Modeling and Verification of Aircraft Stability Controller

Divya Udayan J
Dept. of Internet & Multimedia
Konkuk University

ABSTRACT

Stability control system is one of the most critical systems inside an aircraft. Design verification and validation of such a system is very essential to reduce development cycles and cost of system development. This paper evaluates the possibility and effectiveness of SCADE software in the design verification of stability controller model of an aircraft. Dataflow and state machine can be integrated in the SCADE suite for the formal verification of temporal logics of the hardware system. This technique is effective in finding out violations of system invariants at an early stage of the design phase. Graphical simulations and system analysis demonstrate the efficiency of this approach.

Keywords

Stability control, Design verification and validation, Data flow, State machine, Scade suit, formal verification.

1. INTRODUCTION

Development of safety critical applications like stability controller for aircraft requires a strict interdisciplinary approach to ensure safety, since failures are often catastrophic [1,2,3]. The development phase involves a large number of stages and therefore reducing the time in each stage is essential. In classical design methods, the system is tested, only after the complete prototype is produced and hence correcting those errors is difficult and may sometimes carry on to the later stages of the development phase which may increase the design costs [4,5]. In order to overcome this problem, a graphical tool known as SCADE (Safety Critical Application Development Environment) [6] is used in our work. SCADE is a graphical environment which allows the hierarchical definition of each system components and the automatic code generation. SCADE software is available as SCADE SUIT and SCADE DISPLAY. SCADE SUIT covers all phases in software engineering process from the system specification, automatic code generation down to simulation and system testing.

In this paper, we have tried to evaluate the possibility and effectiveness of the SCADE software with an example of stability controller in the aircraft domain as the preliminary step. Such a simplification would expand its mission capability by enabling more personnel to successfully operate the plane and even enabling autonomous operation.

The remainder of the paper is structured as follows. Section 2 describes the aircraft stability controller. Section 3 presents the system design employed to obtain the verification of the controller. Section 4 discusses the implementation details. Finally, concluding remarks and future work is given in Section 5.

2. AIRCRAFT STABILITY CONTROLS

The aircraft stability controller addresses specific properties that are desirable for the airplane. The pilot should be able to control the roll rate rather than actuator position [7]. This will help the pilots who are unfamiliar with the aircrafts to still fly the vehicle based on maneuvering principles. Whenever an aircraft changes its flight altitude and position, it encounters a rotation about any one or more of the three axes, which are imaginary lines that pass through the center of mass of the aircraft. The science of air vehicle orientation and control in three dimensions is known as Flight dynamics[8,9,10]. The three critical flight dynamics parameters are the angle of rotation in three dimensions known as roll, pitch and yaw. Figure 1. shows the axes to control the stability of an aircraft.

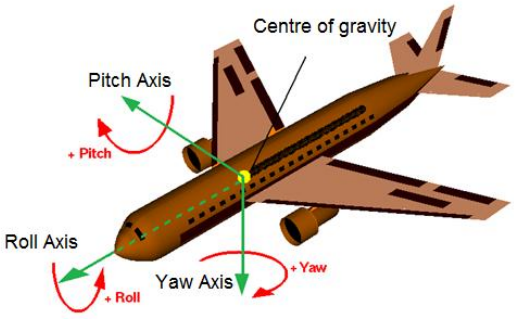
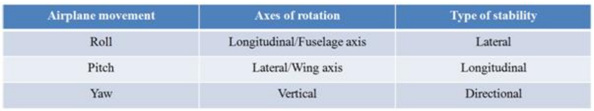


Fig 1: Axes to control the stability of an aircraft

Each axis will be perpendicular to the other two at the point where all the axes intersect. The axis which extends lengthwise through the fuselage from the nose to the tail, is the longitudinal axis. The axis, which extends crosswise from wingtip to wingtip is the lateral axis and which passes vertically through the center of mass, is the vertical axis.

Aircraft engineers develop stability control systems for the vehicle's orientation about its center of gravity. These control systems generate forces in various directions or moments about the center of gravity of the aircraft, and thus rotate the aircraft in pitch, roll or yaw [11,12]. If the airplane movement is roll and axes of rotation is longitudinal/fuselage axis, then lateral stability can be achieved. If the airplane movement is pitch and axes of rotation is lateral/wing axis, then longitudinal stability can be achieved. If the airplane movement is yaw and axes of rotation are vertical, then directional stability can be achieved. Table 1. shows the aircraft movement, axes of rotation and type of stability associated with each axis.

Table 1. Aircraft movement, axes of rotation and type of stability



Airplane movement	Axes of rotation	Type of stability
Roll	Longitudinal/Fuselage axis	Lateral
Pitch	Lateral/Wing axis	Longitudinal
Yaw	Vertical	Directional

3. SYSTEM DESIGN

In the design of safety critical systems like stability controller for aircraft, the tools and techniques are chosen in a manner to reduce the number of design faults that are introduced during post specification and also to protect the final software from their effects. One important characteristic of the software development process is that, in general, the earlier a fault is introduced, the more severe and expensive (in time and cost) is its effects [13]. Therefore mistakes in the requirement specification, modeling and planning phases are of most concern in managing the development of software. This fact leads to bridging the gap between system requirements and implementation by using visible traceability tool like SCADE in design verification and validation. The Figure 2 illustrates the proposed system design for the verification process.

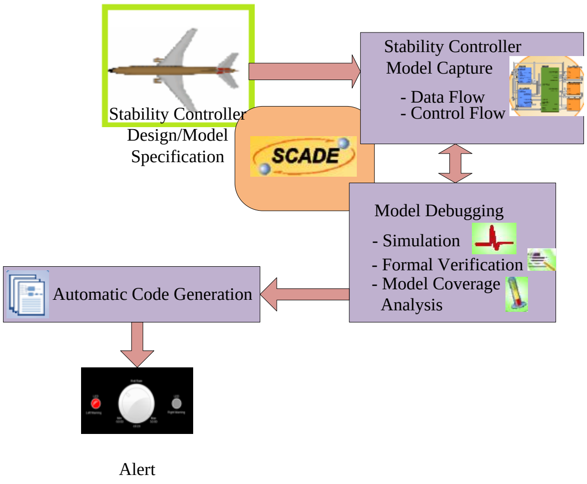


Fig 2: Proposed System Design

The stability controller requirements specification is framed as a result of a process of requirements elicitation which involves experts in avionics domain. The specification is based on an analysis of the safety of the aircraft and involves any other relevant information like details of earlier accidents. The requirements specification of the stability controller describes the functionality of the system including any protective measures and performance (in particular safety) criteria. In the model capture stage, the requirements specification is taken and the corresponding Data flow and Control Flow diagrams are formulated. The Data flow diagrams are graphical representation of the flow of data through each block of the system. They can be considered as an overview of the system which can later be elaborated. Thus a structured design can be made from this. The control flow diagram consists of sequential steps, with if-then-else conditions, repetition, and/or case conditions. Suitable geometric shapes are used to represent operations, data or equipment and arrows indicate the sequential flow from one state to another. In the next phase, Model debugging, we run the program to check whether the output is same as we expect. If a fault is identified, then the solution to rectify the fault can be made at an early stage. By simulation, we imitate the real roll rate and pitch of the aircraft [14,15,16]. We may also try different combinations of parameters to see if we can get the desired results. Thus we can provide eventual real effects of alternative conditions and provide a lifelike experience to the user. In this phase, formal verification is also done to check the correctness of the system. The verification of the controller is done by providing a formal proof on the abstract mathematical model of the system. One approach to do this is model checking which includes exploring all states and transitions in the model. Another approach is by using logical inference which consists of using proof objectives which evaluates some property to be proved as a Boolean condition. After model debugging automatic source code is generated using the SCADE KCG Code generator. The last stage is integrating the model design with the graphical panel, which is supported by SCADE Suit Rapid Prototyper for any alert generation. The implementation details of each stage are explained in the next section.

4. IMPLEMENTATION

Designing the stability controller with SCADE involves the following stages.

4.1 REQUIREMENT SPECIFICATION

The roll rate calculation subsystem calculates the plane roll rate, according to the joystick command and the adverse yaw coupling effects. Adverse Yaw function is given by the formulae

*rollCoupling = (leftAdverseYaw-rightAdverseYaw) * gain factor*

*rollRate = (joystickCommand-rollCoupling) * gain factor*

The absolute value of the aircraft roll rate is saturated to 25.0 i.e. Roll rate will never be less than -25 degrees/sec or greater than +25 degrees/sec. The roll rate warning subsystem computes left and right warning alarms which get activated, respectively if the aircraft roll rate is strictly less than -15 degrees/sec or strictly greater than 15 degrees/sec. The roll mode management subsystem computes the plane roll mode as either OFF, Nominal or Failsoft according to the ON/OFF button pressed and the aircraft roll rate value. The pitch is calculated from the variations in altitude and aircraft engine rpm.

4.2 MODEL CAPTURE

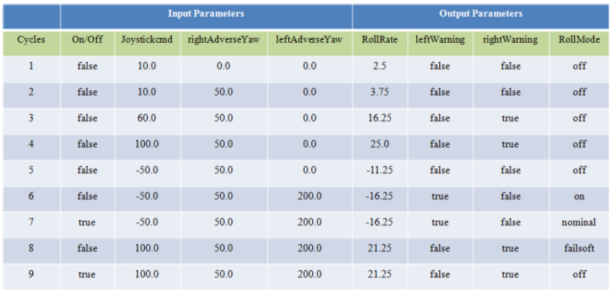
The model capture stage understands the workflow from the specifications of the model and capture them using modeling tools. The graphical formalisms used to design the stability controller are Data Flow and Control Flow diagrams. Safe State machines (SSM) [17] are used to represent the Control Flow diagrams. The detailed Control Flow is shown in the Figure 3. Then these graphical formalisms define the data structure of the model using data types and constants.

4.3 MODEL DEBUGGING

The third stage of the workflow is a three stage process.

(i) Semantic Check: The SCADE controller model is checked before simulation code or target code is generated. It is also possible to check model semantics at any time.

Table 2. Test cases



	Input Parameters				Output Parameters			
Cycles	On/Off	Joystickcmd	rightAdverseYaw	leftAdverseYaw	RollRate	leftWarning	rightWarning	RollMode
1	false	10.0	0.0	0.0	2.5	false	false	off
2	false	10.0	50.0	0.0	3.75	false	false	off
3	false	60.0	50.0	0.0	16.25	false	true	off
4	false	100.0	50.0	0.0	25.0	false	true	off
5	false	-50.0	50.0	0.0	-11.25	false	false	off
6	false	-50.0	50.0	200.0	-16.25	true	false	on
7	true	-50.0	50.0	200.0	-16.25	true	false	nominal
8	false	100.0	50.0	200.0	21.25	false	true	failsoft
9	true	100.0	50.0	200.0	21.25	false	true	off

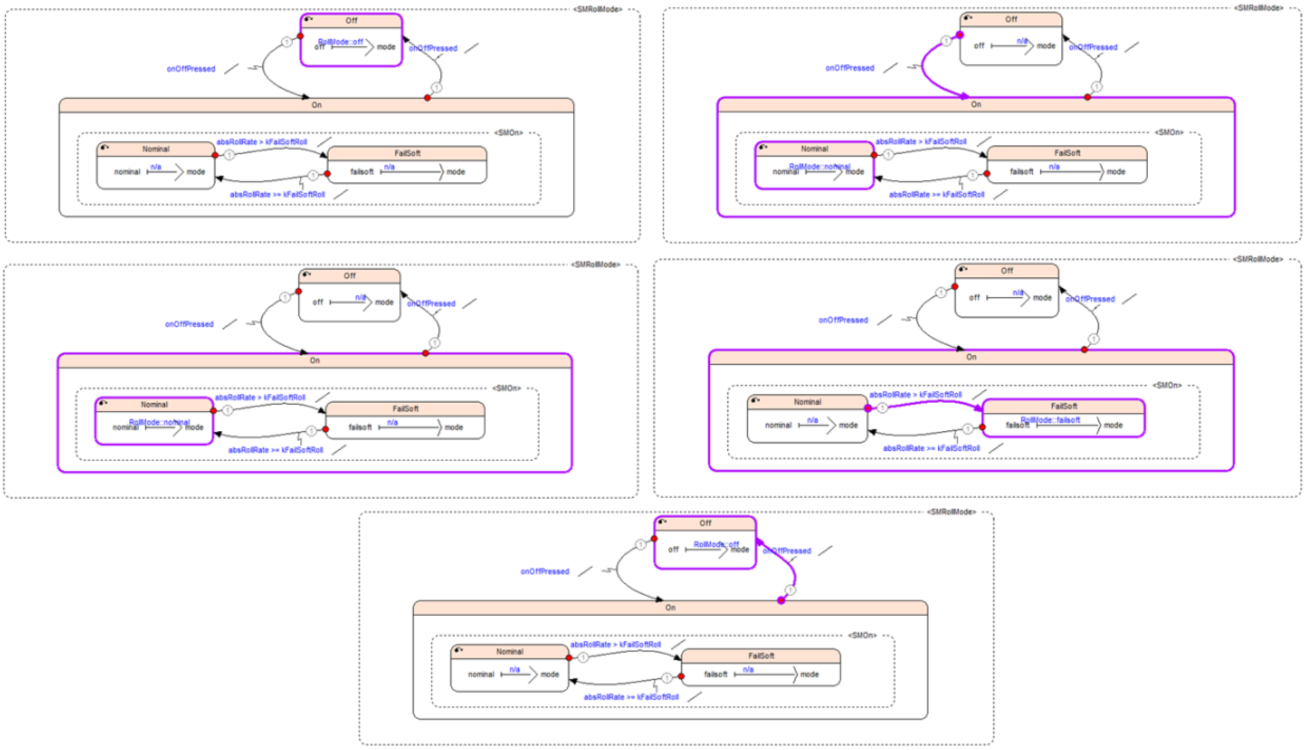


Fig 3: Detailed Control Flow diagram of RollMode Management

(ii) Simulation: The stability controller is then run interactive simulation sessions to dynamically check the model. The SCADE suit provides facility for different docking windows. The simulation can be viewed using graph and watch windows. The test cases for roll control simulation are shown in the Table 2. The results of the test cases are depicted in the Figure 4. It shows the variation in output when any one/more of the input parameter changes.

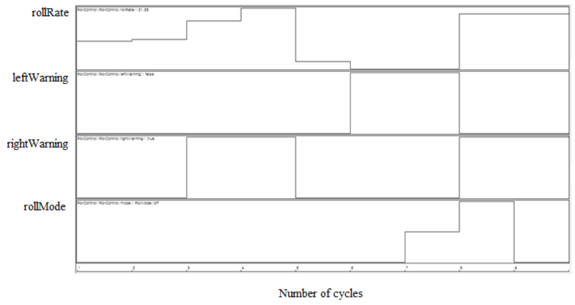


Fig 4: Test case results

(iii) Formal Verification: Formal verification analysis is also possible in SCADE Suit. The proof objectives are represented as a separate SCADE node, called an 'Observer' node. The observer node evaluates the property to be proved as a Boolean condition that takes the value false if it is broken. The property is encoded using standard SCADE design elements; therefore the new notation of representation is not required. The observer node is connected to the node where the design verification is carried out. In our work, for verifying system correctness, Observer property, Divide by zero and Overflow stack conditions were checked. The result of the analysis is shown in the Figure 5. The Proof meaning is also explained in the SCADE Suit documentation [6,18] by Esterel Technologies as shown in the Table 3.

Table 3. Proof Meaning

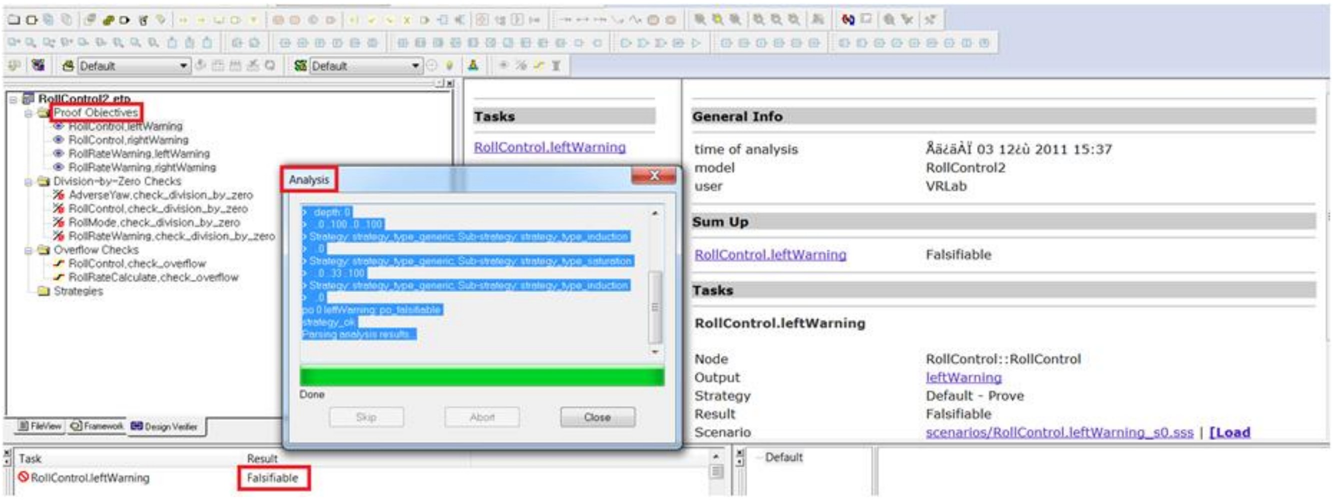
Proof result	Meaning
Valid	The verified property is always true mathematically
Falsifiable	Property is false because Design Verifier detects a valuation of the system inputs such that the output of the observer operator is not equal to the value specified in the proof objective
Indeterminate	The proof reaches no significant conclusion
Interrupted	Either manually aborted the analysis in the status window, or strategy time-out
Stop Depth Reached	The analysis reaches its execution cycle depth set in the debug strategy and Design Verifier cannot report any significant result.
Raised an error	The cause of error displays in the message field of the report.
Error: Non linear property	Verification is impossible because the property is expressed with non-linear expressions or functions.
Contradictory	Contradictory assertions in the analyzed design.

4.4 CODE GENERATION

The next stage consists of automatic code generation. Since the SCADE language is formally specified, most of the checks can be performed at model level during the design phase. After the semantic check, the build command generates the code automatically. The generated code structure is same as any C-file with its corresponding header file for each operator in the model, unless the operator is expanded. For each non-expanded operator, two functions are generated, one for the operator memory reset and the other for its activation. The call graph of the C function corresponds to each model. The generated code is correct and optimized by SCADE KCG Code generator.

4.5 GRAPHICAL DISPLAY

The last stage of the implementation consists of integrating the model design with the graphical panel, which is supported



RollControl2.etp
Proof Objectives
RollControl.leftWarning
RollControl.rightWarning
RollRateWarning.leftWarning
RollRateWarning.rightWarning
Division-by-Zero Checks
AdverseYaw.check_division_by_zero
RollControl.check_division_by_zero
RollMode.check_division_by_zero
RollRateWarning.check_division_by_zero
Overflow Checks
RollControl.check_overflow
RollRateCalculate.check_overflow
Strategies

Analysis

```
>   depth: 0
>   0..100..0..100
> Strategy: strategy_type_generic; Sub-strategy: strategy_type_induction
>   0
> Strategy: strategy_type_generic; Sub-strategy: strategy_type_saturation
>   0..33..100
> Strategy: strategy_type_generic; Sub-strategy: strategy_type_induction
>   0
po 0 leftWarning: po_falsifiable
strategy_ok
Parsing analysis results...
```

Done

Task	Result
RollControl.leftWarning	Falsifiable

Tasks
RollControl.leftWarning

General Info

time of analysis	Åä¿äÀÏ 03 12¿ù 2011 15:37
model	RollControl2
user	VRLab

Sum Up

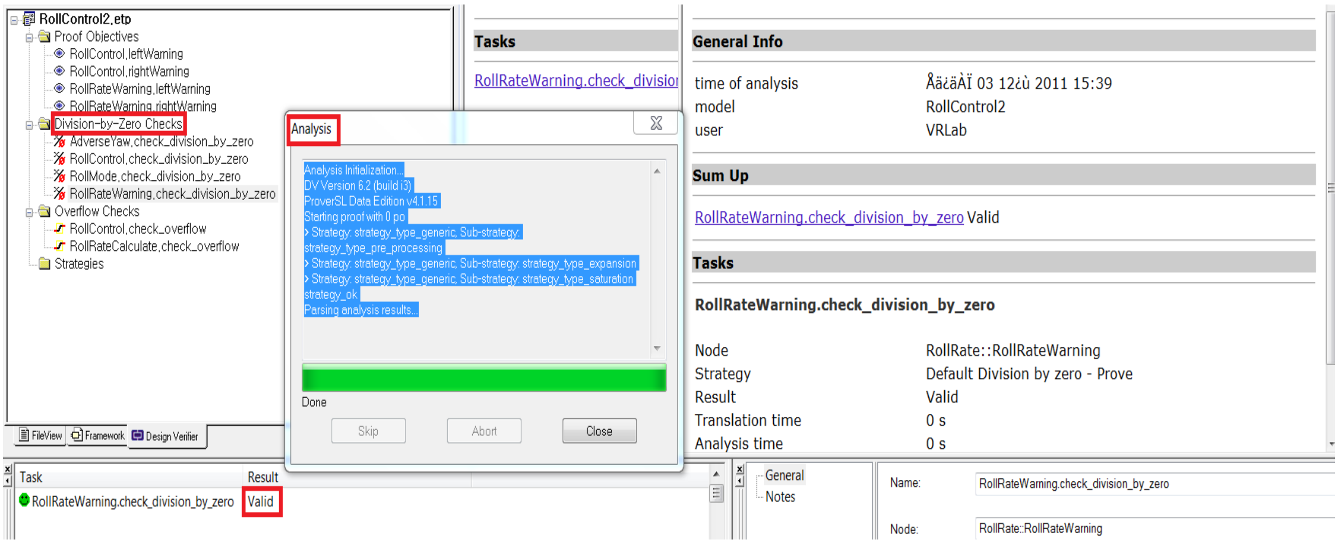
RollControl.leftWarning	Falsifiable

Tasks

RollControl.leftWarning

Node	RollControl::RollControl
Output	leftWarning
Strategy	Default - Prove
Result	Falsifiable
Scenario	scenarios/RollControl.leftWarning_s0.sss | [Load

Fig 5(a): Verifying system correctness – Observer property



RollControl2.etp
Proof Objectives
RollControl.leftWarning
RollControl.rightWarning
RollRateWarning.leftWarning
RollRateWarning.rightWarning
Division-by-Zero Checks
AdverseYaw.check_division_by_zero
RollControl.check_division_by_zero
RollMode.check_division_by_zero
RollRateWarning.check_division_by_zero
Overflow Checks
RollControl.check_overflow
RollRateCalculate.check_overflow
Strategies

Analysis

```
Analysis Initialization...
DV Version 6.2 (build i3)
ProverSL Data Edition v4.1.15
Starting proof with 0 po
> Strategy: strategy_type_generic; Sub-strategy:
strategy_type_pre_processing
> Strategy: strategy_type_generic; Sub-strategy: strategy_type_expansion
> Strategy: strategy_type_generic; Sub-strategy: strategy_type_saturation
strategy_ok
Parsing analysis results...
```

Done

Task	Result
RollRateWarning.check_division_by_zero	Valid

Tasks
RollRateWarning.check_divisior

General Info

time of analysis	Åä¿äÀÏ 03 12¿ù 2011 15:39
model	RollControl2
user	VRLab

Sum Up

RollRateWarning.check_division_by_zero Valid

Tasks

RollRateWarning.check_division_by_zero

Node	RollRate::RollRateWarning
Strategy	Default Division by zero - Prove
Result	Valid
Translation time	0 s
Analysis time	0 s

Name: RollRateWarning.check_division_by_zero
Node: RollRate::RollRateWarning

Fig 5(b): Verifying system correctness – Division-By-Zero check

by SCADE Suit Rapid Prototyper. The Rapid Prototyper builds interactive graphical panels by connecting inputs/outputs between models and graphical panels. It includes a library of control widgets including buttons, LEDs, knobs, text and numerical entry boxes. The prototype simulation of the warning signals is shown in Figure 6. If the value of roll rate is less than -15 degrees/sec, the left warning alert is activated and if roll rate is greater than +15 degrees/sec , the right warning alert is activated.

5. CONCLUSION AND FUTURE WORK

This paper presented the design and verification of the stability controller of an aircraft starting from requirement specification passing through control design and its verification. Experiments using the various test cases were conducted and prototype simulations of the warning signals were demonstrated to further validate the design against system requirements. This represented two main advantages i.e. cost reduction and development time reduction. The objective is to validate the effectiveness of SCADE tool as well as to evaluate its scalability in testing real-world applications. Similar experiments can be conducted in other research domains like rail-road transportation and high-end automotive systems where a design environment is essential to connect developers through their participation in the execution of the engineering process. Our future work is to test the feasibility of SCADE tool with the real control software for airplanes and its effect on cost reduction.

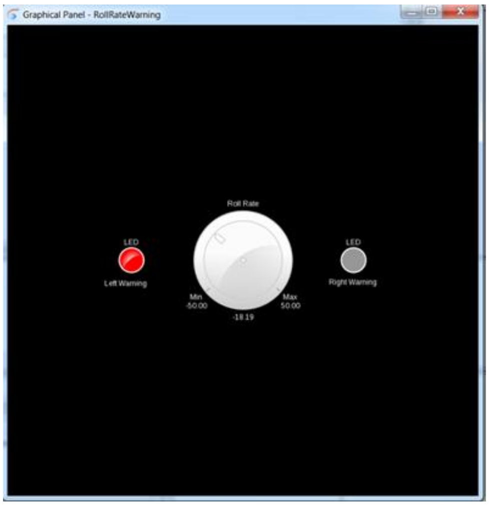


Fig 6: Prototype simulation of warning signals

REFERENCES

[1] Nelson, R. C. 1989. Flight stability and automatic control. New York, McGraw-Hill.

[2] Perkins C.D. and Hage R.E. 1949. Aircraft Performance, Stability and Control. John Wiley.

[3] Pamadi, B. N. 2004. Performance, stability, dynamics, and control of airplanes. Aiaa.

[4] Davies, M.2003. The standard handbook for aeronautical and astronautical engineers. New York: McGraw-Hill.

[5] Ogata K. 1984. Modern Control Engineering. Prentice-Hall, India.

[6] Esterel Technologies http://www.estereltechnologies.com/industry/avionics/.

[7] Anderson, J. D. 2005. Introduction to flight (Vol. 199). McGraw-Hill.

[8] Etkin, B., & Reid, L. D. 1982. Dynamics of flight: stability and control (p. 103). Wiley.

[9] Houghton E.L and Carruthers N.B. 1982. Aerodynamics for Engineering students. Arnold.

[10] McCormick B.W. 1995. Aerodynamics, Aeronautics and Flight Mechanics. John Wiley.

[11] Abzug M., and Larrabee E. 2002. Airplane Stability and Control: A History of the Technologies that Made Aviation Possible. Cambridge University Press.

[12] Maine, Richard E. and Kenneth W. Iliff. 1986. Application of Parameter Estimation to Aircraft Stability and Control - The Output-Error Approach. NASA RP-1168.

[13] Yoo, J., Jee, E., & Cha, S. 2009. Formal modeling and verification of safety-critical software. Software, IEEE, 26(3), 42-49.

[14] Bellmann, T. 2009, September. Interactive simulations and advanced visualization with modelica. In Proceedings of the 7th Modelica Conference, Como, Italy. ISBN 978-91-7393-513-5. ISSN 1650-3740

[15] Bünte, T., & Chrisofakis, E. 2011 A Driver Model for Virtual Drivetrain Endurance Testing. In: Proceedings of the 8th International Modelica Conference

[16] Bhatt, D., Hall, B., Dajani-Brown, S., Hickman, S., & Paulitsch, M. 2005. Model-based development and the implications to design assurance and certification. In Digital Avionics Systems Conference, 2005. DASC 2005. The 24th (Vol. 2, pp. 13-pp). IEEE.

[17] Colaço, J. L., Pagano, B., & Pouzet, M. 2005, September. A conservative extension of synchronous data-flow with state machines. In Proceedings of the 5th ACM international conference on Embedded software (pp. 173-182). ACM.

[18] Esterel Technologies. http://www.myscadesupport.com/