

Evaluation of Formal Methods Tools Applied to a 6U CubeSat Attitude Control System*

Kerianne H. Gross,¹ Jonathan A. Hoffman,² Matthew Clark³
Air Force Research Laboratory, Wright-Patterson Air Force Base, OH, 45433

Eric D. Swenson⁴, Richard G. Cobb⁵
Air Force Institute of Technology, Wright-Patterson Air Force Base, OH, 45433

Michael W. Whalen⁶
University of Minnesota, Minneapolis, MN, 55455

and

Lucas Wagner⁷
Rockwell Collins, Cedar Rapids, IA 52402

Exhaustive test of complex and autonomous systems is intractable and cost prohibitive; however, incorporating formal methods analysis throughout the system design process provides a means to identify faults as they are introduced and drastically reduce the overall system development cost. Software errors on fielded spacecraft have resulted in catastrophic faults that could have been prevented had formal methods been applied to the system design. In this research, formal methods, such as model checking and limited theorem proving, are applied to the requirements, architecture, and model development phases of the design process of a reaction wheel attitude control system for a 6U CubeSat. The results show that while feasible, several gaps exist in the capability of formal methods analysis tools. The tools are capable of expressing and analyzing some of the properties of the system, but more work is needed to properly address inherent nonlinearities in complex systems.

Nomenclature

\mathbf{h}_x	Angular momentum of subsystem x	\mathbf{q}	Spacecraft quaternion vector
$\dot{\mathbf{h}}_x$	Time rate of change of angular momentum of subsystem x	$\dot{\mathbf{q}}$	Time rate of change in the quaternion vector
D	Moment of inertia of a reaction wheel	$\boldsymbol{\Omega}$	Angular velocity of the spacecraft
ψ_i	Angular velocity of reaction wheel i	$\dot{\boldsymbol{\omega}}$	Angular acceleration of the spacecraft
$\boldsymbol{\psi}$	Reaction wheel angular velocity	\mathbf{q}_c	Commanded quaternion vector
\mathbf{I}	Mass moment of inertia of the spacecraft	\mathbf{q}_p	Present spacecraft orientation quaternion
\mathbf{H}_{tot}	Total spacecraft angular momentum	\mathbf{q}_e	Quaternion error
α_i	Angle between array reaction wheel i momentum vector and x-axis	β	Angle between reaction wheel momentum vectors and z-axis
K_p	Proportional Gain	K_D	Derivative Gain

*Distribution A: Cleared for Public Release 88ABW-2015-3778

¹ Aerospace Engineer, Autonomous Control Branch, 2210 8th Street.

² Computer Engineer, Autonomous Control Branch, 2210 8th Street.

³ Electrical Engineer, Autonomous Control Branch, 2210 8th Street.

⁴ Associate Professor, Department of Aeronautics and Astronautics, 2950 Hobson Way, AIAA Senior Member.

⁵ Associate Professor, Department of Aeronautics and Astronautics, 2950 Hobson Way, AIAA Associate Fellow.

⁶ Director, University of Minnesota Software Engineering Center, 200 Union St SE, Ste 6-202.

⁷ Senior Research Scientist, Trusted Systems, 400 Collins Road NE.

ψ_{com}	Commanded time rate of change in angular velocity for each of the four reaction wheels	\mathbf{S}	Orientation of the angular momentum of each reaction wheel with respect to the principal axes of the spacecraft
\mathbf{M}_{ext}	External moments on the spacecraft	\mathbf{S}_{pinv}	Pseudoinverse of the \mathbf{S} matrix
\mathbf{x}	Spacecraft state vector	$\boldsymbol{\omega}^\times$	Skew-symmetric matrix of angular rates
$\dot{\mathbf{x}}$	Time rate of change of the state vector	\mathbf{U}	Identity Matrix

I. Introduction

Formal methods tools allow designers to express desired system properties in a limited set of precise statements and then exhaustively search all potential states that the system could take until they return a proof that the properties are true or an example of how the system can violate the properties. Two of the formal methods tools used in this research are being explored in the DARPA High-Assurance Cyber Military Systems (HACMS) Program.¹ To date, the combination of formal methods tools proposed in this research have been used successfully to design a coupled tanks control system governed by limited modes, Boolean states, and dynamics expressed in simple math;² however, an example with more complex dynamics and governing equations is desired to evaluate the performance and limitations of the selected formal methods tools.

The 6U CubeSat attitude control system is used as a challenge problem to assess the maturity and capability of a set of formal methods analysis tools. An early design verification and validation (V&V) approach is presented that applies formal methods analysis to the requirements, architecture, and model design phases. First, the Specification and Analysis of Requirements (SpeAR) framework is used to apply model checking and limited theorem proving analysis to the system requirements. Second, the Assume Guarantee Reasoning Environment (AGREE) annex is used to complete assume-guarantee contract analysis to the architecture of the system written in the Architecture Analysis & Design Language (AADL). Finally, Simulink Design Verifier is used to analyze the Simulink model of the system. The complex control algorithms and equations of motion that describe the 6U CubeSat control system challenge the limits of these tools and provide an example to evaluate the feasibility of the approach and identify gaps in tool capabilities.

This paper first presents the Air Force's motivation for the research. Second, previous space software faults that could have been prevented by formal methods are discussed. Third, the governing equations and control laws of the CubeSat attitude control system are introduced. Next the formal methods tools are described. Finally, the capabilities and limitations of the tools for the example system are identified, and recommendations are made for further development of these formal methods toolsets.

II. Motivation

In 2010, the Air Force Chief Scientist highlighted the need for alternative verification and validation methods for autonomous systems in his Technology Horizons Report, stating "It is possible to develop systems having high levels of autonomy, but it is the lack of suitable [V&V] methods that prevents all but relatively low levels of autonomy from being certified for use."³

While certification of autonomous systems encompasses both hardware and software V&V, the software that defines the autonomous decision making capability arguably presents the largest challenge. Software certification is challenging even without the incorporation of autonomy. It is estimated that 70% of software cost is rework and certification.⁴ In addition, 70% of software faults are introduced early in the development cycle, but only 3.5% of those faults are found early; over 80% of faults are found after unit test when the estimated nominal cost for fault removal is 300-1000 times higher.⁵ Investing in V&V earlier in the design enables earlier error detection when the costs associated with a design change are much lower than later in the design process.

In a 2014 report,⁶ the Verification and Validation of Complex and Autonomous Systems (VVCAS) team at the Air Force Research Laboratory's Aerospace Systems Directorate described the need to expand the current V&V paradigm. In traditional systems engineering practices, V&V is conducted after the system design is complete via modeling, simulation, and test. As software complexity increases, depending on traditional V&V practices alone becomes intractable and infeasible. Formal methods-based analytical proofs could supplement and reduce reliance on traditional simulation and test evidence. The VVCAS team envisions a systems engineering practice that supplements traditional modeling, simulation, test, and evaluation techniques with analytical V&V evidence generated throughout the system engineering process. Following publication of the 2014 report, this vision was also adopted in DoD Autonomy Test and Evaluation, Verification and Validation (TEVV) Investment Strategy.⁷

III. Historic Spacecraft Software Failure Cases Addressable by Formal Methods

This section presents historical examples of spacecraft failures attributed to software errors that could have been found prior to flight had formal methods analysis been applied during the design process.⁸ Errors such as divide by zero, overflow, mixed precision, out-of-bounds array access, unreachable code, conflicting requirements, and conflicting units can be caught using a variety of formal methods tools and techniques.

One historical example of a mixed precision spacecraft software failure occurred on STS-49. During this Space Shuttle mission to rendezvous with and repair the Intelsat satellite, the Lambert Targeting Routine used to calculate rendezvous firings contained a mixed precision error that prevented the routine from converging and nearly caused the mission to be aborted.⁹ While the state-vector variables were double precision, the limits used to bound the calculation were single precision. The mission was recovered by a workaround that allowed state-vector information to be relayed from the ground. Later analysis showed that formal methods would have found this error prior to flight.⁸

Ariane 5 had a catastrophic failure that could have been prevented with the use of formal methods. The Ariane 5 is a European heavy lift launch vehicle that reused portions of Ariane 4 code on its first flight, Ariane 501. The software contained a bad 64 to 16 bit conversion which caused the vehicle trajectory to veer off course and prompted self-destruction of the system.¹⁰ Static formal-methods based analysis tools were successfully applied to formally verify portions of code for Ariane 502 and the following Ariane flights.¹¹

In 2004 the Mars Exploration Rover, Spirit, was unexpectedly shut down for 10 days after a parameter in the software permitted unlimited consumption of system memory as flash memory was exhausted. This error could have been caught had formal methods been applied.¹⁰

Perhaps one of the most famous spacecraft software failures occurred on the Mars Climate Orbiter (MCO) program, in which a conflict between English and metric units in the software resulted in loss of the system. According to the Mishap Investigation Board Phase I Report,¹² “The [Mars Climate Orbiter Mishap Investigation Board] has determined that the root cause for the loss of the MCO spacecraft was the failure to use metric units in the coding of a ground software file, “Small Forces,” used in trajectory models. Specifically, thruster performance data in English units instead of metric units was used in the software application code.” The requirements for the system specified that the data should be provided in metric units, and many modern formal methods tools such as The Specification and Analysis of Requirements (SpeAR) tool, provide unit checking capabilities to aid in identification of errors such as this.

The Mars Polar Lander mission provides an example of conflicting requirements. One requirement specified that alternative communication methods should be tested if no commands were received after 24 hours, while another requirement instructed the system to go into a “sleep mode” after 24 hours to conserve battery, making the alternate communication requirement unreachable.¹³ Formal methods tools can implement model checking to check for inconsistent requirements by evaluating realizability of a model given the constraining requirements.

The examples listed in this section are just a few of the many examples of software failure in spacecraft. Application of formal methods is a method that could have been used to prevent many of the historical software errors that have appeared in recent decades.

IV. 6U CubeSat Application

The AFIT 6U CubeSat reaction wheel array attitude control subsystem is used as a challenge problem for the formal methods toolset. This section provides background on CubeSats, the AFIT 6U CubeSat, reaction wheel dynamics, the equations of motion for the system, and a set of hypothetical requirements for a spacecraft attitude control subsystem.

A. CubeSats

CubeSats are small satellites built in units, or U, which are nominally $10 \times 10 \times 10 \text{ cm}^3$, or 1 liter, in volume and weigh 1.3 kilograms (3 pounds) or less. CubeSats initially began as a set of amateur radio experiments in the 1970s-1980s, matured as university experiments in the 1990s-2000s, and began being utilized for technology experiments and to carry out real missions in the late 2000s.¹⁴ More recently, CubeSats have been considered for increasingly complex missions including interplanetary and lunar missions such as NASA’s Lunar Flashlight mission, which seeks to determine presence or absence of exposed water ice on our Moon’s south pole.¹⁵ Complex missions can have precise pointing requirements, depending on the requirements of the satellite payload. While actuators exist for small CubeSat designs, larger actuators and accompanying control algorithms are needed to provide greater control authority for larger CubeSat designs.

B. CubeSats for DoD Missions

CubeSats are generally much less expensive to design, test, transport and launch than larger traditional satellites and therefore present a potential time and cost effective approach to meet DoD project needs.¹⁶ However, the limited size and weight of smaller CubeSats provide little space for mission-specific payloads and the advanced subsystems required to support them. To meet the increasing needs associated with anticipated DoD missions, AFIT's Center for Space Research and Assurance (CSRA) is currently developing 6U, 12U, and 27U CubeSats which promise larger payload capacity than the traditional 1U and 3U versions.¹⁷

C. Reaction Wheel Actuators

Reaction wheels are common actuators used for precision spacecraft attitude control that work by exchanging momentum with the spacecraft. By contrast, actuators such as thrusters or magnetic torque devices interact with the environment to apply an external torque to the spacecraft, which changes the spacecraft total angular momentum. In a torque free environment, the total angular momentum of the spacecraft remains constant. The angular momentum of the wheels is equal and opposite to the angular momentum that it exchanges with the spacecraft. Reaction wheels allows one to provide precise attitude control; however, once the maximum angular velocity rates of reaction wheels are reached, they are no longer able to control the spacecraft attitude in certain directions, a phenomenon known as saturation.

D. AFIT 6U CubeSat Testbed

AFIT's 6U CubeSat Attitude Determination and Control System (ADCS) Testbed, as pictured in Figure 1, includes the following components and subsystems: ADCS board, motor control board, reaction wheel array (RWA), electrical power system (EPS) board, battery pack, command data Handling (CDH) board with WiFly

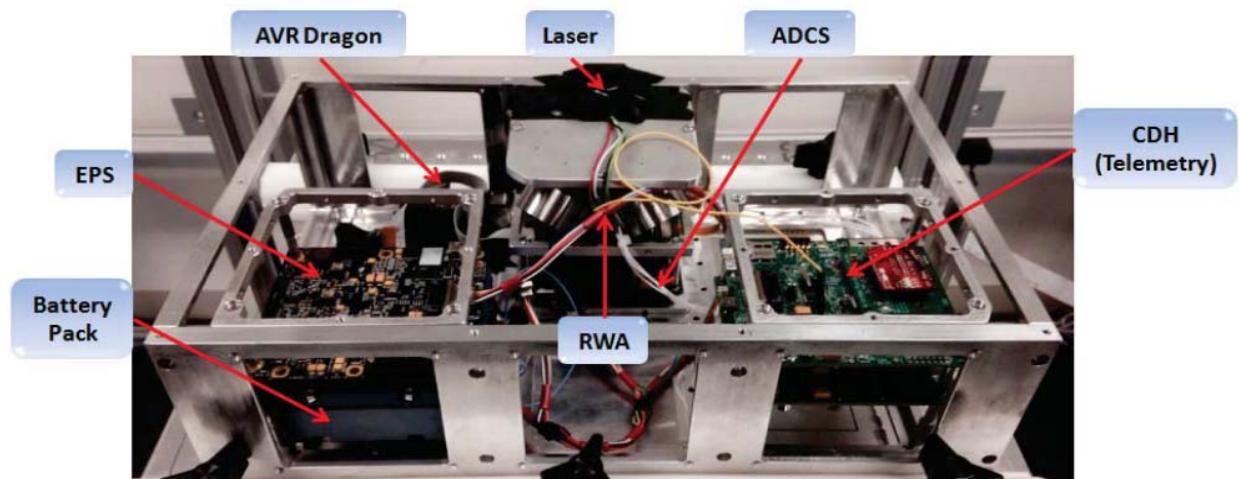


Figure 1. AFIT 6U ADCS Testbed¹⁸

capability, and a laser pointer. The AFIT reaction wheel array (RWA) implemented on the ADCS testbed includes four wheels in a pyramid configuration that provides redundancy if a failure were to occur in one of the wheels. The RWA is currently not flight qualified because it cannot survive launch vehicle conditions; however, its low friction and redundancy make it an excellent system for the laboratory. The AFIT CubeSat Four Wheel RWA is shown in Figure 2.

The reaction wheels of the AFIT four wheel pyramid RWA are not aligned with the principal axes of the testbed body frame. The angle between the reaction wheel momentum vector and the x-axis, represented as α , for wheels 1, 2, 3, and 4 is 315, 45, 135, and 225 degrees, respectively. The angle between the reaction wheel momentum vector and the z-axis, represented as β is 45 degrees.

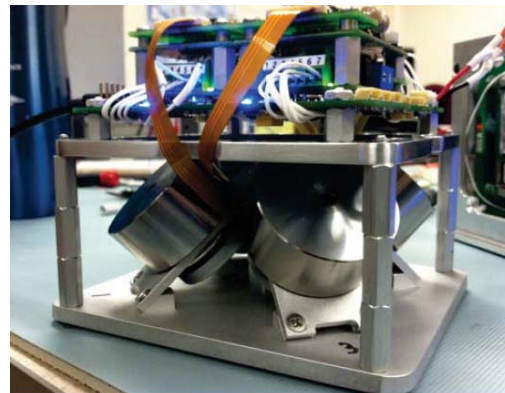


Figure 2. AFIT's CubeSat Four Wheel Pyramid Reaction Wheel Assembly¹⁹

E. Reaction Wheel Dynamics

The individual reaction wheel's angular momentum is:

$$\mathbf{h}_{rw} = D\psi_i \quad (1)$$

where (\mathbf{h}_{rw}) is equal to the wheel's mass moment of inertia (D) times the rate of spin (ψ_i) about the axis of the spin, where the subscript i corresponds to the particular reaction wheel.

A 4x3 torque mapping matrix is needed to transform the torque vector of each wheel to the body frame. The coordinate system is shown in Figure 3. The angular momentum of the reaction wheel array is given by:

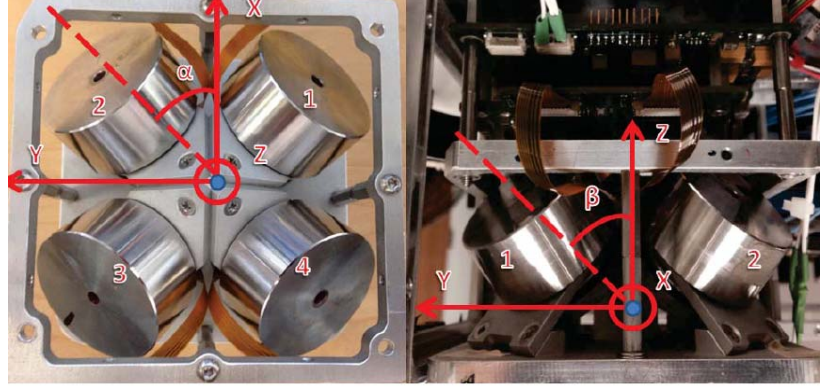


Figure 3. Four Wheel Pyramid RWA Coordinate System¹⁸

$$\mathbf{h}_{rwa} = \mathbf{S}\boldsymbol{\psi} \quad (2)$$

where \mathbf{h}_{rwa} is the angular momentum vector of the reaction wheel array, $\boldsymbol{\psi}$ is an array containing the angular velocity of each of the reaction wheels, and \mathbf{S} is a matrix that orients the angular momentum contribution of each reaction wheel with the principal axis of the spacecraft. The \mathbf{S} matrix is defined as:

$$\mathbf{S} = D \begin{bmatrix} \cos\alpha_1 \sin\beta & \cos\alpha_2 \sin\beta & \cos\alpha_3 \sin\beta & \cos\alpha_4 \sin\beta \\ \sin\alpha_1 \sin\beta & \sin\alpha_2 \sin\beta & \sin\alpha_3 \sin\beta & \sin\alpha_4 \sin\beta \\ \cos\beta & \cos\beta & \cos\beta & \cos\beta \end{bmatrix}. \quad (3)$$

When the body frame axes are aligned with the principal axes, the MOI of the body of the spacecraft can be diagonalized. The angular momentum of the spacecraft is given by:

$$\mathbf{h}_{s/c} = \mathbf{I}\boldsymbol{\omega} \quad (4)$$

where $\mathbf{h}_{s/c}$ is the angular momentum of the spacecraft, \mathbf{I} is the moment of inertia of the spacecraft, and $\boldsymbol{\omega}$ is the angular velocity vector of the spacecraft. The total angular momentum of the spacecraft is given by:

$$\mathbf{H}_{tot} = \mathbf{h}_{s/c} + \mathbf{h}_{rwa}. \quad (5)$$

The relationship between external moments and the rate of change of angular momentum in the body frame is:

$$\mathbf{M}_{ext} = \dot{\mathbf{H}} = \dot{\mathbf{h}}_{s/c} + \boldsymbol{\omega}^\times \mathbf{h}_{s/c} + \dot{\mathbf{h}}_{rwa} + \boldsymbol{\omega}^\times \mathbf{h}_{rwa} = \mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega}^\times \mathbf{I}\boldsymbol{\omega} + \dot{\mathbf{h}}_{rwa} + \boldsymbol{\omega}^\times \mathbf{h}_{rwa}. \quad (6)$$

where $\boldsymbol{\omega}^\times$ is the skew-symmetric matrix of spacecraft angular rates.

F. Proportional –Derivative Control Equations

Quaternion error is computed from:

$$\mathbf{q}_e = \begin{bmatrix} q_{4c} & -q_{3c} & q_{2c} & q_{1c} \\ q_{3c} & q_{4c} & -q_{1c} & q_{2c} \\ -q_{2c} & q_{1c} & q_{4c} & q_{3c} \\ -q_{1c} & -q_{2c} & -q_{3c} & q_{4c} \end{bmatrix} \mathbf{q}_p. \quad (7)$$

where q_{ic} terms are the commanded quaternions, \mathbf{q}_p is the present quaternion array, and \mathbf{q}_e is the error quaternion vector.

The control law for the change in angular momentum of the reaction wheel can also be expressed as:

$$\dot{\mathbf{h}}_{\text{rwa}} = K_P \mathbf{I} \mathbf{q}_{e(123)} + K_D \mathbf{I} \boldsymbol{\omega} - \boldsymbol{\omega}^\times (\mathbf{I} \boldsymbol{\omega} + \mathbf{h}_{\text{rwa}}) \quad (8)$$

where $\dot{\mathbf{h}}_{\text{rwa}}$ is the reaction wheel array angular momentum change, K_P and K_D are the proportional and derivative gains of the controller, and $\mathbf{q}_{e(123)}$ is a vector of the first three values of the quaternion error vector.

The resulting reaction wheel commanded angular acceleration is calculated as:

$$\dot{\boldsymbol{\psi}}_{\text{com}} = \mathbf{S}_{\text{pinv}} \dot{\mathbf{h}}_{\text{rwa}} \quad (9)$$

where $\dot{\boldsymbol{\psi}}_{\text{com}}$ is a vector of the commanded change in angular velocity for each of the four reaction wheels, and \mathbf{S}_{pinv} is the pseudoinverse of the \mathbf{S} matrix.

G. Equations of Motion

The state vector \mathbf{x} is defined here as:

$$\mathbf{x} = [\mathbf{q} \ \boldsymbol{\omega} \ \boldsymbol{\psi}]^T = [q_1 \ q_2 \ q_3 \ q_4 \ \omega_1 \ \omega_2 \ \omega_3 \ \psi_1 \ \psi_2 \ \psi_3 \ \psi_4]^T. \quad (10)$$

The system can be expressed with as¹²:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{q}} \\ \dot{\boldsymbol{\omega}} \\ \dot{\boldsymbol{\psi}} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{4 \times 4} & 0.5 \mathbf{Q}_{4 \times 3} & \mathbf{0}_{4 \times 4} \\ \mathbf{0}_{3 \times 4} & -\mathbf{I}^{-1} \boldsymbol{\omega}^\times \mathbf{I} & -\mathbf{I}^{-1} \boldsymbol{\omega}^\times \mathbf{S} \\ \mathbf{0}_{4 \times 4} & \mathbf{0}_{4 \times 3} & \mathbf{L} \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{0}_{4 \times 3} \\ \mathbf{I}^{-1} \\ \mathbf{0}_{4 \times 3} \end{bmatrix} \mathbf{S} \dot{\boldsymbol{\psi}} + \begin{bmatrix} \mathbf{0}_{4 \times 3} \\ \mathbf{I}^{-1} \\ \mathbf{0}_{4 \times 3} \end{bmatrix} \mathbf{m}_{\text{ext}} + \begin{bmatrix} \mathbf{0}_{4 \times 4} \\ \mathbf{0}_{3 \times 4} \\ \mathbf{U}_{4 \times 4} \end{bmatrix} \dot{\boldsymbol{\psi}} \quad (11)$$

where $\mathbf{U}_{4 \times 4}$ is an identity matrix and $\mathbf{Q}_{4 \times 3}$ is:

$$\mathbf{Q}_{4 \times 3} = \begin{bmatrix} q_4 & -q_3 & q_2 \\ q_3 & q_4 & -q_1 \\ -q_2 & q_1 & q_4 \\ -q_1 & -q_2 & -q_3 \end{bmatrix}. \quad (12)$$

H. Requirements Set

Hypothetical requirements are presented here for the 6U CubeSat highlighting the limitation of the actuators, the pointing accuracy of the control system, slewing range and rate, drift, settling time, rise time, and percent overshoot. The first two requirements are derived requirements that arise once a RWA is selected to actuate the satellite. These two requirements constrain the output of the control algorithm to prevent damage to the reaction wheel motors:

- R01 The commanded time rate of change of angular velocity shall not exceed the maximum allowable angular acceleration of the reaction wheel.
- R02 The commanded angular velocity shall not exceed the maximum allowable angular velocity of the reaction wheel.

The remaining requirements describe the expected performance of the attitude control subsystem in the context of its ability to control the 6U CubeSat attitude:

- R03 The pointing accuracy shall be at least 1 degree as a threshold and 0.08 degrees as an objective.
- R04 The pointing range about the z axis shall be 0 to 360 degrees.
- R05 The pointing range about the y axis shall be 0 to 360 degrees.
- R06 The pointing range about the x axis shall be 0 to 360 degrees.
- R07 The slew rate shall be > 3 deg/sec as a threshold and > 7 deg/sec as an objective.
- R08 After settling, the drift rate shall be < 3 deg/min as a threshold and < 1 deg/min as an objective.
- R09 After settling, the total drift shall be <= ± 0.5 degrees as a threshold and <= ± 0.1 degrees as an objective.
- R10 The 5% settling time shall be <= 5 minutes as a threshold and <= 2 minutes as an objective.
- R11 The 2% settling time shall be <= 7 minutes as a threshold and <= 3 minutes as an objective.
- R12 The rise time shall be <= 5 minutes as a threshold and <= 2 minutes as an objective.
- R13 The percent overshoot shall be <= 50% as a threshold and <= 25% as an objective.

V. Methodology

This section describes the compositional structure of the attitude control subsystem and the toolset proposed for the analysis of the subsystem requirements, architecture, and model. For consistency across all of the tools, the governing equations of the systems are discretized with a fixed time step.

A. Compositional Structure

A compositional structure is critical for efficient formal methods analysis and system modularity. Breaking up a system compositionally enables formal methods analysis at the lowest component levels of the design to be composed into subsystem and system level certification arguments. In addition, a compositional structure promotes modularity by allowing components such as an actuator or sensor to be changed and analysis to be completed on directly impacted components only up to the system level rather than completing the entire system analysis over again.

For this research, the satellite is broken up compositionally into a controller, which contains the controller software, and the environment, which includes all the hardware that the controller interacts with. The environment is further functionally decomposed into the reaction wheel array and the structure. The sensors have been abstracted and it is assumed that the controller receives perfect truth data. The reaction wheel is broken down one more level to the individual reaction wheels. The flow of signals between the components is depicted in Figure 4. The simulated environment outputs the current system state including the orientation of the spacecraft in a quaternion vector, the rotation rates of each of the major axes of the spacecraft, and the rotation rates of each of the individual reaction wheels. At each step in the simulation of the system, a unit delay is placed between the output of the environment and the controller. The controller receives the state information from the previous time step and uses that to calculate a commanded angular acceleration with a proportional-derivative controller for each of the reaction wheels in order to approach and eventually achieve the desired spacecraft orientation. The angular acceleration commands are received and implemented by each of the individual reaction wheels. The reaction wheels then contribute angular momentum to the reaction wheel array which exchanges its total angular momentum with the spacecraft structure. The structure then updates the state of the spacecraft and the environment outputs the system state.

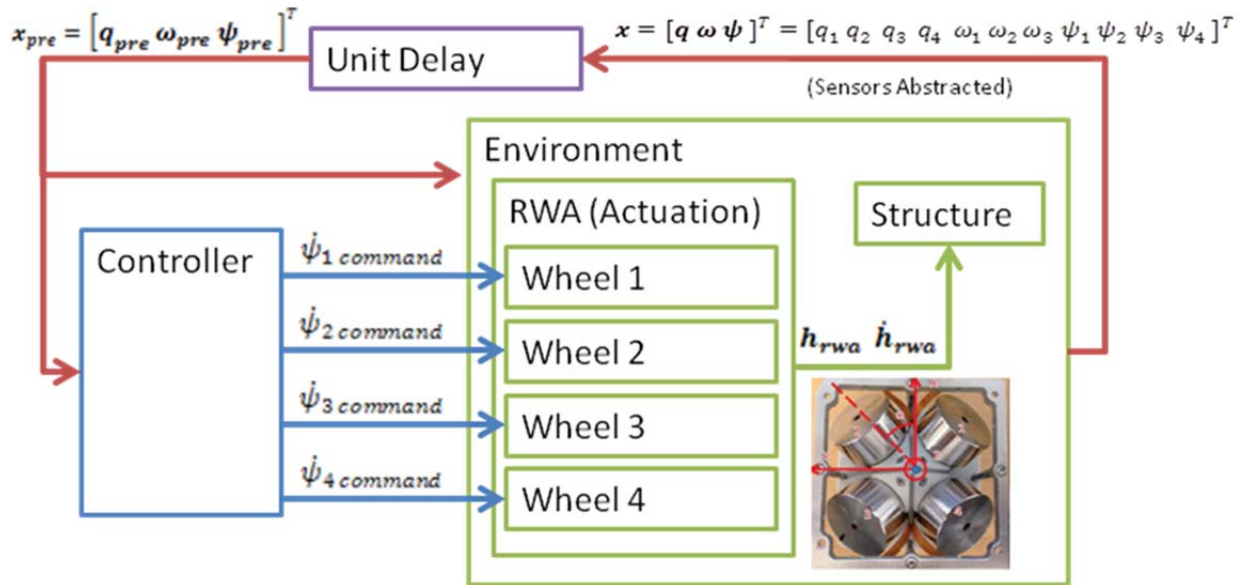


Figure 4. 6U CubeSat Compositional Structure and Signal Flow

B. System Requirements and SpeAR

The requirements for the control system are expressed in a mathematically rigorous framework currently under development by Rockwell Collins and AFRL called the Specification and Analysis of Requirements (SpeAR) tool. The SpeAR tool allows subject matter experts to write system requirements in peer-reviewed specification templates²¹ that allow for formal analysis. This part of the process serves two purposes. First, expressing requirements in a formalized notation gives them a precise, mathematical meaning. This requires engineers to unambiguously state what the true meaning of each requirement is. Second, SpeAR translates formalized requirements into logical expressions that can be analyzed using model checking techniques.

By comparing system requirements to high level system properties in a model checker, requirements and derived requirements added throughout the design of the system can be analyzed for completeness and consistency. The properties and requirements documented in SpeAR are translated into a Lustre²² model and sent to JKind,²³ a k-induction model checker.²⁴ JKind uses Satisfiability Modulo Theory (SMT) solvers, such as Z3,²⁵ to inductively prove that a given model meets its formal specification. In the instance of SpeAR, the model is defined by the formalized system requirements and the specification defined by its properties. JKind will either report that the system's properties are satisfied by the given requirements, or it will provide a sequence of inputs, referred to as a counterexample, that demonstrates how a property can be violated. This counterexample can be used by domain experts to refine either the requirements or the properties until a proof is obtained.

An example requirement written in SpeAR is shown in Figure 5. The first requirement of the reaction wheel attitude control system is that the acceleration command from the controller should not exceed the maximum allowable acceleration of the reaction wheel. As discussed in Section IV. H., Requirement 1 is a safety property set by the physical limitations of the motor, control system, and power supply, and is designed to prevent damage to the reaction wheel. The requirement is broken out for each individual reaction wheel, and the first wheel sub requirement is documented as "r_01a_ctrl", which is a naming convention where "r" indicates that it is a requirement versus an assumption, "01" is the number of the requirement, "a" is the first sub requirement, and "ctrl" denotes that it is a controller subsystem requirement. The requirement is interpreted formally as: under any conditions (globally) the commanded change of angular velocity (as calculated in equation 9 and represented here as psi_dot_comm_1) is always less than or equal to the maximum allowable angular acceleration of the system (MAX_PSI_DOT_RW) and greater than or equal to the negative of the maximum allowable angular acceleration. Both the negative and positive limits are included because they represent the maximum velocity in different directions.

```
//R01 The commanded change of angular velocity
from the control algorithm shall not exceed the
maximum allowable angular acceleration of the
reaction wheel.
r_01a_ctrl = global :: always psi_dot_comm_1 <=
MAX_PSI_DOT_RW and psi_dot_comm_1 >= -
MAX_PSI_DOT_RW; //[rad_p_s_2]
```

Figure 5. Example requirement specified in SpeAR

One of the strengths of SpeAR is the ability to express and analyze non-functional requirements that describe non-functional properties. For instance, the ADCS for satellites usually has strict requirements on maximum weight, cost, and power consumption. If the ADCS is comprised of a microcontroller, reaction wheel array, and magnetic torque coil, designers can specify requirements for each component and verify that the sum each of the individual components does not violate the overall ADCS requirements, e.g. the total cost of the individual components is less than the allowable cost for the subsystem. While this capability may seem trivial, it is valuable on a larger scale for complex systems that may have thousands of individual components distributed among dozens of subsystems. The compositional nature of SpeAR allows designers to pull component-level verification results up to complete subsystem-level verification, and subsystem-level verification up to complete system-level verification. Another valuable analysis feature of note in SpeAR is the ability to define units used throughout the system and analyze compliance.

C. System Architecture and AGREE

The properties and requirements expressed in SpeAR are then used to develop the system architecture in the Architecture Analysis & Design Language (AADL). AADL was standardized by SAE in 2004 as an architecture modeling framework to allow analysis of system designs prior to detailed development.²⁶ AADL allows for the inclusion of annexes for additional functionality. The Assume Guarantee Reasoning Environment (AGREE) AADL annex²⁷ was developed by Rockwell Collins and University of Minnesota as part of the DARPA High Assurance Cyber Military Systems (HACMS) program to use assume-guarantee contracts to evaluate the behavior of the subsystem components with the greater system. Properties in SpeAR are documented as guarantees of the system in AGREE, and requirements developed in SpeAR are used to build the system behavior. Like SpeAR, AGREE utilizes JKind and Z3 to conduct model checking and theorem proving to assure that the guarantees hold. Again, like SpeAR, Z3 outputs proofs or counterexamples

```
-- R01 The commanded change of angular velocity
from the control algorithm shall not exceed the
maximum allowable angular acceleration of the
reaction wheel.
guarantee "r_01a_ctrl: RW1 commanded angular
acceleration shall not exceed maximum allowable
angular acceleration" :
    (psi_dot_comm_1 <= MAX_PSI_DOT_RW) and
    (psi_dot_comm_1 >= -MAX_PSI_DOT_RW);
```

Figure 6: Example guarantee specified in AGREE

that your architecture fulfills the guarantees with the given assumptions. The example requirement in Figure 5 is expressed in AGREE as shown in Figure 6. In the AGREE annex, the statement in quotation marks is what is output as proven or falsified in the analysis, but is not actually analyzed. It is proven if the statement listed below the quoted section is proven to be true by the model checker. If the guarantee is falsified, a counterexample is produced showing how the system architecture violated the guarantee.

D. System Model and Simulink Design Verifier

The verified architecture provides the framework for a Simulink model and simulation of the control system. The assumptions and guarantees developed during the architecture phase are used to manually write the Simulink files. The model is then analyzed using Simulink Design Verifier to prove that the properties (documented as guarantees in AGREE, and as properties and requirements in SpeAR) hold true throughout the modeling phase. The same requirement example from Figures 5 and 6 is carried through into Figure 7, which shows the requirement specified as a property in Simulink Design Verifier.

In Simulink, like in SpeAR and AGREE, the model was broken down from the system level into sublevels of the Controller and Environment as described in Section V. A. The file structure in Simulink was designed to mimic the architectural breakdown. Simulink model reference blocks were used to reference each of the subsystems and compose the entire system. Three files are included in each of the subsystem folders: the Simulink model of the subsystem, a model of the properties of the subsystem, and a verification model that references the subsystem and properties models with model reference blocks in order to conduct the formal analysis. The file structure is shown in Figure 8. The benefit of using model reference and this file structure is the ability to quickly to substitute an alternative implementation for a subsystem to compare performance both in simulation and verification analysis.

```
%R01 The commanded change of angular
velocity from the control algorithm
shall not exceed the maximum
allowable angular acceleration of the
reaction wheel.
```

```
r_01a_ctrl = (psi_dot_comm_1 <=
MAX_PSI_DOT_RW) && (psi_dot_comm_1 >=
-MAX_PSI_DOT_RW); %[rad_p_s_2]
```

Figure 7: Example property specified in Simulink Design Verifier

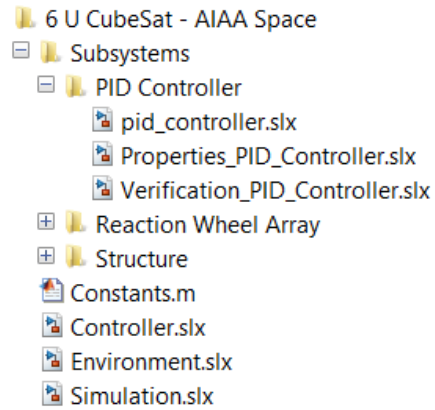


Figure 8: Compositional file structure in Simulink

VI. Analysis

SpeAR, AGREE, and Simulink Design Verifier were applied to the requirements, architecture, and model of a 6U CubeSat Reaction Wheel Assembly design owned by the Air Force Institute of Technology (AFIT). This section discusses the limitations of the nonlinear systems analysis and provides an example of non-functional system analysis.

A. Derived Requirements

The first two requirements, which describe the output commands of the control algorithm can be analyzed by the tools. In SpeAR, the following derived requirements are described formally:

- If the commanded acceleration would cause the reaction wheel to exceed its maximum velocity, the acceleration command is set to zero.
- If the commanded acceleration would exceed the maximum allowable acceleration, the acceleration command is set to the maximum allowable.

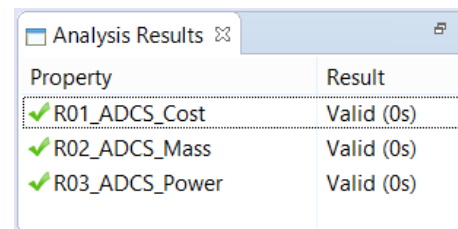
SpeAR searches the space defined by these derived requirements for an instance in which R01 and R02 can be violated. In the next design stage, these derived requirements become part of the architecture implementation in AADL which is searched for instances in which the guarantees described by R01 and R02 can be violated. In Simulink, these derived requirements are implemented at the end of the control algorithm to check and constrain the algorithm output. These constraints are used to prove R01 and R02 hold true as properties of the model.

B. Nonlinear Systems Analysis

The control equations (8 and 9) and equations of motion (11) for the 6U CubeSat reaction wheel attitude control example discussed in this paper are nonlinear. These nonlinear equations are not well suited for analysis in SpeAR, AGREE, or Simulink Design Verifier because none of these tools currently support nonlinear math; however, SpeAR and AGREE allow users to select which SMT solver to use for the formal methods analysis, and Z3 is an option that provides some support for nonlinear math, so it is possible to expand SpeAR and AGREE capabilities to leverage more of Z3's capabilities in the future. While many of the requirements can be expressed in the tools, very limited analysis can be completed because nonlinear control equations and equations of motion define whether or not the requirements are met; however SpeAR, AGREE, and Simulink Design Verifier may be able to conduct more complete analysis on a linearized version of this model. For instance, requirements 4, 5, and 6 deal with controllability of the system, and could be easily expressed and analyzed for a linear system. A linear system can be determined to be completely controllable, or able to reach any state, if the controllability matrix associated with that system is of full rank. This determination is not trivial for nonlinear control systems, though methods do exist and could be explored in the future.

C. Non-Functional Requirement Analysis

While functional requirements describe the behavior of a system or subsystem, non-functional requirements describe properties such as cost, power, and weight that can often be described mathematically. Analysis was conducted on a sample attitude control subsystem that contained a reaction wheel array, listed as RWA, and a magnetic torque actuator, described as MT. In SpeAR, the high level subsystem requirements are listed as "Properties," while derived requirements that describe the components selected to meet subsystem requirements are listed as "Requirements." In this example, the cost, mass, and power of the RWA and MT are shown to meet the total subsystem requirements. The analysis shows that all properties are proven valid, as shown in Figure 9. The requirements and properties written in SpeAR are shown in Figure 10.



Property	Result
✓ R01_ADCS_Cost	Valid (0s)
✓ R02_ADCS_Mass	Valid (0s)
✓ R03_ADCS_Power	Valid (0s)

Figure 9: Non-functional requirement analysis example results in SpeAR

```
Requirements: //Derived Requirements at the subsystem level

// The RWA shall not cost more than $210K excluding software development and labor
R01_ADCS_Cost_RWA = global :: always cost_RWA <= 210000.0;
// The RWA shall not exceed a mass of 1.1 kg
R02_ADCS_Mass_RWA = global :: always mass_RWA <= 1.1;
// The RWA shall not require more than 16 W of power
R03_ADCS_Power_RWA = global :: always power_RWA <= 16.0;

// The MT shall not cost more than $210K excluding software development and labor
R01_ADCS_Cost_MT = global :: always cost_MT <= 210000.0;
// The MT shall not exceed a mass of 1.1 kg
R02_ADCS_Mass_MT = global :: always mass_MT <= 1.1;
// The MT shall not require more than 30 W of power
R03_ADCS_Power_MT = global :: always power_MT <= 13.0;

//Total ADCS Subsystem Power
R01_ADCS_Power_total = global :: always power == power_RWA+power_MT;
R02_ADCS_Cost_total = global :: always cost == cost_RWA+cost_MT;
R03_ADCS_Mass_total = global :: always mass == mass_RWA+mass_MT;

Properties: //System Level Requirements
// The ADCS shall not cost more than $420K excluding software development and labor
R01_ADCS_Cost = global :: always cost <= 420000.0;
// The ADCS shall not exceed a mass of 2.3 kg
R02_ADCS_Mass = global :: always mass <= 2.3;
// The ADCS shall not require more than 30 W of power
R03_ADCS_Power = global :: always power <= 30.0;
```

Figure 10: Non-functional requirement analysis example in SpeAR

VII. Recommendations

A. Toolset Development Recommendations

Throughout this research, several gaps were identified in the SpeAR, AGREE, and Simulink Design Verifier tools that could greatly improve their usability and applicability to complex systems such as the 6U CubeSat attitude control system. While most recommendations are specific to individual tools, all of the tools could improve with nonlinear math support. The largest barrier for the 6U CubeSat requirements analysis is the lack of support for nonlinear math in SpeAR, AGREE, and Simulink Design Verifier, making exhaustive proof of the 6U CubeSat requirements infeasible without first linearizing the system.

SpeAR and AGREE could both be improved with support for matrix multiplication and a built-in notion of time. In the case of the 6U CubeSat, the PID controller and equations of motions for the satellite are most succinctly expressed and easiest to debug in matrix form. Without matrix support, these equations become very long, and it is more difficult to identify errors. As many of the requirements for this system and other control systems deal with system performance over time, adding a notion of time to SpeAR and AGREE would also be exceptionally useful. One workaround for this gap is to include time as an additional state that increments by the time step defined by the system rate. A built-in notion of time would simplify the expression of time-dependent properties.

SpeAR could also be improved with changes to the interface and unit expression capabilities. The current version of SpeAR allows the user to define the interface of the subsystem with named and typed inputs and outputs. However, while it currently allows inputs to be expressed separately, outputs are currently defined as a unique type that contains all the outputs of the system. Future versions of SpeAR will allow the user to express each output of a system separately. One of the promising features of SpeAR is its ability to perform unit checking on basic units. However, when more complex units were utilized in this example, SpeAR was not able to perform correct unit checking. This issue will also be addressed in the next version of SpeAR.

One of the gaps in the process presented in this paper is quantifying the robustness of the analysis to calculation errors. Future work could also be conducted in making guarantees about the robustness of the system to errors in the floating point math used to estimate the equations of motion. Using a tool such as ASTREE²⁸, to quantify the worst case floating point error for each calculation, the designer could use the model checker to explore the tolerance of the guarantees to that error.

B. Next steps for the research

The next step in this research is to linearize the 6U CubeSat model, compare its performance to the nonlinear model, and conduct the formal methods analyses described in this paper. In addition, the abrupt saturation limits of the reaction wheels will need to be addressed in the formal methods analysis. Next, C code from the Simulink model will be analyzed using Mathworks Polyspace Bug Finder and Polyspace Code Prover. The C code will be implemented on the AFIT 6U CubeSat Testbed attitude control subsystem and tests will be conducted to characterize the actual performance of the system. These test results will be combined with the simulation and formal methods analysis results to create an assurance case.

VIII. Conclusions

Conducting formal methods analysis throughout the design process could enable earlier fault detection, and supplement existing simulation and testing processes. A compositional architecture enables modular analysis and consequentially the reuse of verified components with limited analysis and testing. The formal methods tools evaluated in this research are becoming more capable and usable, but still have several gaps. The tools used to analyze the requirements, architecture and model of the 6U CubeSat reaction wheel array attitude control subsystem are sufficient to express system behavior and requirements that can be defined with simple math and inequalities; however, a large gap still remains in their nonlinear system analysis capabilities making them suitable for some applications, but not complete for expressing the complexity of the spacecraft attitude control example presented in this paper.

References

¹Kathleen Fisher, "Using Formal Methods to Enable More Secure Vehicles: Tufts University", 16 September, 2014 DARPA's HACMS Program, URL: <http://wp.doc.ic.ac.uk/riapav/wp-content/uploads/sites/28/2014/05/HACMS-Fisher.pdf> [cited 27 Jul. 2015].

- ²Fifarek, A., Gross, K., Hoffman, J., "Basic Two Controlled Tanks System: A case study to investigate the process to develop a formal and traceable paradigm to generate Requirements/Architectures/Models for System Design," Wright-Patterson AFB, OH, 27 May 2015.
- ³Dahm, W. J. A. "Technology Horizons a Vision for Air Force Science & Technology During 2010-2030," USAF HQ, 2010.
- ⁴Feiler, P. H. "Supporting the ARP4761 Safety Assessment Process with AADL," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 6 Feb. 2014, URL: <https://wiki.sei.cmu.edu/aadl/images/1/13/ERTSEMV2-Feb2014.pdf> [cited 27 May 2015].
- ⁵NIST Planning report 02-3, "The Economic Impacts of Inadequate Infrastructure for Software Testing," May 2002.
- ⁶Clark, M., Kearns, K., Overholt, J., Gross, K., Barthelemy, B., Reed, C., "Air Force Research Laboratory Test and Evaluation, Verification and Validation of Autonomous Systems Challenge Exploration," Air Force Research Laboratory, Wright-Patterson AFB, OH, 13 Nov. 2014.
- ⁷Assistant Secretary of Defense / Research and Engineering (ASD/R&E), Autonomy Community of Interest (COI), "DoD Autonomy Test and Evaluation," Verification and Validation (TEVV) Investment Strategy, Alexandria, VA, Jun. 2015, URL: http://www.defenseinnovationmarketplace.mil/resources/OSD_ATEVV_STRAT_DIST_A_SIGNED.pdf [cited 27 July 2015].
- ⁸McAllister, B. "NASA Future Challenges in Formal Methods," *Sixth NASA Formal Methods Symposium*, Houston, TX, 29 Apr. 2014.
- ⁹"An Assessment of Space Shuttle Flight Software Development Processes" Committee for the Review of Oversight Mechanisms for Space Shuttle Flight Software Processes, National Research Council. National Academy Press, Washington D.C., 1993, URL: <http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19930019745.pdf> [Accessed 20 May 2015].
- ¹⁰Denney, E. "Formal Methods for the Certification of Auto-generated Flight Code" NASA Ames Research Center, URL: [http://ti.arc.nasa.gov/m/pub-archive/1400h/1400%20\(Denney\).pdf](http://ti.arc.nasa.gov/m/pub-archive/1400h/1400%20(Denney).pdf) [Accessed 11 May 2015].
- ¹¹Lacan, P., Monfort, J. N., Ribal, L.V.Q., Deutsch, A., Gonthier, G., "Ariane 5: The software reliability verification process: the Ariane 5 Example" from *Proceedings for the DASIA '98 Conference on 'Data Systems in Aerospace'*, Athens, Greece, 1998.
- ¹²NASA. Mars Climate Orbiter: Mishap Investigation Board, NASA Headquarters, Washington, DC, USA, 1999, URL: ftp://ftp.hq.nasa.gov/pub/pao/reports/1999/MCO_report.pdf [Accessed 11 May 2015].
- ¹³Johnson, C.W., "The Natural History of Bugs: Using Formal Methods to Analyse Software Related Failures in Space Missions," *Formal Methods Europe* (2005), pp. 9-25.
- ¹⁴Samson, J. R., "Thoughts on Fault Tolerance in CubeSats and Small Sats" *The Eight Workshop on Fault-Tolerant Spaceborne Computing Employing New Technologies*, Albuquerque, NM, May 2015.
- ¹⁵Cohen, B. A., Hayne, P. O., Paige, D. A., Greenhagen, B. T., "Lunar Flashlight: Mapping Lunar Surface Volatiles using a CubeSat," *Annual Meeting of the Lunar Exploration Analysis Group*, 2014.
- ¹⁶Wertz, J. R., Everett, D. F., Pushcell, J.J. "Space Mission Engineering: The New SMAD," Microcosm Press, 2011.
- ¹⁷Swenson, E. D. "6U CubeSat – Design, Build, Test," Air Force Institute of Technology, 2014.
- ¹⁸Tibbs, M., "Design and Test of an Attitude Determination and Control System for a 6U CubeSat using AFIT's CubeSat Testbed," M.S. Thesis, Department of Aeronautics and Astronautics, Air Force Institute of Technology, Wright-Patterson AFB, OH, Mar. 2015.
- ¹⁹Dannemeyer, E., "Design and Analysis of an Attitude Determination and Control Subsystem (ADCS) for AFIT's 6U Standard Bus, M.S. Thesis," Department of Aeronautics and Astronautics, Air Force Institute of Technology, Wright-Patterson AFB, OH, Mar. 2014.
- ²⁰Swenson, E., "REBEL RWA Analysis," Air Force Institute of Technology, Wright-Patterson AFB, OH, 4 May 2015.
- ²¹Dwyer, M. B., Avrunin, G. S., Corbett J. C., "Property specification patterns for finite-state verification," *Proceedings of the Second Workshop on Formal Methods in Software Practice*, ACM, 1998, pp. 7-15.
- ²²Halbwachs, N., Caspi, P., Raymond, P., & Pilaud, D. "The synchronous data flow programming language LUSTRE," *Proceedings of the IEEE*, Vol. 79., No. 9, 1991, pp. 1305-1320.
- ²³"Agacek/jkind," URL: <https://github.com/agacek/jkind> [cited 14 Jan. 2015].
- ²⁴De Moura, L., Ruess, H., Sorea, M., "Bounded Model Checking and Induction: From Refutation to Verification," *Proceedings of the 15th International Conference on Computer Aided Verification*, Vol. 2725, 2003.
- ²⁵Z3, Open Source Software, Microsoft, URL: <https://z3.codeplex.com/> [cited 11 May 2015].
- ²⁶"Architecture Analysis and Design Language," URL: <http://www.sei.cmu.edu/architecture/research/model-based-engineering/aadl.cfm> [cited 14 Jan. 2015].
- ²⁷Whalen, M. W., Gacek, A., Cofer, D., Murugesan, A., Heimdahl, M., Rayadurgam, S., "Your 'What' Is My 'How': Iteration and Hierarchy in System Design," *IEEE Software*, Vol.30, No. 2, 2013, pp. 54-60.
- ²⁸Cousot, P., Cousot, R., Feret, J., Miné, A., Rival, X., "The Astrée Static Analyzer," URL: <http://www.astree.ens.fr/> [cited 20 Jul. 2015].