Air Vehicle Model-Based Design and Simulation Pilot

Henson Graves
Stephen Guest
Jeff Vermette
Yvonne Bijan
Harold Banks
Greg Whitehead
Bill Ison
Lockheed Martin Aeronautics
P.O. Box 748
Fort Worth, TX 76101
817-935-5911
henson.graves@lmco.com

Keywords:
SysML, Simulation-Based Design, Model-Based System Engineering, dynamic modeling, capability analysis, statistical methods

ABSTRACT: *Integrating design models with operational simulations allow engineers to obtain quantitative results about design performance and incorporate the results in the design process far in advance of implementation. To demonstrate the technological maturity of modeling and simulation-based product design, this pilot effort developed executable SysML design models for a 1553 avionics architecture. These models were integrated with behavioral/performance models of a production model aircraft operating in a digital environment. The 1553 model provides a link between a model of new avionics functionality (e.g., terrain following, obstacle avoidance) with a model of an existing avionics system. The 1553 design model represents the behavior of message flow between devices such as sensors and displays and specifies actual or desired performance such as message transmission times. The 1553 design model hosted actual terrain following software and the integrated simulations were used to determine how well the system would perform given various designs for collecting and fusing sensor data. One benefit of this design modeling approach is full documentation of simulation experiments including documentation of operational and design models as well as the operational scenarios. This development effort successfully met the technical challenge of cost effectively building design models with sufficient fidelity to provide valid results.*

1. Introduction

The Design Space Mapping Pilot objective is to demonstrate that the integration of operational simulation with development of executable design models is sufficiently mature to form the core of product design processes. The design task executed by the Pilot added a new capability to an existing air vehicle production model. To demonstrate technological maturity of integrating design with simulation, the data results must have sufficient fidelity and must be cost effectively obtainable. The challenge is in choosing the appropriate level of fidelity for the models. If the models have insufficient fidelity then the results have limited utility. However, high fidelity models may be very expensive to obtain or build. For the Pilot, we built high fidelity behavioral models of existing hardware and software. The ability to build these models cost effectively validates the feasibility of the approach. We used a layered software architecture for the design models which allows increasingly higher fidelity of performance models to be substituted, as they are needed or become cost effective to generate. The layers also allow for an established baseline that can be reused for multiple trade studies. The results of executing these models provided a depth of analysis that other wise would have been unavailable until the product was built. The Pilot paralleled an ongoing trade study and had access to subject mater experts. The trade study allowed the Pilot to be completely realistic in its task.

1.1 Integration of design modeling with operational simulation

The Pilot integrates model-based design with simulation-based analysis. The integration is achieved by developing executable design models and operating

them in a simulation environment. The team used Rhapsody to develop and execute the SysML design models. The Rhapsody execution environment was integrated with the operational models through shared memory. The result was an integrated environment containing dynamic models of the design, as well as actual flight software, behavioral models for an air vehicle, and models of the operating environment. The simulation integration provides quantitative results about design behavior that can be incorporated early in the design process well in advance of implementation.

Product development or upgrades typically start with operational capability needs and progresses through design to implementation and verification. Analysis is performed at each stage of development, first to solidify functional and subsystem requirements and verify feasibility, then as the design is synthesized to establish that the design meets its requirements [1]. Developing a product design by constructing and iterating design models for a system is now common practice in both software and physical design [2]. Simulation has traditionally been used for specific

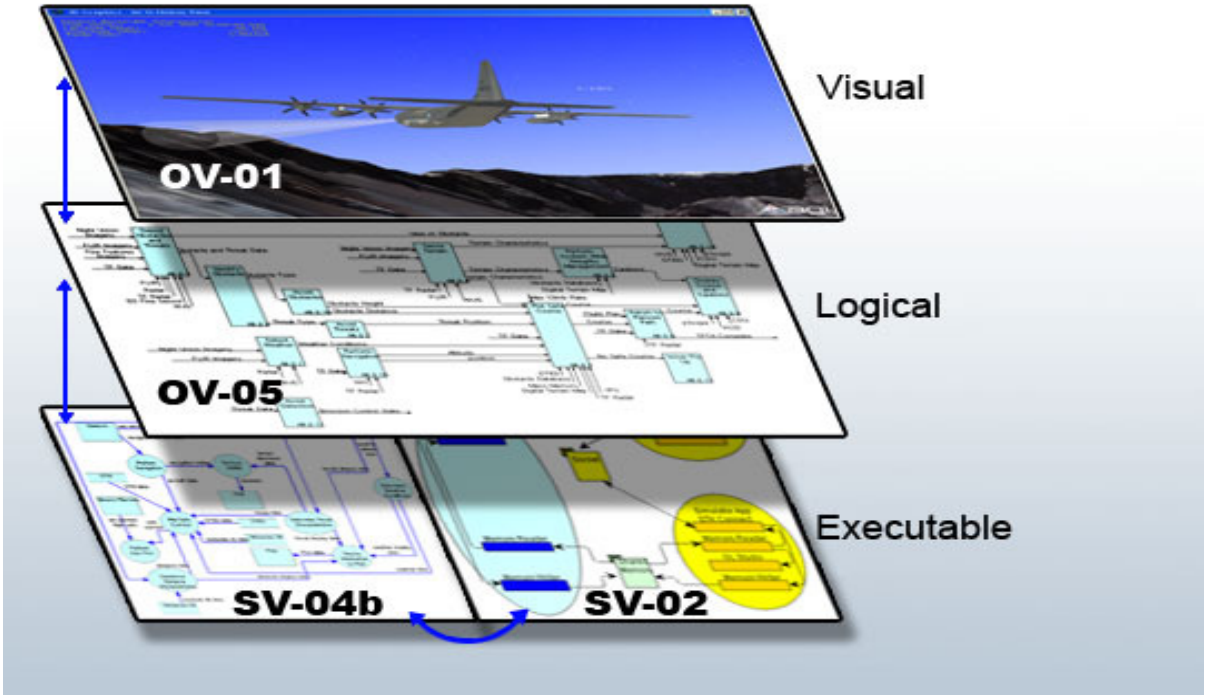

Figure 1 Visual, logical, and execution views of simulation environment

Figure 1 illustrates three integrated views of a simulation environment; a visual, logical and executable view stacked on top of each other. The views are snapshots of the same simulation from different perspectives. The top picture depicts the air vehicle in its operating environment. Other components of the visual view may be cockpit displays and simulation monitors. The middle layer is a logical view of the entities and their relationships. During the course of a design mapping project, the logical view will evolve. Early logical views may primarily be behavioral requirements models which then evolve to a design model. The executable view describes how the simulations are interfaced and partitioned to run in a distributed computing environment.

1.2 Using Simulation within the Design Process

design tasks, such as evaluation of flying characteristics with high-fidelity pilot-in-the-loop simulations. These simulation tasks occur late in the design cycle normally after initial system design and require a high level of fidelity not available in the early design period [3]. However, integration of design modeling with simulation for requirements development and verification after design is less common. The Pilot not only integrates design modeling with requirements analysis and verification, but enables fidelity to be increased as required without changing the simulation framework. The result is a flexible architecture that supports a wide range of models with their associated fidelities.

The accuracy of simulation results is a function of fidelity of the model used to obtain the results [4]. Models have fidelity provided they represent the

concepts and relationships at the level of abstraction needed. With this integrated simulation architecture, low fidelity models, can in certain situations, provide sensitivity results relevant to design optimization [5]. As higher fidelity models become available, the design can mature without the need for the same scope of trade studies. This reduces the opportunity for design errors to crop up later in the design process. Many of the operational simulation toolsets tend to be highly optimized for performance (in order to support high fidelity real-time simulations) and can be tailored to support large numbers of runs necessary for statistical analysis of trade study results. For the Pilot timing latency played a critical role, so we used a shared simulation clock to synchronize the diverse simulations. The ability to control the speed of the clock allowed us to obtain the level of fidelity needed for specific kinds of analyses.

1.3 Analysis

Traditionally, prediction of behavior for the integrated system is difficult and carries significant risk until a prototype has been built and tested. Small changes in operation of a product's subsystems as it interacts with its environment may have significant impact on the product's successful performance [6]. Simulation of dynamic behavior can significantly reduce risk in understanding behavior. Simulation of operation of the real design can be used to ascertain whether the design can provide the functionality needed to perform a mission capability, to develop allocated component behavior and performance requirements, and to produce validated performance models for the design. The analyses of the simulations can provide insight into the parameters that have the most significant impact on the ability to perform a mission capability. The Pilot simulations provided an analysis of desired capability and existing constraints that allowed us to calculate accurately latency constraints on the ability to integrate sensor data required as input for the functionality to be implemented. The timing constraint analysis provided allocated budgets for a functional decomposition. Based on this analysis we examined the design candidates for implementing the desired capability.

1.4 Test in Simulation Context

Analysts are generally interested in how the implemented systems behaves for inputs (initial conditions) varying over specific ranges. For example, terrain, mission, and pilot responses are varied to understand the terrain following and obstacle avoidance implementation. Performance analysis of a system typically consists of predicting observable attribute values of the system at time points or intervals, or when some event occurs, e.g., when object detection occurs [7]. Observable (test) conditions are often expressed in terms of input-output relationships. Design analysis without simulation can provide estimates that can be used as inputs into performance (I/O) models and can include temporal analysis to make the results more realistic. However, because of concurrency and difficulty of modeling the physics of systems in a complex environment, it is often difficult to make accurate predictions. For example, the field of view of a trainable gun on a moving platform may be obstructed in ways that are hard to envision without geometric modeling that includes dynamics. When a design has been implemented, then real physical tests can be performed and data collected. Design simulations can be used in the same way as real physical tests, but much earlier in the design process. For operational simulation to provide quantitative results, conditions whose values can be measured or recorded must be established. These data results of simulation can be directly analyzed.

Evaluation conditions are often statistical in nature. The I/O relationships are not generally functional. Requirements are often stated in statistical terms, e.g., the accuracy of obstacle detection may be stated as a probability to be achieved. Detailed statistical analysis of I/O relationships is often required. The same process of iterating the collection of data from physical test results can be applied to operational simulations. Varying input parameters can be used to construct regression models that can then be analyzed with statistical methods to understand what parameters made the most significant contribution to the outcome of the simulation experiment. Parameters that might be varied include the power of the radar beam and the sweep pattern. The resulting data sets are used to construct a performance model approximation to the full simulation model using regression analysis (curve fitting). The performance model can then be used by statistical analysis tools to determine the contribution of input variables to the output result. For the Pilot we examined a requirement for computing a safe flight corridor expressed as a probability value. This value, from the allocated functional description for the capability to be implemented was calculated as the result of conditional probabilities of sensor detection and data integration.

1.5 How has modeling and simulation technology improved?

The maturity and cost effectiveness in integrating operational simulation with design simulation results from (1) the availability of physics-based effects

modeling (sensor, aircraft, atmosphere, etc.) for operation in simulation execution environments, (2) the ability to develop executable design models using design hiding and layering techniques which allow an evolutionary development of increasing fidelity models, and (3) the ability to efficiently integrate simulations through a variety of interfaces such as APIs, shared memory, and sockets. Finally, the ability to integrate design and simulation with statistical data analysis on the results of simulation experiments provides quantitative results with known pedigree.

The software tools used to develop and execute the models for this Pilot are common in most aerospace development environments. System Architect was used to develop and document both the integrated simulation environment and the Pilot application design process. Rhapsody was used to develop and execute the design models. STK and Vega Prime were used for simulation visualization and certain aspects of the sensor/radar simulation. Minitab was used for data analysis.

2. The Pilot Application

The Pilot design task added a terrain following and obstacle avoidance capability to an existing air vehicle production model. For the Pilot, the admissible design space is defined by the equipment to be added to the production model with details of how the equipment is to be integrated. The specifications for the production model aircraft were available, as well as specifications for candidate equipment to be added for the modification. The Pilot used simulation of the air vehicle within its operating environment to assist with mission analysis and to establish functional requirements.

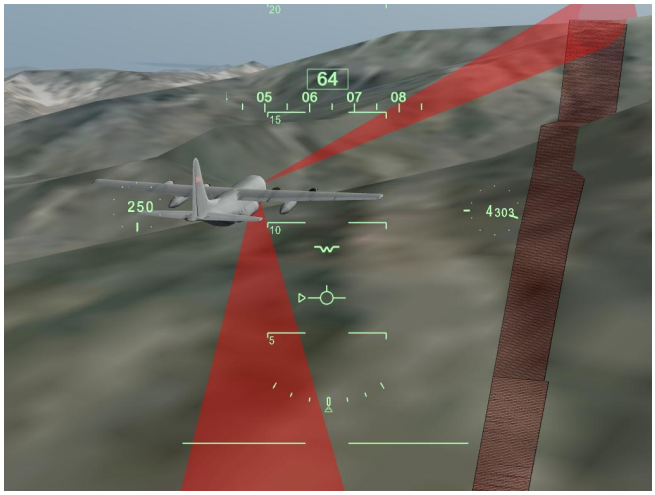

Figure 2 Snapshot of TFTA Scenario

Figure 2 shows a snapshot from the Terrain Following Terrain Avoidance Scenario. The green 250 to the left of the aircraft is the airspeed. The green 4303 is the altitude. The green 64 is the heading of the aircraft. The red cone that points down is the radar altimeter detecting the ground below the aircraft. The red cone pointing from the nose of the aircraft is the terrain detecting radar. The red shadow on the ground is the footprint of the terrain following radar to show its track along the terrain. As the simulation with the executable design models for the TF/TA proceeds, analysts can observe and record the aircraft changing altitude and pitch to avoid the terrain.

The operating environment includes the air vehicle with a pilot or autopilot flying under specific rules over specific kinds of terrain in specific atmospheric conditions. For terrain following and obstacle avoidance, a “safe corridor” function calculates where the aircraft should fly so as to avoid the obstacles based on sensor data integrated with known terrain information. Software that computes a safe corridor, parameterized with performance characteristics of the aircraft, is available. The computational results of the safe corridor function at a given time are the distance needed to obtain a safe altitude. Radar is used to detect obstacles that are not part of the known terrain. The operational loop of sensing data and modifying course is implemented as devices on avionics bus architecture.

The avionics system, which uses the 1553 bus architecture, is in operation and its properties are well understood. Even though the component systems are supplied with performance specifications or operational flight software, the actual performance in operational situations may be hard to predict. In this case, the latencies of moving data on the 1553 bus becomes a significant player because it impacts the time the airplane has to react to changing sensor inputs. Having the ability to determine the sensitivities of the different 1553 message rates as relating to sensor fusion, terrain databases, and flight control commands, allows a more accurate prediction of overall system performance. This in turn may provide the visibility to optimize on a less expensive solution than having to upgrade to a faster bus such as a fiber optic based solution.

For the Pilot, we the used a simulation execution environment to host and integrate both the operational and the design models. It provides all of the system-level services required to simulate an integrated Avionics and Vehicle Systems environment, and allows for distributed processing of all of the subsystem and vehicle system models for performance balancing. The actual distributed topology is self-discovering and may be varied as needed for the

particular experiment. Simulation models are used to represent design context and the candidate designs that are being explored. The design process replaces the requirements model with an iteration of design models which satisfy the properties of the requirements models.

3. Capability Analysis

The requested capability is to enable the aircraft to follow terrain and avoid obstacles while flying close to the ground. Analysis is needed to determine precise requirements that the design should satisfy. This section outlines some of the analysis. For the aircraft to maintain a safe course it must maintain a minimum safe distance from the terrain. A safe distance is dependent on a number of factors such as aircraft speed, altitude, climb rate, direction, terrain profile, and distance of aircraft from the terrain. The analysis of the minimum safe distance required understanding exactly how the aircraft performs for the avoidance navigation in different situations. The initial capability analysis consisted of (1) determining the operational situations, (2) analysis of the capability into functions, and (3) determination of timing intervals available for performing the functions to leave sufficient time for the aircraft to modify its course to maintain safe flight.

3.1 Operational Scenarios

An operational scenario consists of the aircraft operating in a region described by parameters such as aircraft speed, altitude, direction, terrain profile, distance of aircraft from terrain, etc. This description of the operational context of the mission is referred to as a situation. The situation characterizes the aircraft operating environment. Operational simulation was used to determine radar behavior and aircraft performance to determine inputs to the safe corridor function, and to validate the behavior of the aircraft under specified conditions, and to determine the minimum distance needed to climb to safe altitude. These simulations take into account climb rates of the aircraft model. The team used a dynamic model of the aircraft with perfect sensor detection of terrain to analyze radar detection ability. The results are input arguments for the safe corridor function. Varying terrain shape and aircraft speed provides validation information for a safe corridor function. At a more detailed level, the weight of the aircraft when the climb begins, as well as, altitude and atmospheric conditions are all used to determine safe corridor inputs. We used simulation to determine the affect of mountains blocking the radar view of what is behind the mountains, which is something hard to predict without simulation or actual flight evidence.

3.2 Functional Analysis

From the functional analysis of the capability, we determine that the aircraft can plot a safe course provided it can (1) sense terrain obstacles, (2) has time integrate data about terrain with other sensed information regarding the aircraft's position, speed, direction, and weather, (3) compute the course change, and then (4) has time, climb rate, necessary to achieve a safe course.

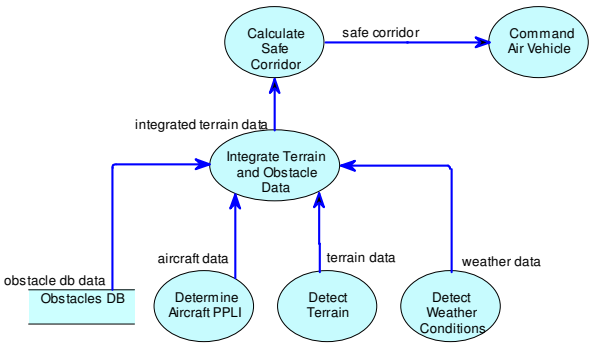

Figure 3 Functional Decomposition of Capability

Analysis of the desired capability led to a functional decomposition into six primary functions. These functions are displayed as ovals in Figure 3. The functions require input from other functions and from databases which are also described in Figure 3. These functions can be directly allocated to existing or candidate hardware and software components of the aircraft system. The design task is to define the precise behavior and performance for the functions and verify that the candidate components can execute the detailed function descriptions within the performance constraints.

3.3 Aircraft Performance

From a fixed aircraft performance model (constant 250 knots airspeed, terrain following height of 500 feet, and a maximum climb rate based on 3 engine performance of 2000 feet per minute), our simulation indicates that in order to clear a 1355 foot terrain obstruction (above the aircraft altitude), the pull-up must occur at 2.6 nautical miles out. Looking at a similar scenario helps demonstrate the criticality of system timing constraints

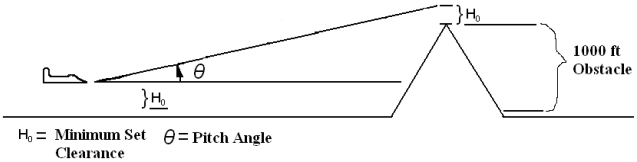

Figure 4 Aircraft Performance Model

and the value of being able to fully simulate the numerous aspects of terrain following (see figure 4).

At 250 knots with a terrain following altitude of 250 feet, the aircraft can maintain a 10 degree pitch angle and would require 1.07 statute miles to clear a 1000 foot obstacle. Iterative integrated simulation trials provide the analytical and statistical results necessary to not only evaluate the performance in this environment, but allow design permutation and optimizations as well.

3.4 Radar Analysis

While the radar to be used in this application has verified specifications such as the distance for which objects can be detected at various power levels, the actual variability of the radar's ability to detect objects is highly dependent on how specific parameters of the radar are set as well as the ability of the radar to scan regions as the aircraft is maneuvered. The Pilot objective was to determine how to optimize detection ability. The better the detection ability the more time that is available to determine inputs and make the calculations needed to calculate aircraft course changes necessary. A radar model with the correct sweeping behavior was used to analyze the terrain detection behavior. We found the behavior of the radar on the moving air vehicle platform was not entirely predictable from documented specifications for the radar. Radar sweep rate and power affect detection distance, as well as the shape of the actual terrain. Higher radiated power makes detecting the aircraft easier and decreases its survival rate.

This radar has an active and blended mode which affects the power and scanning distance of the radar. The active mode requires more power and can scan further out but it increases the probability of the aircraft being detected by enemies. The blended mode has a range of 3.8 nautical miles and a reduced probability of the aircraft being detected. The Pilot utilized the smaller detection range mode of blended. The radar can scan from -30 to 30 degrees Azimuth with respect to the nose of the aircraft. While the aircraft is flying a straight path, it only scans directly ahead.

The radar scans in an upward direction, stops when it detects horizon, turns around and moves back down to the bottom of the vertical scan, then scans upwards again. It does not scan while moving downward. Each scan is called a bar. While the aircraft is flying straight, it uses 2 to 3 bars. Three bars provide more data but it takes longer to scan. However, when the aircraft is turning it can scan up to 30 degrees in the direction of the turn. While turning, the mission computer increases the number of bars to scan ahead of the aircraft and in the direction of the turn.

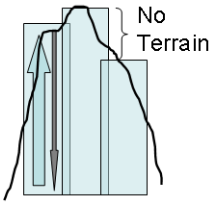

Figure 5 Radar Scan Behavior

In Figure 5, the simulation environment consists of a mountainous region with two ridges. The elevation scanning range is -30 to 10 degrees with respect to the horizon. There is an overlap between the scan bars. For the pilot, the overlap was set to 0. There are normally 2 scan bars per sweep when the aircraft is flying straight, but there can be up to 15 bars while turning. The scan beam moves up by 0.325 degrees as it scans a bar. The radar scans upwards at 60 degrees per second and moves downward at 85 degrees per second. Radar data is sent from the sensor to the computer after the completion of a bar. The sensor takes 0.25 seconds to turnaround on either end of the bars.

For terrain detection we used an experimental situation defined by terrain profiles, and weather, to determine radar detection performance. For example, a testable condition for the terrain following system is the elapsed time from obstacle detection until information regarding object location and type of obstacle are computed and fed into the safe corridor function. The following table describes aircraft performance constraints, radar specifications, and terrain configuration used in simulation experiments to determine terrain detection performance.

Table 1 - Configuration

Aircraft	
Max dive rate	-5.5 degrees
Max climb rate	610 meters per minute
Max bank angle	30 degrees
Max turn rate	3 degrees per second
Radar	
Max Range in Blended mode	3.8 nautical miles
Azimuth scanning range	-30 to +30 degrees with respect to aircraft
Elevation scanning range	-30 to + 10 degrees with respect to horizon
Beam Size	2.9 x 3.9 degrees (Az x El)
Overlap (Placeholder)	0 degrees Azimuth
Scanning pattern	2 – 15 bars
Beam step size	0.325 degrees Elevation
Environment	
Mountainous	2 ridges
Height	1000 feet

The aircraft's flight path is limited by the parameters in Table 1 and cannot be altered. The dive angle during descent cannot exceed -5.5 degrees. The climb rate varies with weight and temperature, but for the purposes of the pilot, it was set to a max value of 610 meters per minute. The max bank angle is the maximum angle of the aircraft during turns. The bank angle reduces the aircraft's ability to climb and is limited to 30 degrees. The max turn rate is limited by the ability to scan the area ahead of the aircraft in the direction of the turn without outpacing the scans. Table 2 describes a collection of simulation runs with the input parameters that were varied.

Table 2 - Input Parameters

Run	Aircraft	Radar
1	250 knots	2 bars, normal sweep pattern, normal sweep rates
2	250 knots	3 bars, normal sweep pattern, normal sweep rates
3	250 knots	2 bars, normal sweep pattern, double sweep rates
4	250 knots	2 bars, full sweep pattern, normal sweep rates
5	250 knots	2 bars, no pauses for the sensor to turn around, normal sweep rates
6	200 knots	2 bars, normal sweep pattern, normal sweep rates

Run 1 is the baseline run with every parameter set to nominal values. Run 2 modified the number of bars which increases the detail of the scan but increases the time to perform a full sweep. Run 3 doubled the scan rate in the upwards and downwards direction which decreased the time it takes to perform one bar and the time it takes the sensor to return to the bottom elevation. Run 4 scanned the full elevation range instead of stopping after detecting horizon, which results in scanning areas that do not need to be scanned, unnecessarily increasing the scanning time. Run 5, removed the turn around time which effectively decreased the amount of time the scanner is not scanning. Run 6 slowed the aircraft down to gives it more response time for avoiding obstacles. The pilot analyzed the detection distance for the five runs performed at 250 knots. The detection distance has a lognormal distribution that skewed towards the aircraft due to the radar continuously detecting the same terrain as the aircraft moved closer to the ridges in the scenario.

Only the full sweep has a different detection range. The full sweep pattern takes longer than the others which impacts the detection distance. This indicates that ending the sweep when the horizon is detected has an impact on the detection distance while the other variables had little or no impact. This is a useful feature to have for the radar since scanning past the terrain also increases the probability of detection.

The radar analysis and the aircraft performance analysis provided inputs to the safe corridor function. We validated the overall performance of the operational loop of sensing data and modifying course based on realistic radar performance and perfect ability to integrate sensed data with terrain data and provide results to the safe corridor function. In order to clear an obstacle, the radar needs to detect the obstacle far enough away to for the aircraft to have time to climb to the clearance altitude without exceeding the maximum climb rate. The larger the obstacle, the further out the aircraft needs to detect it.

The radar, after TF/TA modifications, will have a 2.07 statute mile look ahead scan. For this case, that means we will have one mile (at 250 knots) or 14.4 seconds to react once the radar has detected the obstacle. This does not, however, take into account that the sweep pattern timing can delay the detection by almost a second since the radar does not capture data as it moves from the end of one scan bar to the beginning of the next.

Since we only need 1.07 mile detection range, if we chose to lower the radar power such that the scan range was reduced to 1.2 miles (to lower the probability of detection), that means that we now have 0.13 miles, or 1.63 seconds before the pull-up must be established. Based the radar sweep performance, it might be roughly a second before the radar can detect the obstacle and therefore barely more than 0.6 seconds is available to move the radar data to the proper subsystems, process the information and propagate a pull-up command.

4. Requirements Analysis

Requirements analysis is used to precisely define the requirements and analyze how the requirements can be verified. In this section, we focus on the requirement that an aircraft has a safe course to fly for a given flight scenario. The previous analysis of the aircraft flying within various operational situations (e.g. terrain, obstacles, and weather), characterized features of the situations that could affect the ability of the radar to detect an obstacle and the ability of the aircraft to safely navigate to avoid the obstacles. A situation in which the aircraft must have a safe course is described by parameters such as aircraft speed, altitude, heading, terrain profile, distance of aircraft from terrain, etc. that characterize the aircraft operating environment. We

assume that the general terrain is known and that any unknown obstacle on the terrain has limited height above the terrain which can be avoided by flying over or around it as long as the obstacle can be detected at a sufficiently far distance. The probability of having a safe course depends on the time the aircraft takes to traverse the distance from the point the obstacle is detected to the obstacle being greater than the time it takes the mission computer to integrate various data plus the time it takes to calculate a safe course plus the time it takes to fly to the safe position above the terrain.

An aircraft has a safe course in a situation provided the avionics system can compute a new course that the aircraft can actually implement (e.g. within climb rate constraints) in the time available. The difference between detection time and time to change course is the time budget available to the avionics system to integrate the data, compute a safe course and command a course change. The result of the operational analysis is an allocated time budget for the avionics system to integrate the terrain data with other sensed information regarding the position, speed, and heading of the aircraft and weather conditions. For example, situation 1 (S1) is flown at a given speed of 250 knots, with a clearance altitude of 500 feet, flying level and with an obstacle height of 1000 feet. The distance to detect terrain obstacles with a probability of 99% is 4 nautical miles. The time budget available to integrate data and calculate a course change in this situation is t_b. The height of the obstacle above the height of the aircraft plus the clearance altitude is y, the horizontal climb distance is x_c, the speed of the aircraft is v, the distance the aircraft flies to the object is d, the time the aircraft takes to over fly the object is t, the horizontal detection distance is x_d, the max climb rate is c_m and the horizontal component of speed is v_h. See Figure 6.

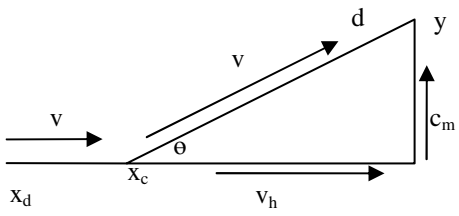

Figure 6 Geometry

The maximum time allowed for calculation depends on the speed of the aircraft, maximum climb rate and height of the obstacle.

$t = y / c_m$ $v_h = v * \cos(\theta)$ $\theta = \arcsin(c_m / v)$
$x_c = v * \cos(\theta) * t$ $x_c = v * \cos(\theta) * y / c_m$
$x_c = v * \cos(\arcsin(c_m / v)) * y / c_m$
$t_b = (x_d - x_c) / v$
$t_b = (x_d - v * \cos(\arcsin(c_m / v)) * y / c_m) / v$

The climb rate required to clear an obstacle is the height of the obstacle divided by the time it takes to clear the obstacle. The time it takes to clear the obstacle depends on the speed of the aircraft.
$d = (x_c^2 + y^2)^{0.5}$ $t = d / v$
Climb rate $= y / t = y * v / (x_c^2 + y^2)^{0.5}$

If the computed climb rate is less than the maximum climb rate, the aircraft will be able to clear the obstacle. The requirement for an aircraft having a safe course in a given situation, S1, can be specified in terms of a conditional probability.

probability (SafeCourse|S1) > 0.99

This statement is interpreted as the probability that a randomly chosen flight course is in SafeCourse given the situation S1 is greater than 99%. The probability(SafeCourse|S1) is the product:

probability(DetectObstacle|S1)*
probability(ComputeSafeCourseInTime|S1)*
probability
(SufficientTimeAvailableFlySafeCourse|S1).

However, the probabilities of DetectObstacle and ComputeSafeCourseInTime are close to 1 and so the verification of the probability of a safe course reduces to the probability of sufficient time is available to fly the safe course. From a functional analysis, we obtain that the aircraft has time to fly a safe course provided it

- has time to integrate data about terrain with other sensed information regarding the aircraft's position, speed, direction, and weather,
- has time to compute the course change, and
- has the time and climb rate necessary to achieve a safe course.

The time to achieve a safe course depends on the distance from the obstacle when it is detected, height of the obstacle, aircraft speed and climb rate.

At the current time, we are planning to run simulations to collect data regarding the time to achieve a safe course using radar detection distances with varying speeds and climb rates. Also, we are planning to record the success/failure of avoiding obstacles during the various situations.

5 Design Analyses

The design analysis is based on dynamic models of the avionics system. We built executable models of the 1553 bus and controller architecture. The bus controller model is initialized with the same data files as used in the aircraft model. The model uses a table of performance parameters to achieve the same statistical performance for data transfer as the actual bus. The design models used for the upgrade are developed using model-based design principles and are executed in the distributed simulation execution environment used for operational simulations such as the radar and the aircraft performance.

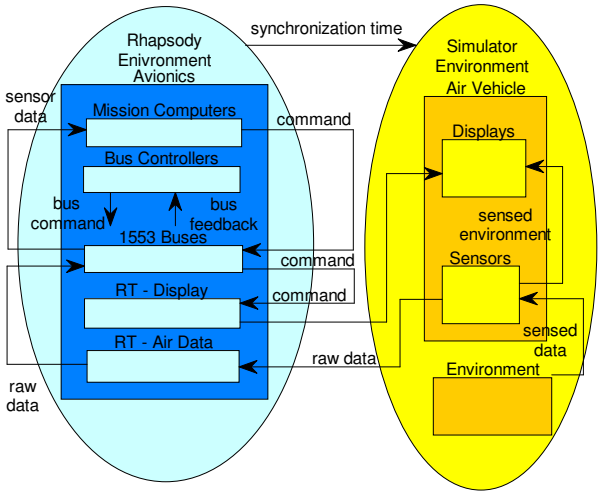

Figure 7 System View of Integrated Models

The major design question was whether the equipment could be integrated on the existing avionics bus to achieve the performance needed, or whether the performance could be achieved only by adding an auxiliary bus for data integration. While the behavior of the avionics controller and bus are deterministic, the time needed for computations, the behavior of a pilot, the air vehicle, and other entities in the environment are not deterministic. A critical aspect of the design analysis is determining whether the avionics architecture can integrate sensor data and provide the integrated data used as input to software that calculates where the air vehicle should fly. Analysis of the design alternatives requires understanding whether the 1553 bus architecture can provide the data transfer speed and volume needed to integrate sensor data with terrain data to be used to compute a safe corridor function.

Figure 7 provides a system view of the models that are part of the combined execution environment. The blue oval is the Rhapsody execution environment for the SysML models and the yellow oval is the Vega execution environment for the operational models. The arrows show the data flow between and among the models.

6. The Simulation Execution Architecture

The simulation architecture uses virtual shared memory for time, sharing data between applications and other global states. All of the executable models are clock driven with a simulation clock. The clock can be set at slower or faster than real time, and it can be stopped and restarted. All of the simulations operate using simulation time. Bridge models are used to provide interfaces to both design model execution and simulation model execution.

Figure 8 is a SysML [8] design model for the avionics system. The behavior of this model is described by state charts. The model controls message flow between devices such as sensors and displays and specifies actual or desired performance such as message transmission times. The model is the bridge between models for the design of new functionality to be added to the avionics system, such as terrain following and obstacle avoidance, and a simulation environment that models the behavior and performance of operating aircraft.

In Figure 8, the "Simulation Control" element controls the graceful start-up and user control of the entire simulation, essentially modeling the power-on sequence of the aircraft's avionics and vehicle systems as well as injection of power-cycling or simulated software restart events. The "Simulation Clock" element is the master time source for every participating executable in the simulation environment. The clock can be set slower or faster than real time, and it can be paused and restarted. A critical capability in such a simulation environment is that the data transfers across simulated networks have performance characteristics that match the actual hardware being simulated. The simulation clock therefore waits until all of the executables in the distributed environment have completed performing the tasks that the real hardware would have completed in a given time slice before it advances the simulation time. This forces synchronization across the entire simulation, in a manner that guarantees accurate fidelity of data flow, with a tradeoff between time slice size and real-time performance. This tradeoff can be managed by distributing the simulation elements across more or faster processors. The simulation architecture uses virtual shared memory for time, sharing data between applications and other global states. All data logging occurs at simulation time in order to factor out any simulated vs. real time differences. The "Bus A"

and "Bus B" are instances of the generic 1553B Bus model described in section 6. Each bus model communicates with the Simulation Clock as well as with all of the subsystem models which are virtually attached to that bus.

Several example Avionics subsystem models are shown on the diagram, along with "Shared Memory" and Vehicle Systems model placeholders. The "Fuel Sensor" and "Fuel Gauge" models are attached to Bus B, while the "Radar Sensor" model is attached to Bus A. The "Fuel Supply" model is not attached to either bus. It provides data to the Fuel Sensor model through shared memory, using the same services as the entire Vehicle Systems suite of simulation models.

With the executable avionics model we are now in a position to do the simulation experiments to determine the available time for integrating data and computing the change course.

7. The Bus Architecture Model

The 1553B Bus model provides a simulated Mux which manages the transfer of data among models which themselves are simulations of various subsystems (see Figure 9). The subsystems supported by this architecture can be both avionics and vehicle systems models, allowing for any necessary subset of the entire air vehicle system to be simulated. The 1553B model supports varying degrees of fidelity, with corresponding performance implications, in order to allow for engineers to study different aspects of candidate designs. The model is generic and reusable, with all communications passing through four ports.

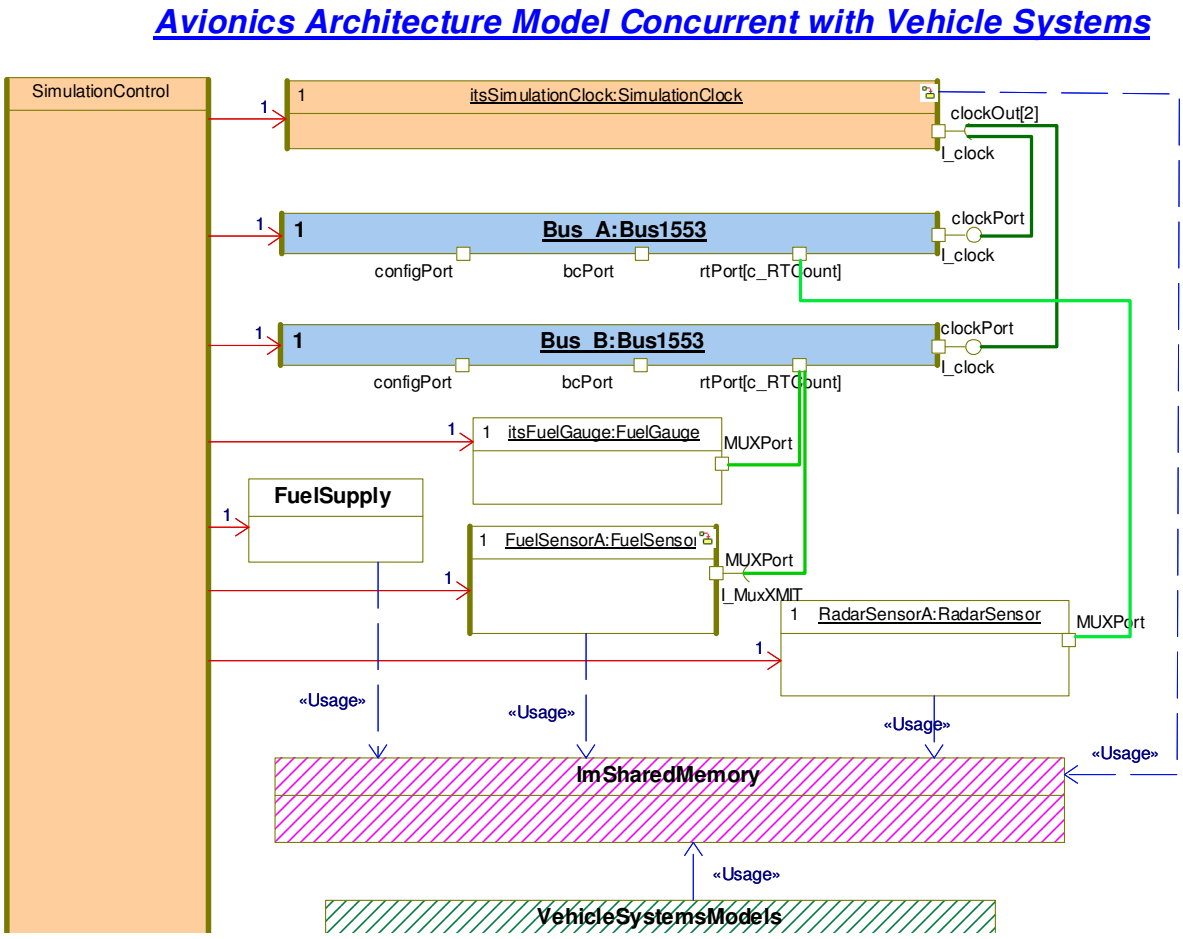

Figure 8 The Avionics Design Model

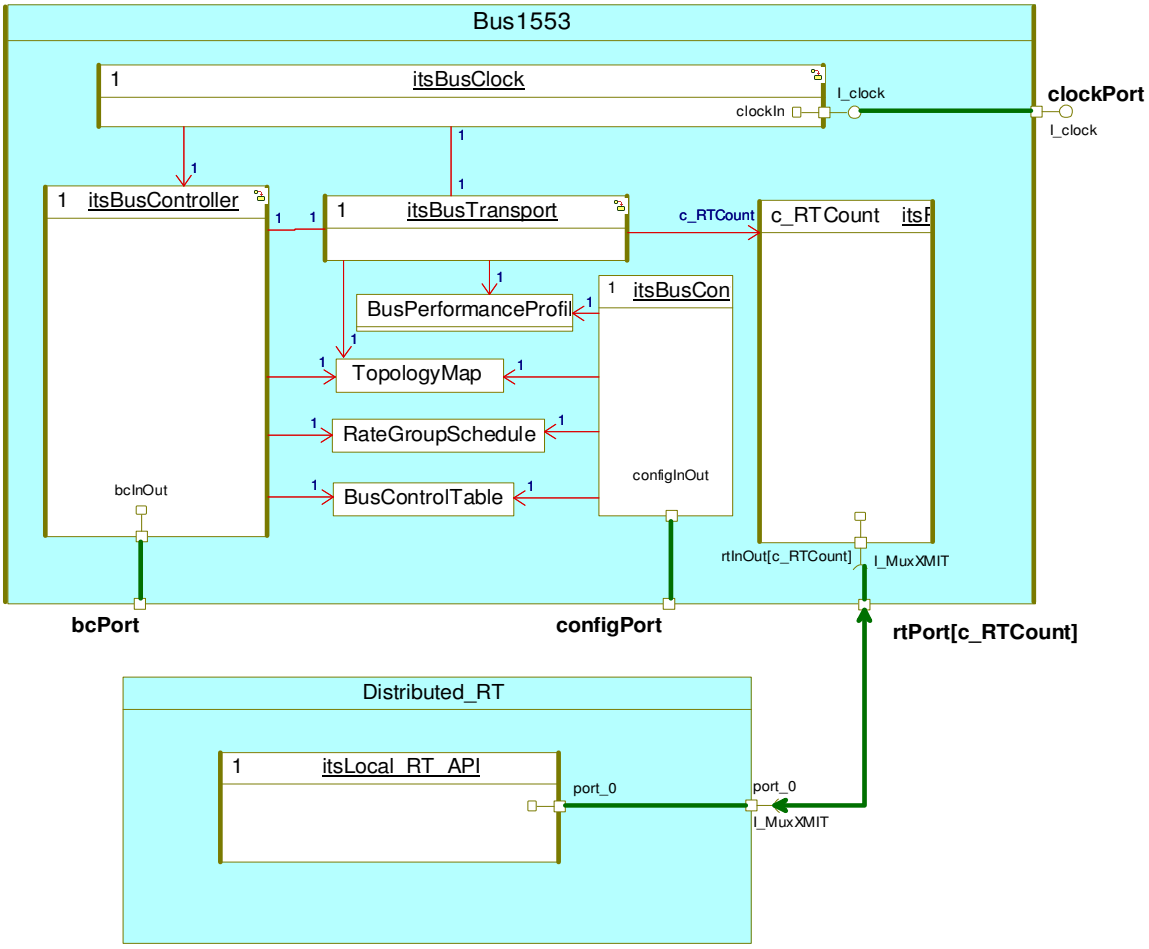

Figure 9 The Avionics Design Model

The "Config Port" provides for all tailoring and customization of the Mux behavior. The model is generic, and reuse is supported by providing the topology information, messaging schedules, rate groups and bus transport performance characteristics via this port. The "Clock Port" provides all time services into and out of the Mux model. This allows for some of the most powerful capabilities of the model. Simulation time is maintained rigorously such that data transfers across the modeled 1553B Muxes matches the behavior that would occur in an actual hardware environment. For each data transfer, the Mux model determines how much real time would have passed, and provides this information back to the overall simulation clock in order to pace all of the other models and keep everything in sync. Simulated time can therefore be paused or run faster or slower than real time, within the limits of the processors used for the experiment. The "BC Port" provides for simulation control of the Bus Control functions for the modeled Mux. This allows for studying several aspects of the air vehicle dynamically that are difficult to analyze in a static environment. For example, the model supports the simulation of backup bus control switchovers, including messages lost during a switchover and the behavior of a backup bus controller during startup. Also, the model supports failure injection, allowing for simulation with various fail/retry schemes to study their effects on the simulated system behaviors. The "RT Port" is a set of ports which provide the modeled attachment points for all of the subsystem models. Each RT supports send/receive services, time synchronization, the ability to be distributed across a set of processors, communications with the modeled Mux via TCP/IP port/socket services, and data logging for offline analysis post-test. Data can be transferred to subsystem models directly via compiled RT code or indirectly through shared memory structures without modifying the source code of the subsystem models. This allows for great flexibility in reuse of legacy models.

8. Conclusion

While at this point in time not all of the simulation trials have been run and the data collected and analyzed, we believe that the pilot successfully demonstrates that the capability to integrate design models with physics based operational simulations is entirely feasible with the current maturity of modeling and simulation-based product design tools. The results of simulation exercises are fully incorporated into the design process, and the data has a documented pedigree. A description of a simulation trial includes the air vehicle configuration, the operational environment description, the fidelity of all of the models used, as well as, the actual data recorded as part of the simulation trial. A description of a simulation exercise includes multiple trials where parameters are varied over specified ranges. The results have allowed us to obtain much better estimates of the avionics performance under operational conditions than could have been done without flying the actual aircraft. The effort involved the integration of such toolsets into a common operating infrastructure, along with the development of appropriate fidelity models. The resulting infrastructure is reusable in a variety of contexts, and will be used for other design trades for this aircraft model, and may be used for designing a new generation avionics architecture.

8. References

[1] Johnson, T., Paredis. C., and Burkhart, R. Integrating Models and Simulations of Continuous Dynamics into SysML. Modelica Conference, 2008.

[2] Haley, T., Friedenthal, S., Assessing the application of SysML to systems of systems simulations, Proceedings of the Spring Simulation Interoperability Workshop, September, 2008.

[3] Steinhauser, R., Looye, G., and Brieger, O., Design and Evaluation of Control Laws for the X-31A with Reduced Vertical Tail. Proceedings of the AIAA Guidance and Control Conference, Providence, Rhode Island, August 2004.

[4] Sargent, R., Verification and validation of simulation models. Proceedings of the Winter Simulation Interoperability Workshop, 2005.

[5] A. Kossiakoff, and W. Sweet. Systems Engineering: Principles and Practice. Hoboken: John Wiley & Sons, Inc., 2003.

[6] C.Wasson, System Analysis, Design, and Development – Concepts, Principles, and Practices. Hoboken: John Wiley & Sons, Inc., 2006.

[7] Buede, Dennis M. The Engineering Design of Systems: Models and Methods. Hoboken: John Wiley & Sons, Inc., 2000.

[8] Friedenthal, S., Moore, A., and Steiner. F., OMG Systems Modeling Language (OMG SysML™) Tutorial, INCOSE Intl. Symp, 2006.

Author Biographies

HENSON GRAVES is LM Senior Fellow and the project lead.

STEPHAN GUEST is a LM Fellow in advanced visualization and the operational lead for simulation activities.

JEFF VERMETTE is a Principal Systems Engineer and a Mission Systems and Avionics Senior Architect.

YVONNE BIJAN is a Certified Enterprise Architect and the integrated architect and analyst for the Pilot.

HAROLD BANKS specializes in flight safety simulations.

GREG WHITEHEAD is a Certified Enterprise Architect.

BILL ISON, PhD is a Certified Enterprise Architect.