National Aeronautics and Space Administration

5 . . . 4 . . . 3 . . . 2 . . . 1 . . .

SPACE LAUNCH SYSTEM

December 11, 2019

SLS GNC Model-based Design Approach

Naeem Ahmad, Evan Anzalone, Young Kim, Paul Von der Porten (NASA/MSFC)
Jimmy Compton, Steven Hough, Thomas Park, John Wall (NASA/MSFC/ESSCA/Jacobs/DCI)
Christian Garcia (NASA/MSFC/ESSCA/Jacobs)

Overview

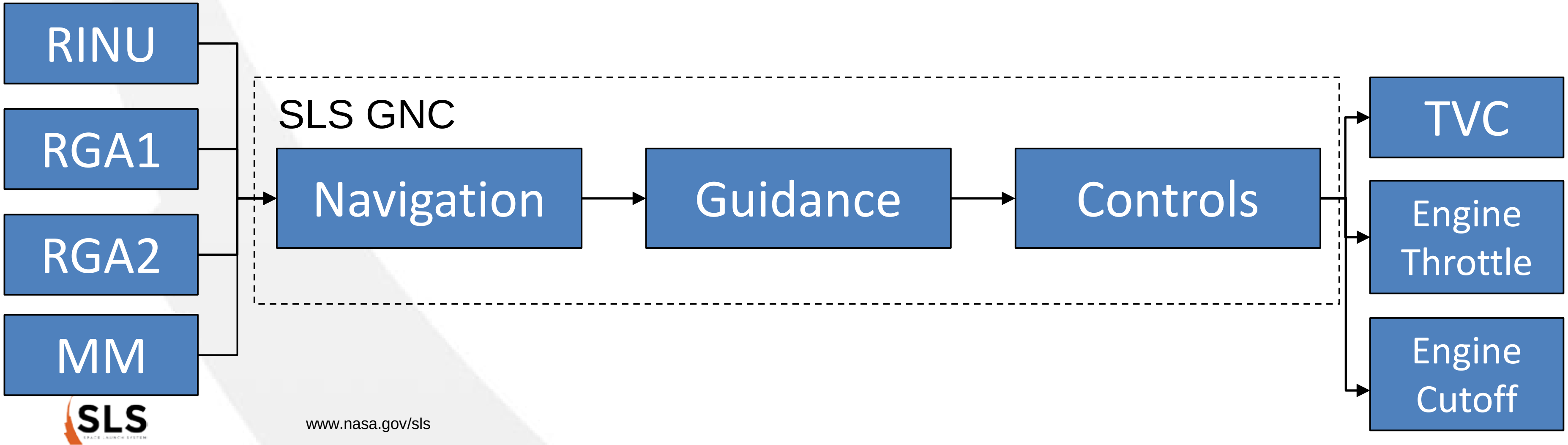
- SLS Top-level GNC Requirements
- SLS GNC Architecture
- Model-based Approach to Analysis and Design
- Simulation Architecture
- SLS GNC FSW Model Description
- Model Delivery Process
- Verification and Validation

GNC Requirements

- **High level requirement: insertion of payload into a predefined orbit with high accuracy (Apogee, Semi-Major Axis, Wedge Angle) and ensure vehicle dynamic stability**
  - Also must do this safely, respond to in-flight conditions, and provide outputs capturing vehicle health and status
- **Provide traceability from system requirements to subsystem requirements to demonstrate verification of vehicle design to meet high level requirements**

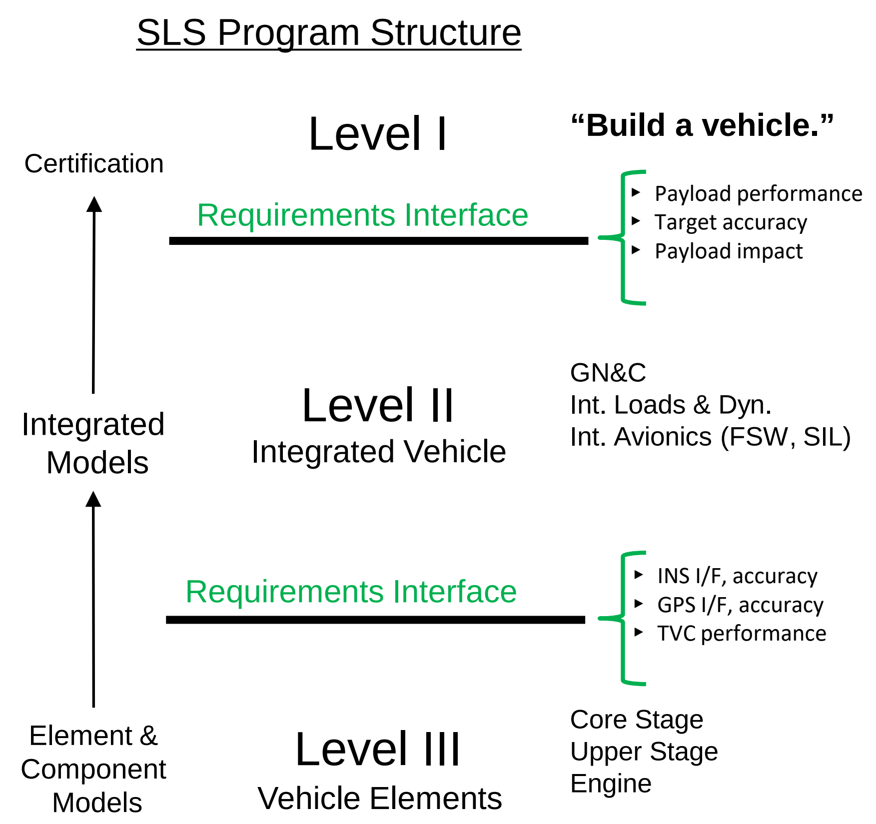
GNC Architecture

- **Inputs:**
  - Sensor observations across vehicle (navigated position, velocity, and attitude, observed angular rates and acceleration), flight parameters, mission manager inputs
- **Navigation Functions:**
  - Down-selection of redundant sensors, convert data into vehicle frame, verify sensor operational status
- **Guidance Functions:**
  - Open and closed loop algorithms to command attitude and engine state to reach orbital target
- **Controls Functions:**
  - Convert attitude and engine commands to actuator level interface, maintain vehicle stability
- **Outputs:**
  - Current vehicle state, sensor health, actuator commands, throttle commands, engine cutoff commanding
- **Analysis through Monte Carlo simulation to demonstrate integrated vehicle performance across nominal and off-nominal scenarios**



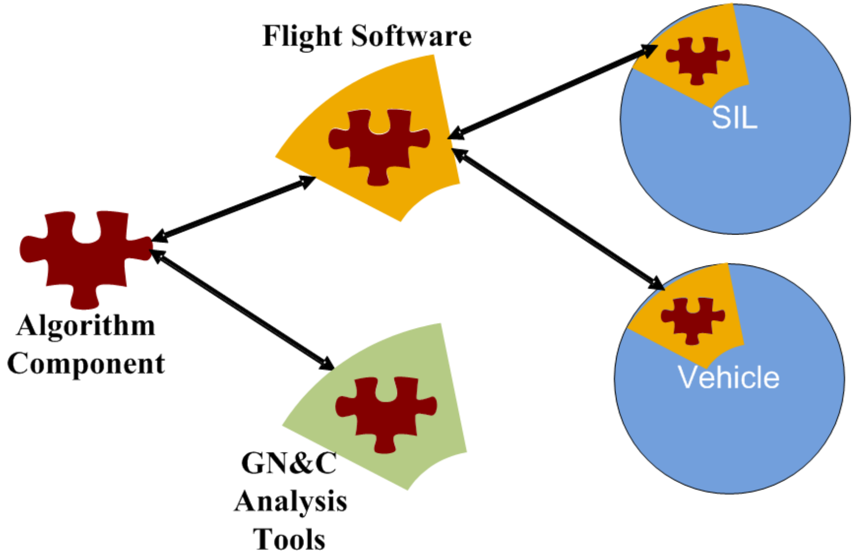
SLS SE&I Model Based Design

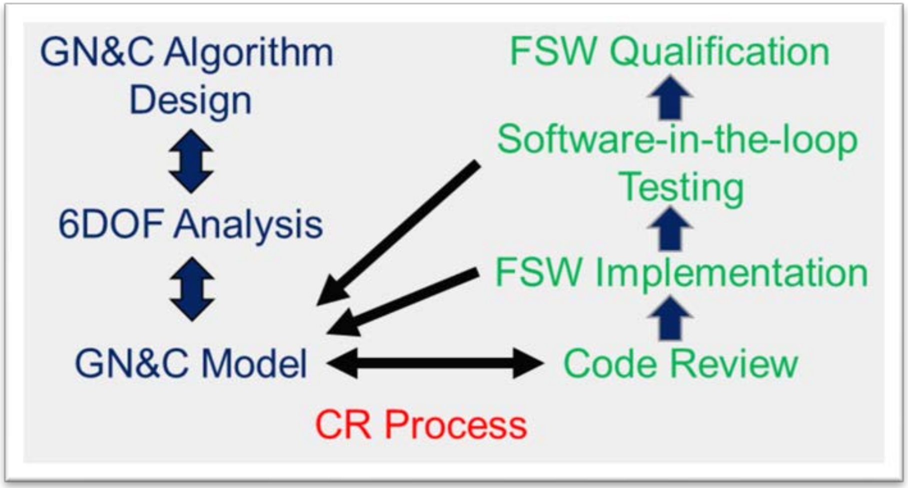
- **Reduced Program structure**
- **Emphasis on heritage hardware**
- **Relatively sparse requirements set over previous design projects**
- **Design Math Models (DMM) convey the design**
  - Controlled at program level
  - Maturity/limitations/use tightly tracked
  - Component models are verified against vendor design and validated against flight hardware (or equiv.)
  - Physics models (e.g. 6DOF sim) verified against other simulations and validated with test data
  - Model parameters of high sensitivity can be elevated to requirements
- **Example**
  - Level II DMMs: GN&C Model, MAVERIC (6DOF Sim)
  - Level III DMMs: Sensor Performance, Actuator Response, Aerodynamics Properties, Flexible Body Dynamics, etc.



GN&C Implementation of MBD Processes

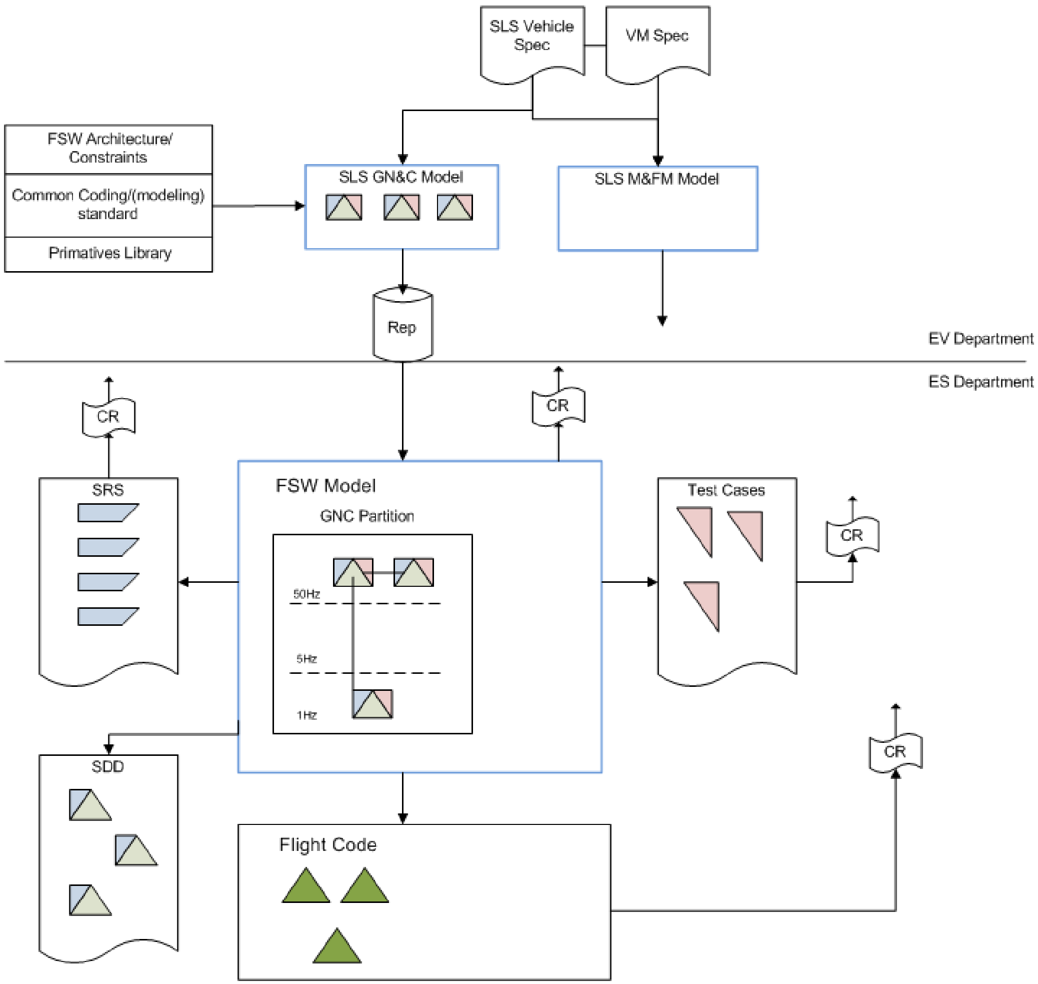
- **Heritage programs used documentation to describe GNC algorithms to FSW team**
  - Test cases included to verify functionality
- **MBD began as pilot program in 2010**
  - Working towards efficient GNC and FSW Process
- **DMM Contents**
  - Executable Algorithms
  - Parameter Definition
  - Technical Memorandum
  - Interface assumptions
  - Unit test cases
- **GNC Model**
  - Pulled directly from simulation architecture
  - Direct tie to performance results
  - Captures implementation of design
  - Includes unit test functionality and hardware testing
- **Detailed CR Process for integration with FSW and internal development**



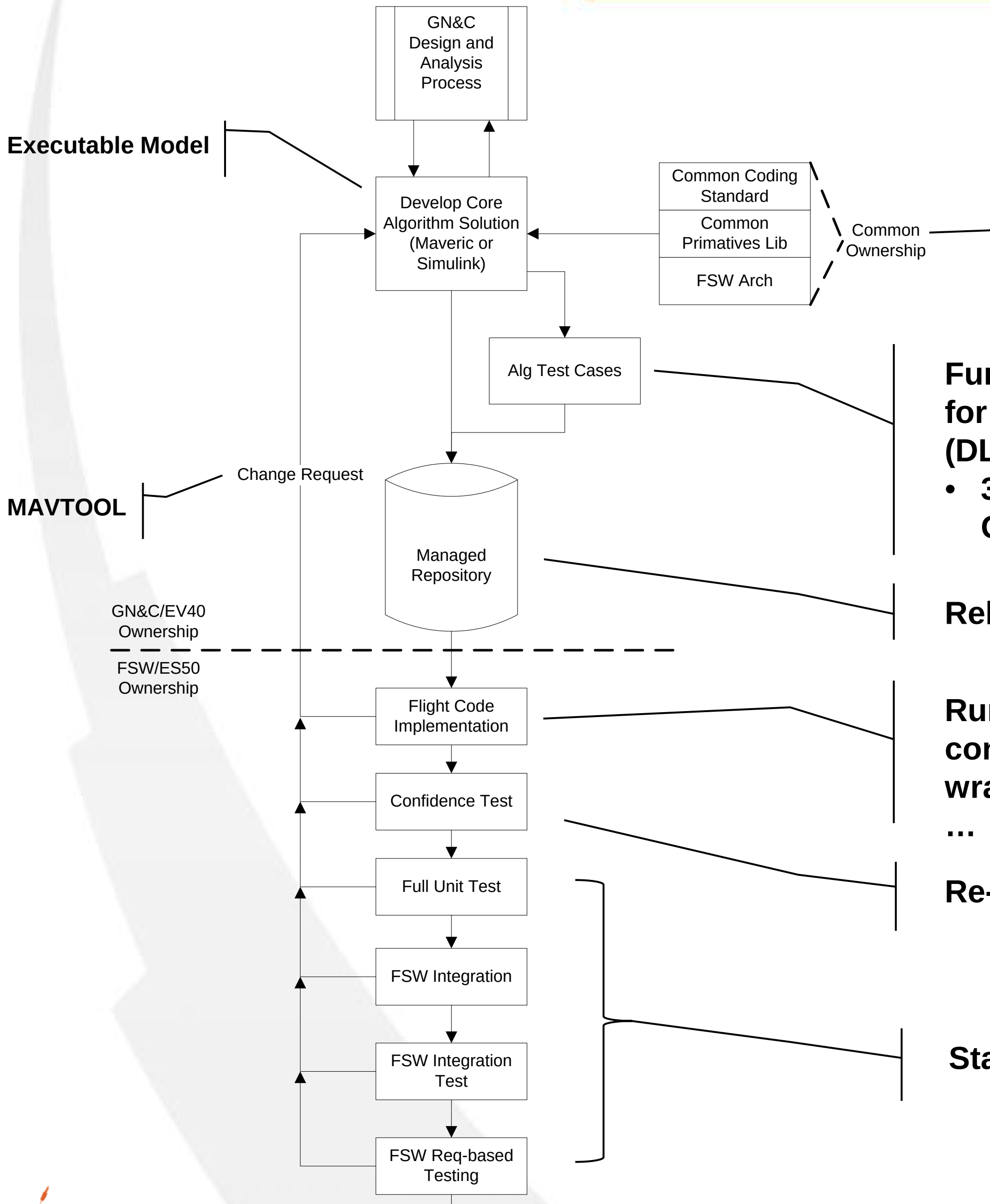


GNC/FSW High-Level Process Flow

- **Component delivery through managed repository**
- **Early introduction of FSW architecture, common primitive library, and common coding/(modeling) standard**
- **<u>Iterative, closed-loop</u> maturation to flyable implementation**
  - Modifications fed back through Change Request
- **Simulations performed to verify system performance with updates**



GNC/FSW Implementation Process Flow



Common component approach includes

- **Common primitive library in place and tested**
- **Standard GN&C/FSW architecture implemented and verified through confidence testing**
- **Coding standard adherence verified by static analysis tool, PRQA**

Functional unit test cases (FUTCs) are provided for every GN&C Design Level Requirement (DLR)

- **38 FUTCs ensure implementation of 107 GN&C DLRs**

Released through MavTool and Windchill

Run official static analysis, code inspections, confirm architecture, remove dead code, build wrappers, execution time measurements, fix bugs, …

Re-run algorithm test cases

Standard FSW approach

Verification and Validation Activities

- **At GNC Level**
  - Changes to simulation and GNC model managed as implemented into MAVERIC 6DOF simulation CR Tool
    - GNC model changes late in the design require formal CR from FSW Team
  - Unit testing between MAVERIC releases to validate integrated performance
  - Used for verifying system performance metrics
- **At GNC Model Delivery**
  - Unit test wrapped around GNC model delivery (includes inputs and outputs from MAVERIC) to validate performance on GNC model alone
  - Unit test repeated on flight-like processor platform and outputs generated
- **At FSW Integration Level**
  - Repeat of unit test with MAVERIC generated data to verify integration within FSW architecture
  - Design-Level Requirement testing and verification
- **At FSW System Level**
  - Test cases and comparisons within Software- and Hardware-in-the-Loop simulations to verify performance
  - Comparison between SIL/HIL results and MAVERIC to identify any mis-modeling and verify GNC performance on flight hardware

Summary of Approach

- **GNC Functionality**
  - Common component approach
    - Common algorithm implementation is iteratively matured to a flyable state.
  - At the G, N, and C component level, the same implementation is exercised in the GN&C development environment and the FSW target environment.
  - GN&C team delivers algorithm components in the form of executable pseudo-code.
    - Iterative, closed-loop maturation process
    - Each component meets the quality and testing standards of the FSW organization
- **Advantages from Cross-discipline Viewpoint**
  - Efficiency through elimination of non-value added and error prone documentation steps
  - Leveraging test and analysis done in previous steps
  - Early introduction of flight software architecture and implementation constraints
  - Controlled mechanism for version/release management and change requests
  - Establish a mechanism which enables rapid prototyping in the target environment
  - Establish a traceability between all algorithm and software artifacts

Conclusions

- **Implementation of MBD on SLS has significantly increased efficiency**
  - Reduces requirements burden
  - Provides explicit communication of component and integrated system design
- **Provides a mechanism for**
  - Detailed modeling and design insight
  - Identification of key vehicle sensitivities
  - Gaining additional insight through testing and validation process
  - Enforcing rigor in modeling through validation
- **DMM V&V process forces high fidelity emulation of hardware**
- **Lessons Learned:**
  - Model form and function should consider user and developer
  - GN&C Model
    - Software requirements drive the software test program
    - Approach conflicted with established FSW processes and culture
  - Component models
    - Good data requirements and supplier integration are key to enabling process
    - V&V plans should be defined early to support data requirements definition and to identify gaps which require additional testing.
    - Sensitivity analyses should be used to identify key performance drivers
    - Commonality between HWIL models and Performance/Analysis models reduces cross-validation effort in verification
  - Interfaces should be included in FSW requirements to ensure compatibility with hardware ICDs

Thank you!

Any questions?