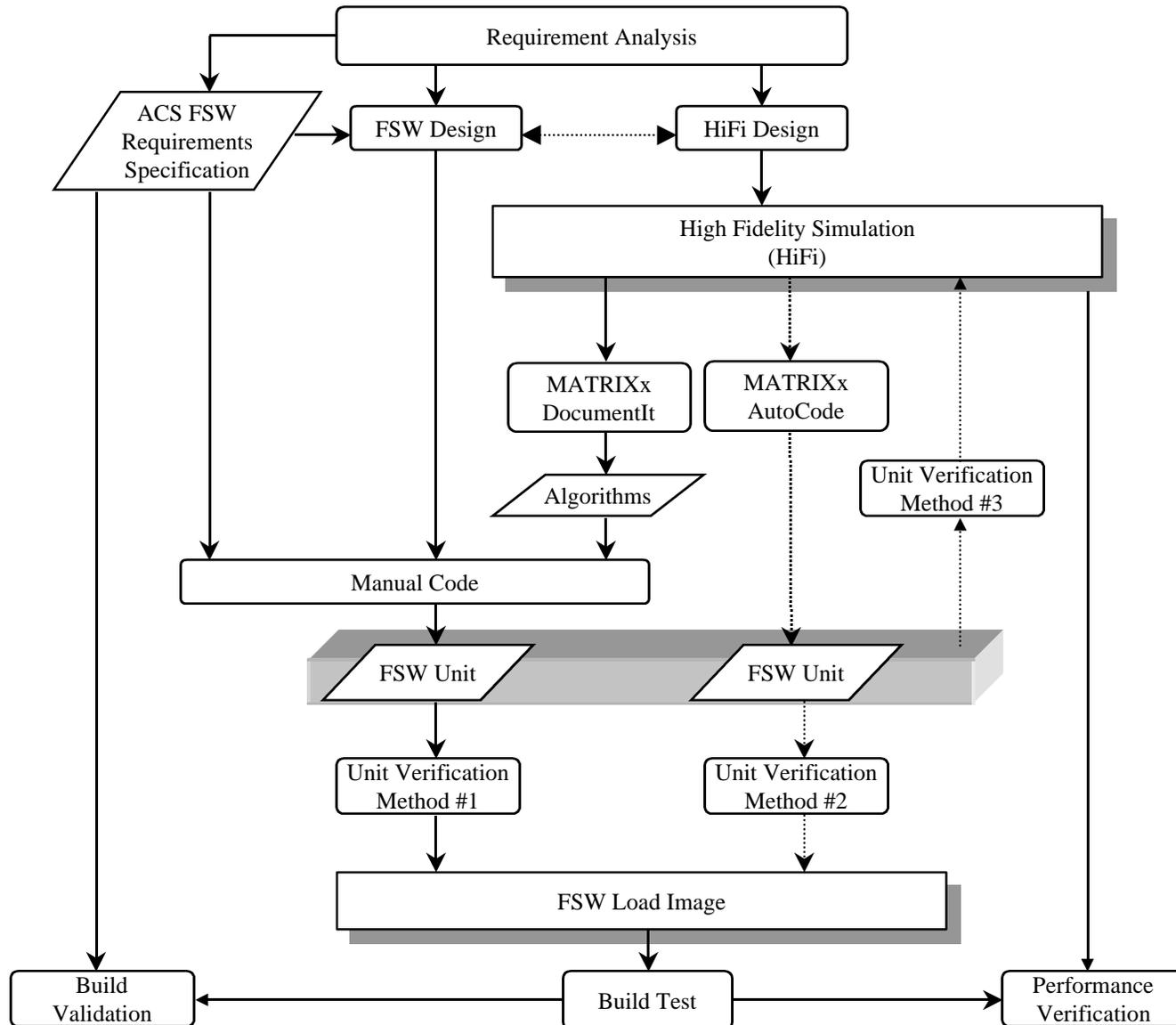


Code Generation Template (TPL) File

- Extended ISI's TPL file (renamed to map.tpl) to
 - Create separate header file and source file for each SuperBlock
 - Create *model_Init()* and *model_Dispatch()* functions to clarify interface and provide placeholder for customized code
- “Closed” automatic code generation
 - ISI supplied TPL libraries generate the code
 - `@declare_percentvars()@` generates all of the `%VAR` declarations



Software Development Process



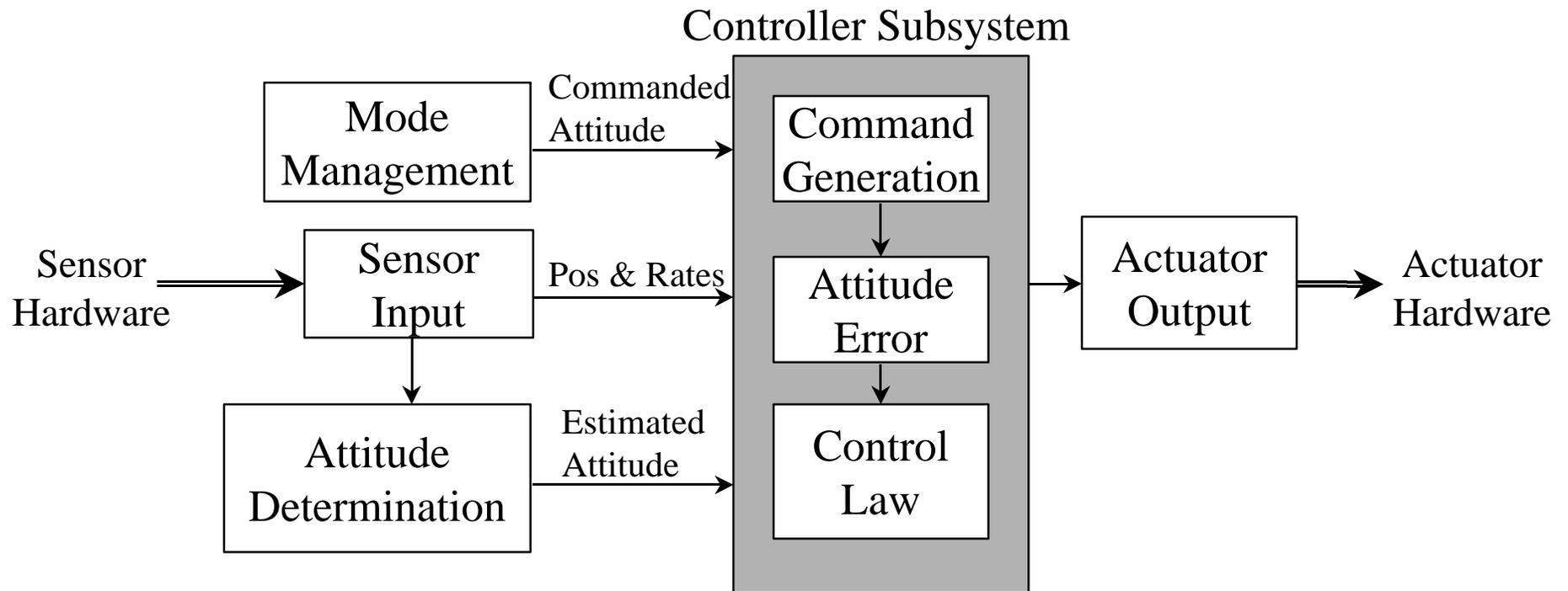


Forces Driving the Automatic Code Scope

- Minimize risks and maintain schedule
- High algorithm-to-code ratio
- Low HiFi-to-FSW architectural coupling
 - No ground command handlers
 - No software bus interfaces
 - No FDC notification, or asynchronous event message generation



Defining the Automatic Code Scope



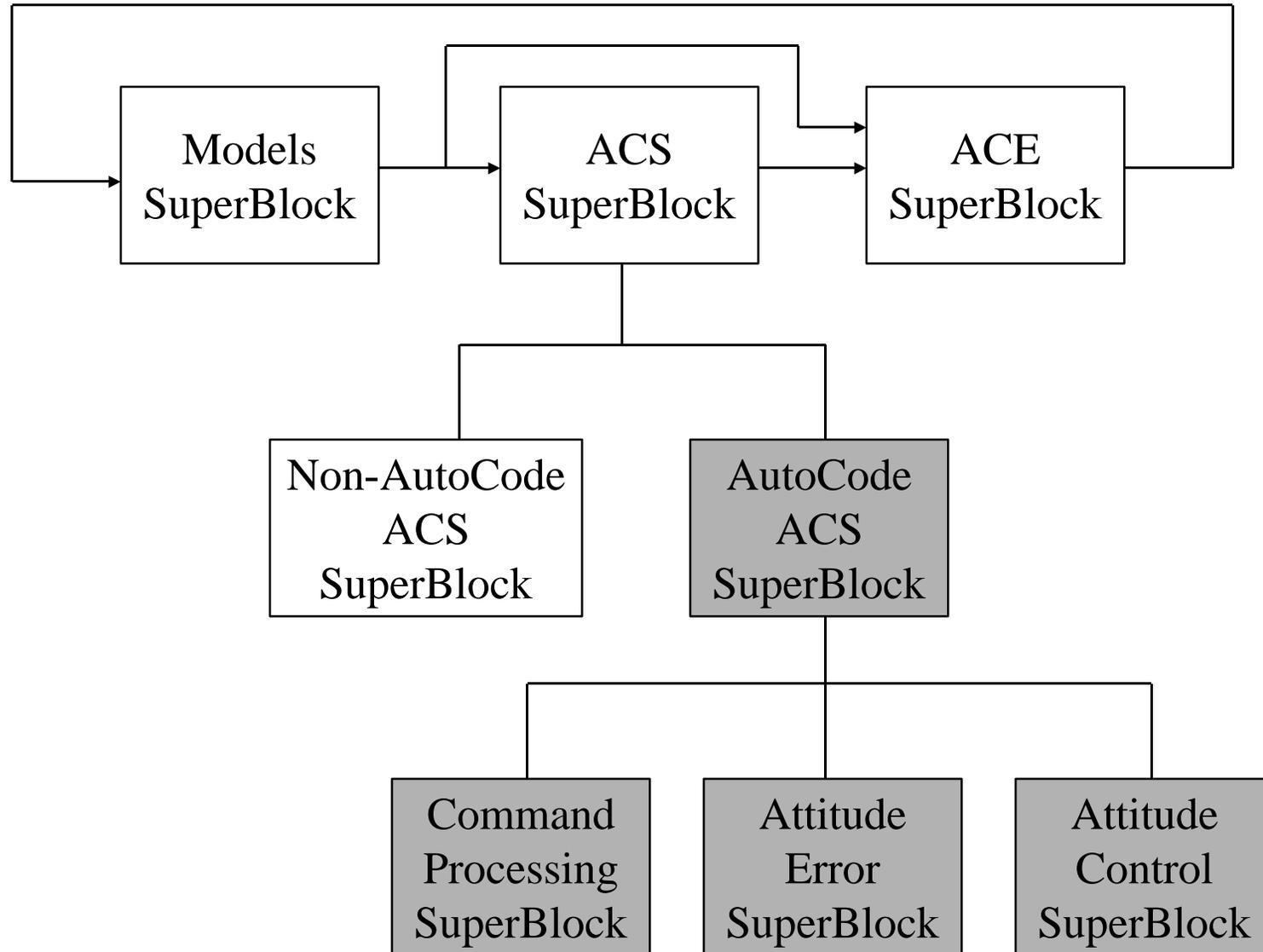


Consequences of the Design

- FSW requirements for HiFi AutoCode SuperBlock
 - FSW commanded and computed values supplied as inputs to top-level procedure SuperBlock
 - FSW telemetry and onboard computation needs define required outputs from top-level procedure SuperBlock
 - Any non-commanded ground modifiable parameter must be defined as a %VAR for inclusion into a FSW table
- Single rate system so no need to use AutoCode's scheduled-subsystem option
- Relatively simple HiFi interface to be managed by manual FSW

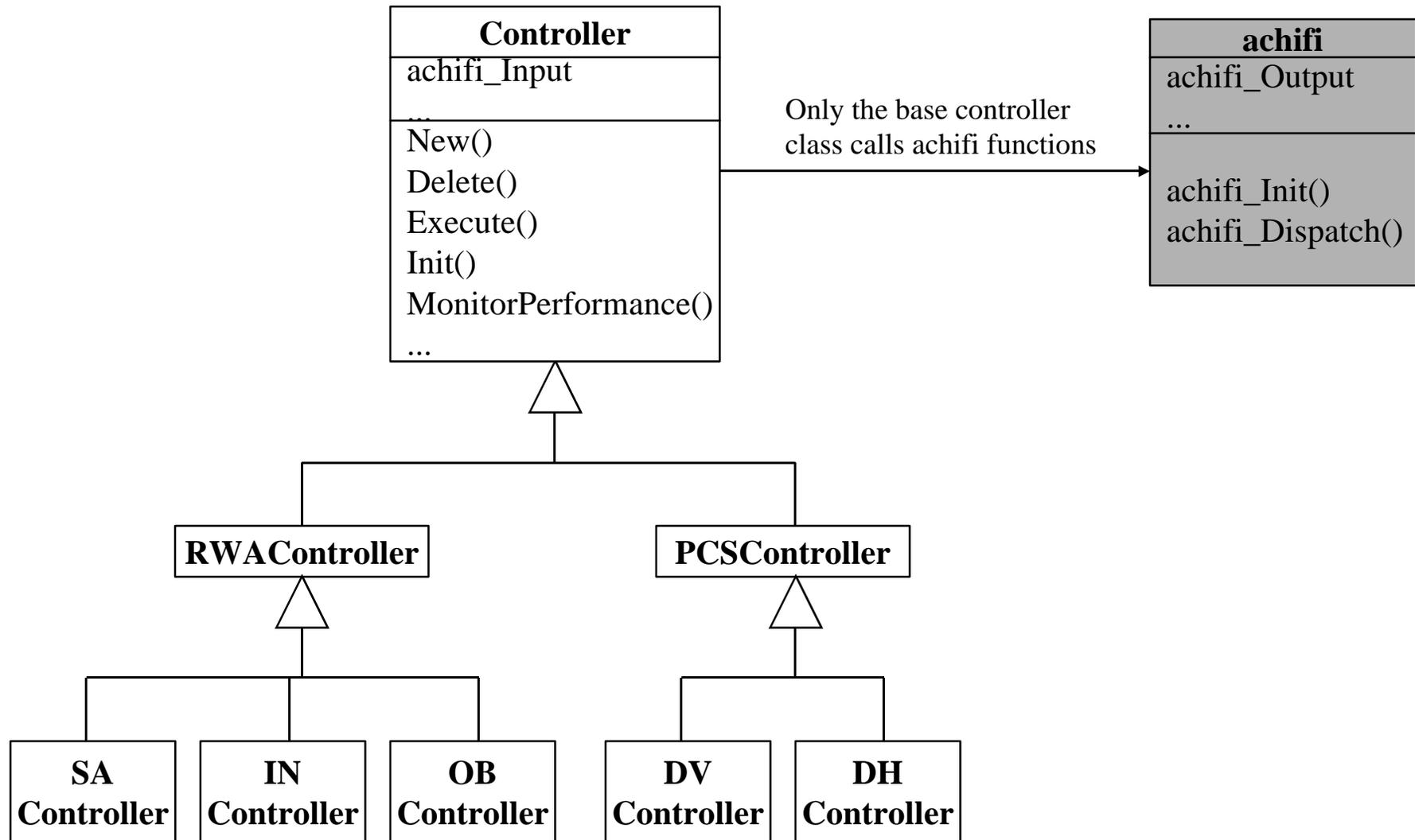


HiFi Implementation





Manual Flight Code Implementation





Manual Flight Code Evaluation

- Object-Oriented Design (OOD) implemented in C
- AutoCode dependencies encapsulated by base controller class
- achifi viewed as a single object
 - Two member functions: achifi_Init() and achifi_Dispatch()
 - achifi_Output treated as read-only output by other FSW subsystems
- OO controllers resulted in small easy to test functions



Automatic Code Evaluation

- Non-ANSI C function prototypes
 - Compiler warnings a nuisance, but no errors due to non-ANSI compliance
- Non-inline SuperBlocks result in code about twice as long as equivalent manual code. Inefficient but . . .
 - Team opted for separate files for each function (couldn't use inline attribute) for easier FSW configuration and maintenance
 - MAP has plenty of CPU and memory resources
 - Manageable file sizes and code is relatively easy to follow
- Inline comments reference SystemBuild block names/ID for traceability

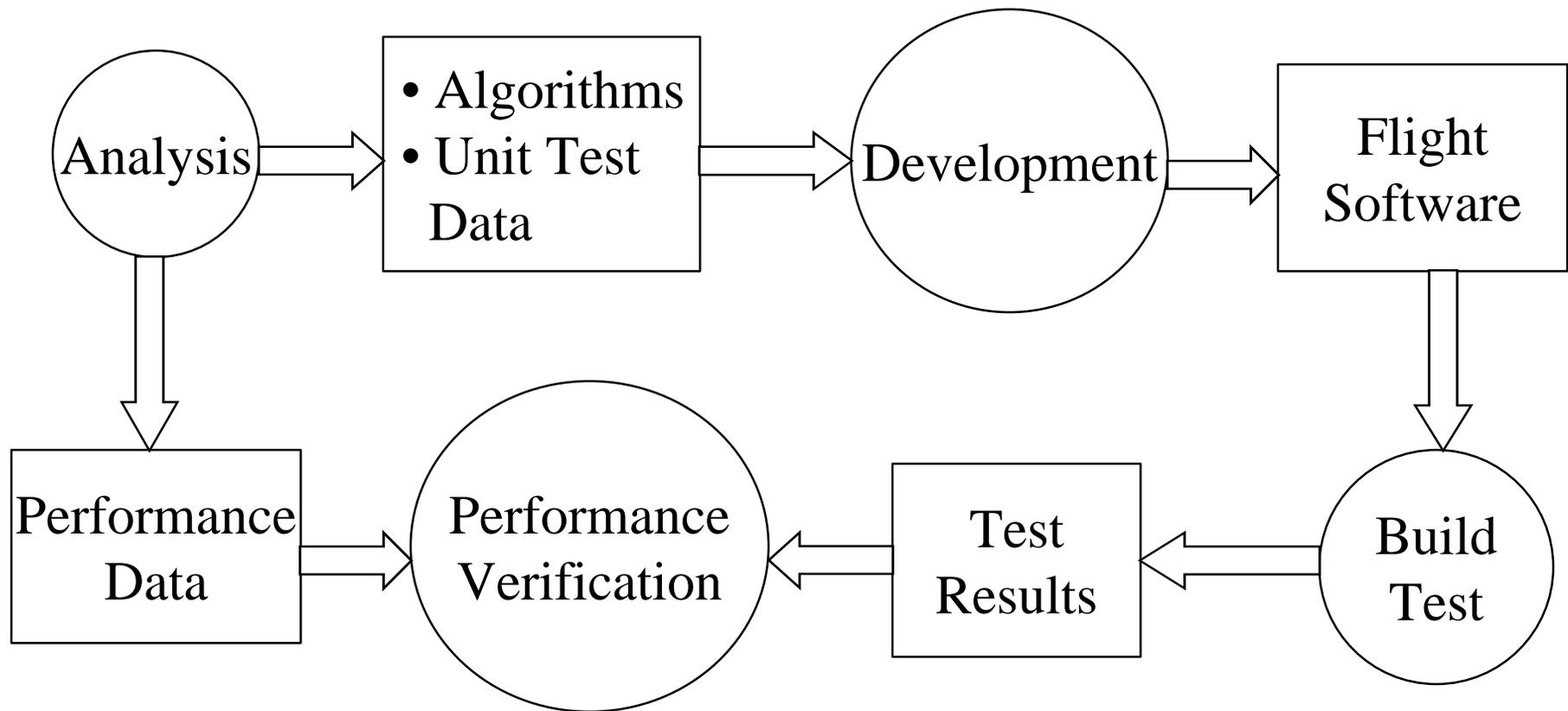


Automatic Code Evaluation

- A 1 second delta-time is hard coded
 - Manually modified generated code to use computed delta time
- Two SuperBlocks did not result in variables being used for %VARs
 - Designed workarounds in HiFi; No manual code changes
- Flight tables require contiguous data, but Xmath %VAR partitions do not translate to C structures
 - Output %VARs to a separate file and used linker scripts to ensure contiguous data storage; No manual code changes
- Team has verified automated code correctly implements the HiFi design



Lessons Learned - Process Improvements





Lessons Learned - Process Improvements

- Improved configuration management
 - Synchronized HiFi and FSW builds
 - Synchronized HiFi and FSW parameter definitions
- Improved communication
 - Analysts integrated into all phases of FSW
 - Adopted naming conventions for HiFi and FSW
 - MathScripts (next slide) documented HiFi tests
- Improved unit testing
 - Similar HiFi and FSW designs enabled better test data flow



Process Improvements

- Improved automation
 - MathScript (Xmath command files) driven HiFi simulation
 - Graphical menus for easy plot manipulation
 - Automated build test plot generation
- Consistent and efficient data analysis
- Improved performance verification
 - Generation of MathScript files from build test procedures and test data
- Many process improvements were the result of the entire tool set, not just AutoCode



Measuring Process Improvement

Lines Of Code(LOC)	Spacecraft	Man Years (MY)	LOC/MY
33,318	XTE	13.8	2414
17,525	MAP	6.1	2872

- Indicates MAP has been more productive but...
 - Metrics are limited to total developer time charged to a project; not activity specific
 - No metrics for analysts or build testers
 - MAP production rate is inflated
 - Automatic code is less efficient and 5,356 of MAP LOC (31%) are automatically generated