DESIGN OF FLIGHT GUIDANCE AND CONTROL SYSTEMS USING EXPLAINABLE AI

Lance Sherry, Center for Air Transportation Systems Research @ George Mason Univ., Fairfax, Va.
James Baldo, Data Analytics Engineering, George Mason University, Fairfax, Va.
Brett Berlin, Data Analytics Engineering, George Mason University, Fairfax, Va.

Abstract

Advances in Artificial Intelligence (AI) have enabled the development of Flight Guidance and Control Systems (FG&CS) using Machine Learning (ML) from large archives of operational flight data. As the scope of the missions of drones/Small Unmanned Airborne Systems (sUAS) and Urban Air Mobility (UAM) vehicles increase (e.g. operations beyond-visual-line-of-sight (BVLOS) over people), the need for proof-of-compliance for safety regulations is required.

Approaches to meet the regulatory airworthiness requirements for software assurance, such as those embodied in DO-178, require coverage of paths through the code. This "path coverage requirement" can be achieved by a sub-class of ML, known as Explainable ML (X-ML) that enables the derivation of formal models, such as Situation-Goal-Behavior Models (SGBM), from the ML FG&CS.

This paper describes the results of a case-study of the development of an X-ML FG&CS that highlights the limits of an X-ML only approach and the need for a Model-based System Engineering (MBSE) analysis and design to complete the design to meet safety performance standards. The case study identified, specifically, the presence of five design error archetypes. These design error archetypes can be eliminated (i.e. addressed) by analysis of the SGBM using Model-Based System Engineering (MBSE) Tools as described in this paper. The implications and limitations of this approach for developing airworthy FG&CS are discussed.

I Introduction

Flight Guidance and Control Systems (FG&CS) coordinate the real-time, closed-loop control of propulsion and flight control surfaces to achieve the desired mission trajectory specified by 4-D flight plan waypoints. These FG&CS are critical to the safe operations of drones, Small Unmanned Airborne Systems (sUAS), and Urban Air Mobility (UAM) vehicles as their mission scope increases to include operations over people, operations beyond-visual-line-of-sight (BVLOS), and increasing levels of autonomy.

Traditional methods for developing FG&CS are through a manual process of requirements, design, coding, and testing. The airworthiness safety of these systems is assured through a regulatory process. One way to demonstrate proof-of-compliance is through DO-178C - Software Considerations in Airborne Systems and Equipment Certification - an approved process for certification of commercial software-based aerospace systems by the FAA, EASA, and Transport Canada [1] [2].

The DO-178C approach assigns a Design Assurance Level (DAL) to the system based on a safety assessment/hazard analysis by examining the effects of a failure condition in the system: A – Catastrophic, B – Hazardous, C- Major, D- Minor, E - No Effect. The higher the DAL, the more testing objectives must be fulfilled (e.g. algorithm path coverage), and the greater the independence of testing needs to be (e.g. coders cannot test).

One way to satisfy the requirements for DO-178C is to use formal models for the specification of the functional behavior of the FG&CS. Formal models have the advantage that they can be evaluated by model analysis, evaluated by simulation, and can be used to generate code automatically, and generate test-cases. Formal model analysis is known as "verification-by-design" (as opposed to verification-by-testing) and enables early resolution of requirements and design issues. Automated generation of code and test-cases can reduce the time and cost of the system development life-cycle. One example of a formal model compatible with DO-178 requirements is the Situation-Goal-Behavior Model (SGBM) [3] [4].

Advances in Machine Learning (ML) enable the automation of portions of the requirements, design, and coding phases of the development life-cycle by generating an ML FG&CS by processing massive quantities of operational flight data. Most ML algorithms yield "black box" functions that do not provide functional behavior transparency to perform "path coverage." However, the resulting ML FG&CS algorithms must demonstrate proof-of-compliance for safety assurance either through path coverage or via other techniques such as performance-based risk (e.g. refs). Sherry et.al., [5] developed an Explainable – ML algorithm that can be represented as a formal model, such as the SGBM, that can be used for path coverage analysis for software assurance.

A case study developed an X-ML FG&CS for a fixed-wing vehicle from revenue-service and simulated flight data. When the X-ML derived SGBM was generated, the path coverage analysis identified "gaps" in the functional design. Categorization of these gaps identified five design error archetypes. These design error archetypes can be eliminated (i.e. addressed) by leveraging the formal analysis of the SGBM using a Model-Based System Engineering (MBSE) Tools as described in this paper.

This paper is organized as follows: Section 2 provides an overview of FG&CS. Section 3 discusses how X-ML is used for FG&CS development. Section 4 identifies the five Design Error Archetypes resulting from a case-study. Section 5 describes how these can be resolved using MBSE Tools. Section 6 provides a discussion of the implications X-ML FG&CS, and the limitations of this development process.

II Overview of Flight Guidance and Control Systems (FG&CS)

Flight Guidance and Control Systems (FG&CS) generate commands to the vehicle control surfaces and propulsion systems to guide the vehicle to the desired 4-D flight plan trajectory. The commands are based on the state of the environment, human operator instructions, the state of the vehicle, and the state of the vehicle systems (Figure 1). In a fixed-wing aircraft, the FG&CS commands aileron, rudder, elevator, and thrust to maintain the desired trajectory. The desired trajectory is generated by a separate Mission Planning Function.

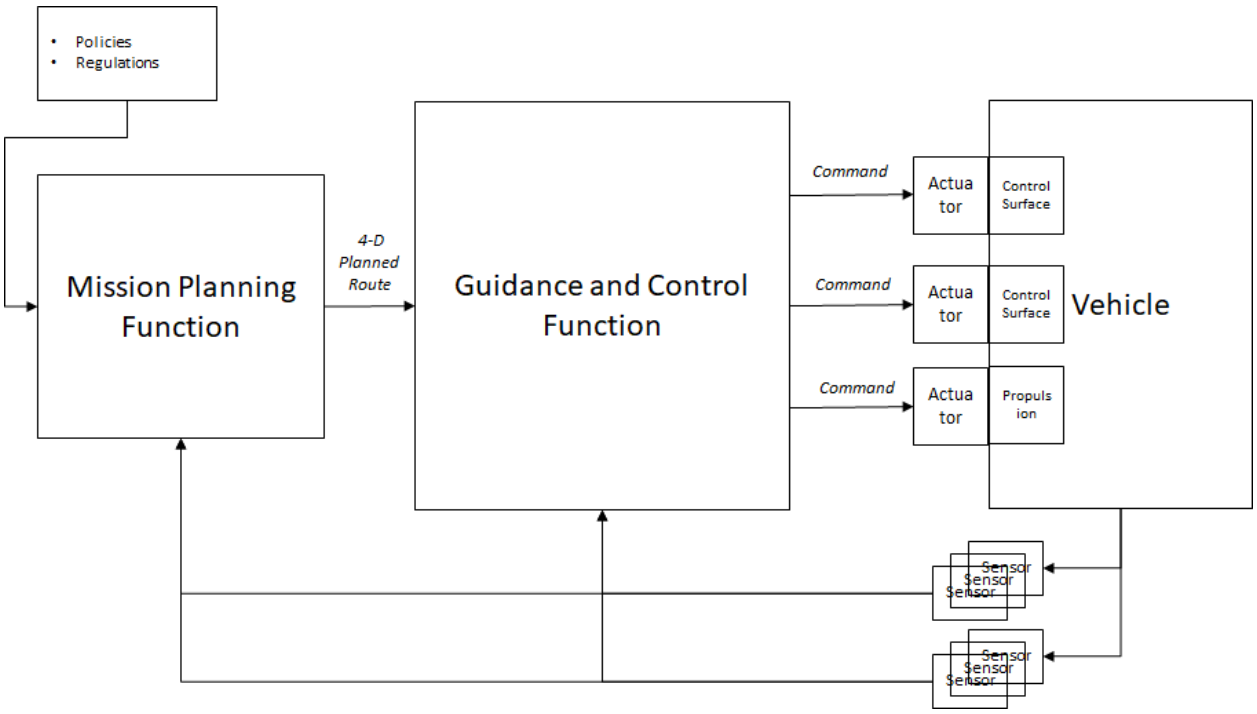

FIGURE 1: Generic Flight Guidance and Control architecture for vehicle.

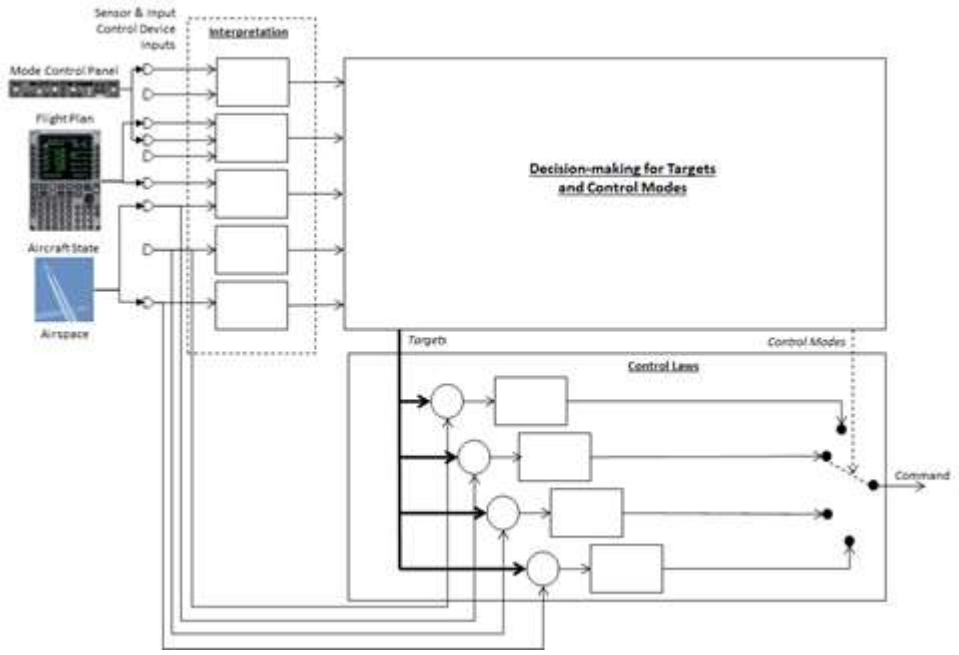

FIGURE 2: Canonical architecture for Guidance & Control Function.

Sherry [6] demonstrated that the behavior of the FG&CS for modern Flight Management System (FMS) and Autopilots/Autothrottles can be isomporphically mapped into the canonical architecture shown in Figure 2. The FG&CS commands to the control surfaces and propulsion system are generated by three functions: (1) Control Laws, (2) Decision-making for Targets and Control Modes, and (iii) Interpretation of input data.

(1) Control Laws

The FG&CS output commands are generated by closed-loop control laws that continuously adjust the commands to achieve a specified target (e.g. speed, altitude, rate of climb/descent, heading, course). A typical fixed-wing FG&CS will have multiple control laws for the same command. For example, a pitch command will have control laws for speed-on-pitch climb, altitude capture, altitude hold, fixed rate-of-climb, etc. Each control law is "tuned" for the specific trajectory maneuver (e.g. capture, hold). The FG&CS avionics for a typical fixed-wing aircraft will have 3 commands (pitch, thrust, roll) that are generated by approximately 10 different control laws.

These control laws are based exclusively on the aerodynamic characteristics of the vehicle using well-established control theory. Although Machine Learning control laws have been successfully demonstrated, the continuous closed-loop nature of the behavior lends itself to control-theoretic approaches.

(2) Decision-making for Targets and Control Modes

The appropriate Targets for each control law (e.g. altitude, speed, thrust, course) and the selection of the appropriate control law (also known as the Control Mode) are determined by a set of decision rules (Figure 2). The decision rules cover all combinations of "states" of the environment, pilot instructions through the user-interface, vehicle state, and vehicle systems state.

The Target and Control Mode selection "logic" can exhibit significant behavioral complexity. In addition to the aerodynamic properties of the vehicle, the logic must take into account the relative position of the vehicle to the flight plan, the active segment trajectory, the next segment trajectory, energy management, and speed envelope, airspace restrictions not already in the flight plan, optimum performance, and other mission-related factors.

The FG&CS avionics for the vertical navigation for a typical fixed wing aircraft will have 6 targets (path, altitude, speed, vertical speed, thrust, heading/course), based on over 350 rules [6]. The activation of one of the 350 rules is based on over 150 inputs with an average of 3 states each.

(3) Interpretation of Input Data

The Inputs to Decision-making for the Targets and Control Modes function are discrete "states." The states are defined by Boolean inputs (e.g. landing gear up, landing gear down), or by inputs with more than two discrete states (e.g. vehicle phase of flight: taxi, takeoff, climb, cruise, …). States are also defined from continuous data (e.g. altitude) by creating discrete states from ranges of values for continuous inputs (e.g. altitude above or below capture region, altitude in capture region, altitude in hold region).

The FG&CF avionics for a typical commercial aircraft will have over 150 inputs with an average of 3 states each.

Approximately 90% of the *functional behavior* of the FG&CS for a fixed-wing aircraft is located in the Decision-making for Targets and Control Modes Function. The Control Laws accounted for less than 10% of the behavior [3],[4],[6]. During the development phase, over 80% of the Problem Reports from simulator testing and flight testing are typically associated with the Decision-making for Targets and Control Modes.

These findings are not surprising. The design of the control laws is based on complex aerodynamics that is linearized and modeled by continuous mathematics. The design of the Decision-making for the Targets and Control Modes, however, requires logical decision-making based in a high-dimensional hyperspace defined by a large number of external factors including: the relative position of the vehicle to the flight plan, the active segment trajectory, the next segment trajectory, energy management, and speed envelope, airspace restrictions not already in the flight plan, optimum performance, and other mission-related factors. In this way, the improvements in development time, cost, and quality can be generated by ML development for the Decision-making for the Targets and Control Modes Function.

III Explainable-Machine Learning (X-ML) for FG&CS

Artificial Intelligence (AI) is the broader concept of machines being able to carry out tasks in a way that we would consider "smart". Explainable-AI (X-AI) are smart machines that can explain what they are doing and why. By this definition, FG&CS is AI.

Machine Learning (ML) is an approach to creating AI systems. ML uses massive sets of data to learn the underlying "algorithm" that can then be implemented in the AI system. Many ML algorithms are "black box" and cannot explain their behavior. Explainable ML (X-ML) is a sub-class of ML in which the ML algorithm can describe what it is doing and why. When the FG&CS software is generated by processing archived operational data it is X-ML.

An X-ML function capable of generating an SGBM from a massive set of operational data has been developed [5]. The X-ML SGBM is an Explainable Machine Learning (X-ML) algorithm for generating functions dominated by decision-making such as an FG&CS from operational data. The Situation-Goal-Behavior Model (SGBM) captures the stimulus-response behavior of the operational domain by defining the operational situations, the operational goal (or intention) of the response to the situation, and the corresponding behavior. The SGBM has several advantageous features. It has a semantic representation that is layered on a formal model. The semantic representation is useful for communicating with operators and mission experts to ensure operational compatibility. The formal model is used for the analysis of the design (i.e. model checking), simulation, and for automatic code and/or test generation.

The SGBM can be visualized in a tabular form known as the SGB Table (Figure 3). The SGB Table has three sections: Goal, Situation, and Behavior. Each section has a natural language description in the terminology used by mission experts.

The Situation and Behavior sections also have formal definitions. Each Situation is defined by a unique combination of Input States using an AND/OR format. Input States represent discrete

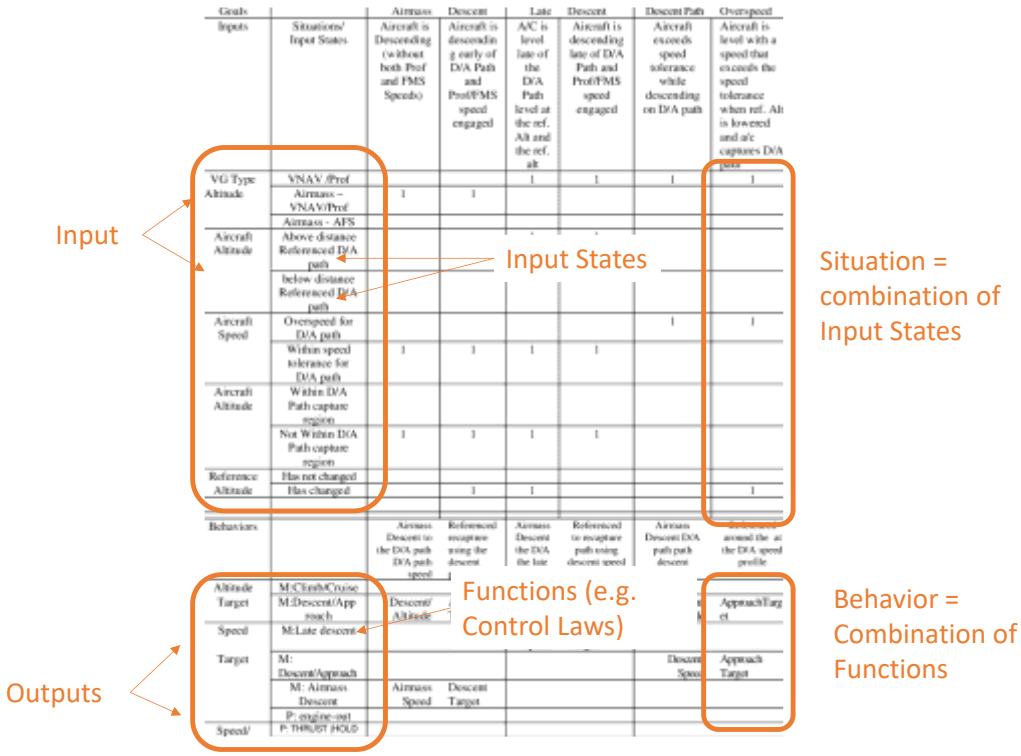

FIGURE 3: Situation-Goal-Behavior (SGB) Model visualized in an SGB Table.

states of the Input (e.g. Boolean True, False). For continuous inputs, the states represent ranges of values based on operational thresholds. For example, for an aircraft, the states of an Altitude Input can be below the Thrust Reduction Altitude, Above the Thrust Reduction Altitude. When an Input is not needed to define an operational Situation, known as a "Don't Care," all the states of an Input are considered "true" for the Situation. To avoid clutter, if a given input is a "don't care" the SGB Table displays all the States for the Input as blank.

Each Behavior is defined by a unique combination of closed-loop Controllers for each actuator command. The output is the actuator command. The closed-loop controller options for each Output are known as Functions. For example, Behaviors for a fixed-wing vehicle include the combination of closed-loop control functions across the pitch, roll, and thrust commands.

Verification-by-Design

The SGBM enables the functional behavior design to be verified by analysis before the function is implemented in software/hardware.

An SGB is considered "Complete" when every possible combination of Input States is included in the table and every possible combination of Input States has an associated Behavior.

An SGB is considered to be "Consistent" when a given combination of Input States appears only once.

This feature of the SGBM is critical, to achieve "airworthiness" certification for path coverage in DO-178.

Generating an X-ML SGBM

The SGBM compatible X-ML model is generated by for data analytic ML steps:

(1) identify the combinations of Functions for each Output (i.e. Behaviors) that exist in the data set.

(2) use the combinations of Functions for each Output (from $#1 above) to identify the ranges of states for Inputs derived from continuous variables (i.e. Interpretation)

(3) generate the functional behavior (i.e. decision-making) by establishing the combinations of Input States that occur in the operational data (i.e. the Situations)

(4) map the Situations (#3) to the Behaviors (#1)

IV Design Error Archetypes in X-ML FG&CS

X-ML SGBM FG&CS were developed for a fixed-wing vehicle using revenue service and simulator aircraft trajectories for a Vertical Navigation (VNAV) Function and an Airborne Collision Avoidance Systems (ACAS).

The Consistency and Completeness analysis on the SGBMs was conducted. These analyses identified "gaps" in the functional behavior. Specifically using the language of the SGBM, the X-ML SGBMs were

Table 1: X-ML SGBM Design Error Archetypes

SGB Design Error Archetype	Source of Design Error	Example	Solution
Missing Input	Signal absent from training/testing data set	Signal for Fly-by vs Fly-over for waypoint sequencing not available. Collision course for more than 3 vehicles not included	FTESS
Missing Input State	Signal is present in training/testing data set, *but* data set does not include full range of signal or change in behavior to identify the missing input state	Wind magnitude is present, but states do not include range to identify Windshear Crossing traffic position is present but states do not distinguish between straight crossing trajectory and turning trajectory	FTESS
Missing Situation (i.e. combination of Input States)	Signals are present, and full range of all signals are present, *but* combination of states resulting in unique behavior are not present	Missing situations due absence of combination of Input states. Examples: discrepancy between Radar Altimeter and Baro Altimeter, or for discrepant airspeed signals from separate sensors	MBSE Model Checking
Missing Situation to Behavior Mapping	Signals are present, full range of all signals are present, combination of states resulting in unique behavior are present, *but* unique behavior is not present	Situation (i.e. unique combination of Input States) is not assigned a Behavior	MBSE Model Checking
Missing Behavior (i.e. combination of Output Functions)	Combination of Output Functions is not present	Example, pitch and thrust combination for specific maneuver is missing (e.g. climb acceleration energy tradeoff between potential and kinetic)	FTESS

missing Situations and/or were missing mapping between Situations and Behaviors. The gaps could be categorized into five types of "design errors" (see Table 1).

Note: the design error archetypes reported here are the consequence of using *operational data* to develop X-ML SGBM FG&CS. They are not due to errors or limitations in the X-ML SGBM process or algorithm.

The Missing Input design error archetype is the result of the absence of a signal in the operational data.

The Missing Input State design error archetype and the Missing Behavior design error archetype are the result of the situation/behavior not occurring in the operational data set.

This Missing Situation and Missing Situation to Behavior Mapping is a short-coming in the operational data set.

V A Model-based System Engineering Environment for the Design of X-ML FG&CS

To address the five design error archetypes a Model-based System Engineering (MBSE) environment is proposed (Figure 4).

An X-ML SGBM FG&CS can be developed using the process outlined in Figure 4. First, the X-ML model is generated from archived operational data. Next, the model is converted into an SGBM. Once the X-ML SGBM is generated, the SGBM can be imported into the MBSE Tool. Using the MBSE tools visualization of SGBMs as SGB Tables, and the tool's model checking features, the functional behavior "gaps" in the X-ML SGBM can be closed. Finally, code and documentation can be generated.

Model Checking the SGBM

The SGBM enables the design of the functional behavior to be verified by analysis before the function is implemented in software/hardware. An SGB is considered "Complete" when every possible combination of Input States is included in the table and every possible combination of Input States has an associated Behavior. An SGB is considered to be "Consistent" when a given combination of Input States appears only once. The model checking is typically embedded in a Model-Based System Engineering (MBSE) tool that supports the SGBM

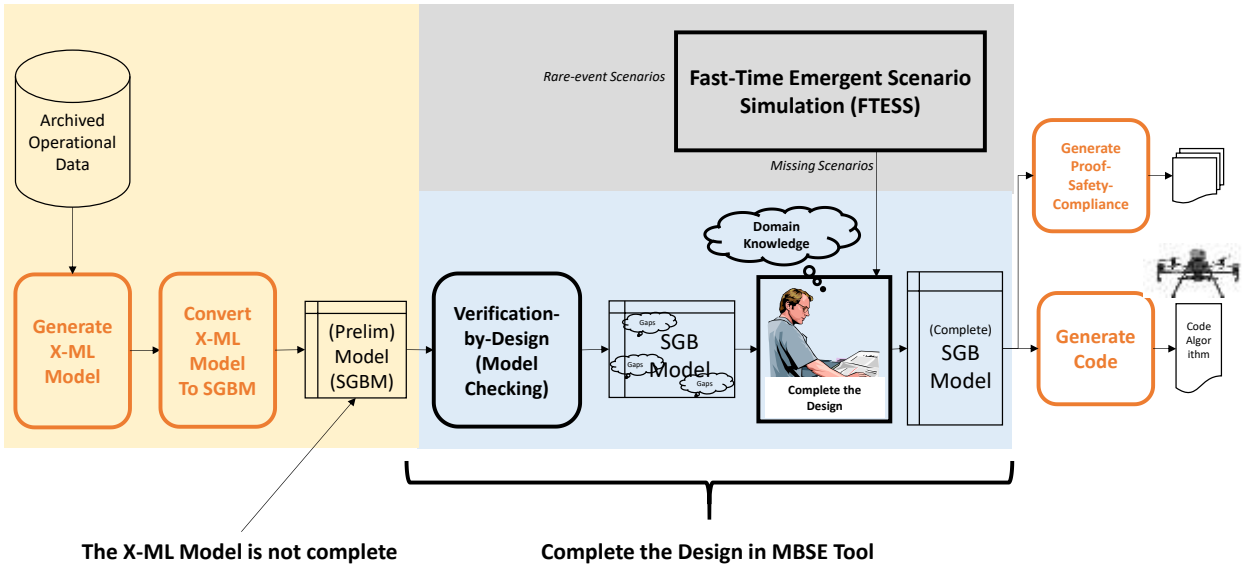

FIGURE 4: Process for development of Flight Guidance and Control Systems using Explainable ML to generate models that can be used by Model-based System Engineering (MBSE) for analysis, verification-b-design, and code generation

modeling language.

This model-checking feature of the SGBM, critical, to achieve "airworthiness" certification for path coverage for DO-178, provides the means to identify the design error archetypes: (1) missing situations and, (2) the missing situation-to-behavior mapping.

Fast-Time Emergent Scenario Simulation

The other three design error archetypes – (1) missing inputs, (2) missing input states, and (3) missing behaviors – are the result of "gaps" in the data set used for the X-ML training and testing. These can be mitigated through a Fast-time Emergent Scenario Simulation (FTESS).

When the environment in which the FG&CS must operate is complex and/or infinite, traditional engineering practices are limited by the ability of humans to imagine the situations and corresponding behavior. Likewise, when developing an X-ML algorithm, the ML must be exposed to all the scenarios with sufficient frequency that the X-ML can: (i) encode all plausible situation-behavior pairs, and (ii) encode subtle differences in situations that require completely different responses from the FG&CS.

When the data is collected from operations, this approach requires logging millions of miles of real-world flying to attempt to capture all plausible situation-behavior pairs. For example, if situations associated with fatal accidents are taken as the least likely to occur and these accidents occur with 1E-8 per mile, then the vehicles need to perform at least 1E-10 per mile to get exposed to these situation-behaviors (i.e. 100 times more miles).

The statistical "rule of 3" says that if N data points with a specified event are to be observed, then the 95% upper bound on the event estimate is 3/N. To achieve 3/N, that would require 3E-10 miles. If a manufacturer has 1E-7 miles flown to date, then they need to repeat the testing completed to date 3,000 times (with no fatalities).

This imagineering – finding the known-unknowns and unknown-unknowns – can be accomplished using a proposed "Fast-Time Emergent Scenario Simulation (FTESS)" Sherry, Shortle, Donohue, Donnelly [7] and Nanduri & Sherry [8]. The FTESS is an agent-based, rare-event simulation "digital-twin" of the system. FTESS can be used to expand the training data for the ML algorithms with a significant reduction in time. The FTESS starts with a "seed" scenario from the real-world and using Monte Carlo techniques on a super-computer generates variances on the seed scenario.

The FTESS leverages techniques for rare-event simulation, edge computing, and runs on a super-computer. There are two main approaches for improving the efficiency of rare-event simulations—importance sampling (IS) and splitting [9], 10], [11]. The idea of IS is to change the underlying sampling distribution so that rare events are more likely. The idea of splitting is to create separate copies of the simulation whenever the simulation gets ''close'' to the rare event of interest, effectively multiplying promising runs that are more likely to reach the rare event. Splitting is useful for systems that tend to take many incremental steps on the path to the rare event. IS is also useful for systems that tend to take a small number of ''catastrophic jumps'' to the rare event.

The FTESS is enabled by inexpensive access to cloud-based Super-computing, such as the GMU Argos Supercomputing Cluster, and agent-based modeling environments, such as MASON [12]. The FTESS can exhibit varying levels of model fidelity according to the development phase.

VI CONCLUSION

There is significant hype around AI for Flight Guidance and Control Systems (FG&CS). The promise of AI is to reduce the development time and improve the functionality of F&GCS when they are developed using Machine Learning (ML) techniques.

ML is an approach to creating AI F&GCS based on massive sets of data used to learn the underlying "behavior." Many ML algorithms are "black box" and cannot explain the encoded behavior. Explainable ML (X-ML) is a sub-class of ML in which the ML algorithm can describe what it is doing and why. X-ML SGBM is an X-ML method for generating a formal model (e.g. the Situation-Goal-Behavior Model (SGBM) from operational data. The SGBM is compatible with airworthiness certification regulations such as DO-178.

A recent demonstration of the development of an X-ML SGBM FG&CS illustrated the potential for

reduction in development time, however, analysis of the functional behavior of the X-ML SGBM FG&CS, identified that the design was not complete. Specifically, the FG&CS functional behavior was: (i) missing specific operational situations, and (ii) if the design included the operational situation it was missing the associated appropriate behavior.

These issues were not caused by the ML process. The absence of the operational situations was a consequence of limitations in the training/testing data. The missing operational situations were either rare events in the mission, or "adjacent" operational situations with different appropriate behaviors.

These gaps in the functional behavior design are known as "incomplete designs." Incomplete functional behavior designs are not a new issue. They exist in F&GCS developed using traditional, manual engineering methods as well.

Model-based System Engineering (MBSE) tools provide the means to model, visualize and simulate functional behavior. More importantly, MBSE model-checking tools provide the means to analyze functional behavior design to identify missing operational situations and missing mappings between operational situations and appropriate behavior.

This paper describes a process for leveraging the best of both worlds by developing F&GCS by marrying the X-ML SGBM with MBSE analysis tools (Figure 4). In this way, a preliminary functional design can be rapidly generated using X-ML. This "incomplete" functional design can then be "patched up" using the MBSE tools to identify and design the missing operational situations and missing operational situation to appropriate behavior mappings. Another advantage of this approach is that the MBSE model can be used to automatically generate code, and generate proof-of-compliance for existing airworthiness certification processes.

References

[1] RTCA DO-178C (2011) "Software Considerations in Airborne Systems and Equipment Certification," December 2011.

[2] Jacklin, S. (2012). Certification of Safety-Critical Software Under DO-178C and DO-278A. Infotech@Aerospace.

[3] Sherry, L. (1995) A formalism for the specification of operationally embedded reactive systems. In Proceedings of the International Council of System Engineering, St. Louis, Missouri.

[4] Feary, M.S. (2010) "A toolset for supporting iterative human — automation interaction in design", Tech. Rep. 20100012861, Mar. 2010.

[5] Sherry, L. J. Baldo, B. Berlin (2020) Explainable Machine Learning (X_ML) to Generate a Situation-Goal-Behavior Model (SGBM). CATSR Internal Report 2020-007.

[6] Sherry, L., M. Feary , P. Polson , R. Mumaw , E. Palmer , J. Powers , J. Rubino (2001) A Cognitive Engineering Analysis of the Vertical Navigation (VNAV) Function. NASA Technical Memorandum 2001-210915. NASA Ames Research

[7] Sherry, L. J. Shortle, G Donohue, O. Donnelly (2019) Fast-Time Emergent Scenario Simulation (FTESS). Report: Center for Air Transportation Systems Research at George Mason University. Report # CATSR 2019-11.

[8] Nanduri, A., L. Sherry (2016) Generating Flight Operations Quality Assurance (FOQA) Data from the X-Plane Simulation. In Proceedings 2016 Integrated Communications, Navigation, Surveillance (ICNS) Conference, Dulles, Va. April 19-21, 2016

[9] Shortle, J.F., and P. L'Ecuyer, "Introduction to Rare-Event Simulation," in Wiley Encyclopedia of Operations Research and Management Science, Hoboken, NJ, USA: John Wiley & Sons, Inc., 2011, p. eorms0006.

[10] Zare-Noghabi, A., & Shortle, J. (2017). Rare event simulation for potential wake encounters. 2017 Winter Simulation Conference (WSC), 2554– 2565. https://doi.org/10.1109/WSC.2017.8247983

[11] Snisarevska, O. L. Sherry, J. Shortle, G. Donohue (2018.) Balancing Throughput and Safety: An Autonomous Approach and Landing System, In Proceedings IEEE ICNS Conference 2018. April, 2018

[12] Calderon-Meza, G., L. Sherry (2010.) The Effects of Airline Adaptive Route Selection on NAS-wide Performance. American Institute of Aeronautics and Astronautics: 10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference Sept, 2010

Acknowledgements

Thank you for technical comments and suggestions from Michael Feary (NASA), Ali Raz, George Donohue, John Shortle, Hadi El-Amine, Syed Zaidi (George Mason University). Thank you also to GMU students Oleksandra Snisarevska, and the data analytics project student team. This research was conducted using internal research funding.

Email Addresses

lsherry@gmu.edu

2021 Integrated Communications Navigation and Surveillance (ICNS) Conference
April 20-23, 2021