Proceedings of the International Conference on Aerospace Science and Technology
26 - 28 June 2008, Bangalore, India

INCAST 2008-011

INDIGENOUS SOFTWARE DEVELOPMENT FOR CIVIL AIRCRAFT SYSTEM

J.Jayanthi[1], Manju Nanda[2], Dr.M.R.Nayak[3]
1 Aerospace Electronics Division, National Aerospace Laboratories, Bangalore, India,
jayanthi@css.nal.res.in
2 Aerospace Electronics Division, National Aerospace Laboratories, Bangalore, India,
manjun@css.nal.res.in
3 Aerospace Electronics Division, National Aerospace Laboratories, Bangalore, India
mrnayak@css.cmmacs.ernet.in

ABSTRACT: *Presently, there is a boom in the aviation industry. Until recently, India had been buying all kinds of aircraft right from jumbo-jet to small executive class. But due to the foresight of our scientists and to make our country march towards the goal of self reliance in technology development, programs were initiated to set foot in the aviation industry. Towards this effort, initially, the prestigious Light Combat Aircraft, Tejas was started in the military sector and a trainer aircraft Hansa in the civil sector. Subsequently, SARAS, a multi role aircraft was planned and currently the program is going on successfully towards Limited Series Production and certification. Lot of indigenous effort in various aircraft domains has taken place successfully for the SARAS aircraft to provide the self-reliance. This paper talks about one of such success story of the indigenous development of a safety critical software for civil aerospace application first time in the country. The software is developed for the Stall Warning and the Aircraft Interface Computer system which is an integral part of the SARAS avionics. The entire software development process from the requirement phase to the certification phase is described. The system is being used as part of the avionics for the country's first 14-seater aircraft. The success of the software development has been proven by the successful flights of the SARAS aircraft and the analysis of the data collected during the flight tests proves the correctness and reliability of the system. The software design, development and qualification experience and the successful flight trials of the embedded system developed as per the civil standard certification has initiated more design projects in the organization saving a lot of money for the organization and the country providing a premier base for achieving self reliance in software development for civil aviation.*

1. INTRODUCTION

Software is pervading in to every system possible currently. The main reason for this shift towards software is adaptability. Gone are the days when software was a monolithic block implementing few functionalities. Present day software is modular in nature and very huge in size. Distributed nature of underlying hardware in some cases also has an impact on the software. Aircraft industry is no exception to this trend. In the aviation sector, now the shift is towards open system architecture where applications can be added like plug and play. Due to the software predominance, lot of reconfigurability is attempted with less number of hardware, making more effective use of the hardware. When role of software is increasing by leaps and bounds, the design and development also becomes more complex. The complexity in design and development also leads to difficulty in verification and validation.

Developing software for aircraft differs from:

- Developing software for other embedded systems like entertainment electronics which become obsolete fast as their customer is bothered about new fashionable features rather than the safety of the equipment. Here the

- schedule is entirely driven by market needs which often vary substantially in a year.
  Developing regular computer software like a word processor where business/life criticality are negligible but compatibility with older versions, adhering to de facto standards and again meeting the market expectations is more important. Here again the quality is compromisable.
- Banking software where there is a financial risk but unlike Aerospace software, a risk concerning interfacing to external hardware is minimal. Also Banking software may not be responsible (i.e., directly) for killing its user.

- In aircraft, risk for life (pilot, crew, passengers) as well as great financial impact is there. Also the software once written does not change for 15 to 20 years and hence the stringent standard. Hence it requires that the avionics software be produced such that it minimizes or removes the risk of a malfunction or failure.

The main difference between avionic software and conventional embedded software is that the development process is required by law and is optimized for safety. To ensure safety of airborne systems, RTCA has evolved a standard to be followed by all software designers. DO-178B, Software Considerations in Airborne Systems and Equipment Certification is guidance for software development published by RTCA. The FAA accepts use of DO-178B as a means of certifying software in avionics. [1], [7]

2. INDIGENOUS AIRBORNE SOFTWARE DEVELOPMENT NEED AND CHALLENGES

Airborne software development and certification of use in aircraft is very expensive. The increased cost factor compared to other software development is due to the effort required to adopt the process as per DO 178B and the artifacts to be generated. DO 178B provides only the guide lines to be followed and the DO 178B guide lines have to be customized for each project. Initial cost of software development for airborne system is exorbitant if it is done for the first time specific to an aircraft. Hence software development for systems which depend on the aircraft structure and dynamics are very expensive when executed by oversea vendors. Further, the process will not be established in our premises and for any further changes however small the change may be, the vendor has to do at a high cost.

Such difficulties are overcome by developing indigenously the software for the aircraft system. The flexible nature of DO-178B's processes and entry/exit criteria make it difficult to implement for the first time, because these aspects are abstract and there is no "base set" of activities from which to work. The intention of DO-178B was not to be prescriptive. Therefore, there are many possible and acceptable ways for a real project to define these aspects. This can be difficult for the first time, if a company attempts to develop a civil avionics system under this standard, and has created a niche market for DO-178B training and consulting. [1]

3. STALL WARNING/AIRCRAFT INTERFACE COMPUTER (AIC) SYSTEM OVERVIEW

The Stall Warning System/Aircraft Interface Computer System is one of the avionics systems in the SARAS aircraft. The stall computation is highly dependent on the aircraft structure and dynamics. It was decided to indigenously develop the software for the system. A qualified hardware was identified for the computer. The following paragraphs present the details of indigenous design and development.

The SARAS is a twin turbo-prop multi-role aircraft with air taxi and commuter service as its primary roles. Two crew members, namely pilot and co-pilot operate the aircraft. It is suitable for flying in all weather conditions and is equipped for day and night flying.

The SWS/AIC architecture are shown in Figure 1. The SWS / AIC consists of two Angle-of-Attack (AOA) sensors, an SWS/AIC computer consisting of two computing subsystems mounted in a single chassis, one pitch trim actuator and one-stick shaker. The aircraft is equipped with two AOA vane sensors, each with two independent position detectors (potentiometer based) to provide AOA information to the two channels of SWS/AIC computer. Each channel of the computer receives Analog, Discrete and ARINC signal as input from the various sensors/switches/equipments interfaced to the computer. The stall warning control laws, and firing tables are embedded as software in each processor to announce stall condition to the pilot. The SWS/AIC provides AOA information as an ARINC output to other systems including the primary displays to the pilot/copilot on the EFIS, and the flight data recorder. [3]

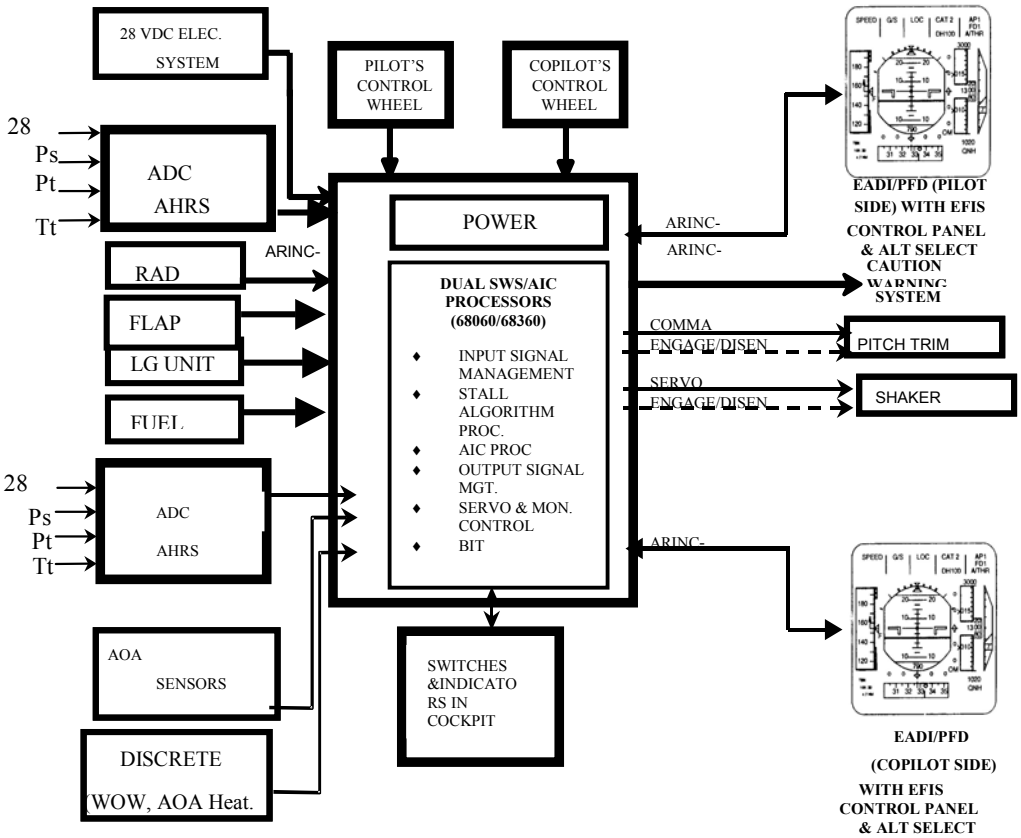

Figure 1:SWS/AIC system Architecture

3.1 SOFTWARE CONSIDERATIONS

The software development for any airborne system commences with the identification of the efforts that is required to be put in for the software development. The efforts required depend on the criticality of the function and its effect on the system and aircraft safety. RTCA DO178B provides guidelines for assessing the level of effort required to meet the safety levels required for the system under development. The level of effort required to show compliance with certification requirements varies with the failure condition category. This is the basis for establishing the software level.

3.2 SOFTWARE DEVELOPMENT FOR STALL WARNING/AIRCRAFT INTERFACE COMPUTER SYSTEM

Stall is a condition in which the aircraft starts loosing height rapidly due to reduction in the lift force. Once the aircraft enters stall, sometimes it may not be possible to come out of it. A failure to announce an impending stall condition can lead to catastrophic condition. Hence, aircrafts are fitted with a warning system to alert the pilot of an impending stall condition. Stall Warning System is a safety critical system. The stall warning system development is for a civil aircraft program. Hence the standard adopted for the software development is RTCA DO 178B. The failure to announce the stall can lead to catastrophic events. Hence the safety analysis carried out recommended software development efforts as per level A of DO178B. The SWS/AIC computer detects the warning conditions for take off, landing, baro altitude mismatch, hydraulic low pressure, pitch trim and V_{MO} (over speed) based on the logic for each warning generation. The SWS/AIC incorporates pitch trim monitoring function and pitch trim actuation. The warnings are provided on a Caution and Warning Panel as discrete outputs, which light the LEDs on the Panel and also generate an audio warning. Stall warning also provides actuation of stick shaker in addition to visual and aural warning. [3].

The entire system functionality has been logically mapped in to software requirement groups like external interface requirements, hardware initialization requirements, failed mode requirements, real time control requirements, external input/output processing, cross channel data link processing, Built In Test requirements, Functional requirements for SWS, functional requirements of AIC, monitor requirements, error handling requirements, performance requirements[2],[4].

The various processes as per DO178B level A were established. The established processes were adhered to strictly during the indigenous development. The software life cycle process for SARAS-SWS/AIC software development is depicted in Figure 2.

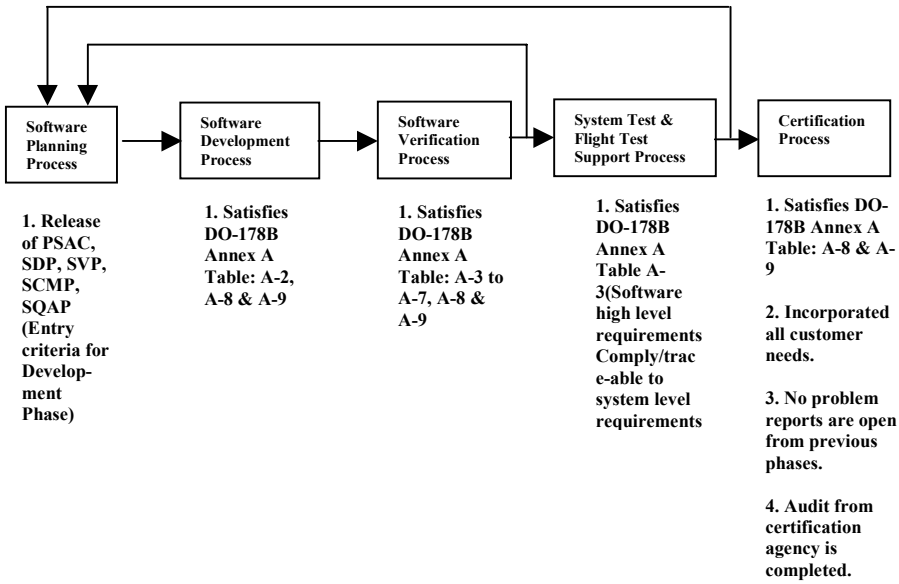

Figure 2: SWS/AIC Software Life Cycle

3.3 CHALLENGES IN ESTABLISHING SOFTWARE DEVELOPMENT STANDARDS

The purpose of software development standard is to define the rules and constraints for the software development processes keeping in view always the safety of the system. The

software development standard includes the Software Requirements standards, the Software Design standards, and Software Code standards. The selection of proper programming language for the safety critical software development is very important. The programming language must be such that it enforces strict discipline in programming. But practically, there are other factors to be considered such as availability of development and debug tools for the chosen language, availability of man power with proficiency in the chosen language. Considering these factors, C was selected as the programming language. Even though C is not ideally suited for development of safety critical software, a safe subset of C can be formed by eliminating the usage of all problem causing features of C. This can qualify 'C' for safety-critical applications. A coding style was also identified. The identified subset of the language and the coding style adopted for the project form the code standard for the project. [5]

The software design standards stipulates how the software is described, the complexity restrictions, design constraints, scheduling, exception handling and interrupt handling. For SARAS SWS/AIC, the software design standards were established to ensure a high level of determinism, testability and maintainability in the code design. The design effort was carried out manually by means of diagrams. A complex design is augmented with a flow chart for easy understanding. [5]

It is highly essential that the chosen development environment does not introduce any error during the development and qualification of the software. An in-house test rig was developed and qualified to ensure that no ambiguity is introduced from the test set up. To maintain the time schedule and reduce the human error, automated tools were used for testing and configuration management. Figure 3 shows the qualified test rig developed for the SWS/AIC system.

Figure 3: Qualified Test Rig used for SWS/AIC Testing

Any airborne software cannot be installed in aircraft unless cleared by certification agency. In addition to following the process, artifacts have to be generated at every stage to provide the proof of compliance. Based on these artifacts, and the exposure provided by the applicant who seeks the certification, the certification authority qualifies the software. The main challenge in certification for the SWS/AIC was that the DGCA were involved for the very first time in the qualification of civil aviation software. In addition, since the product used COTS software, lot of additional tests had to be conducted to gain the confidence of the certification agency. [6]

In-spite of all testing on ground, the real performance of the system can be established only during flight trials. The software development was completed, qualified by DGCA for safety of flight and installed in the SARAS. Even though formal flight trials are yet to commence, the system is being monitored as part of every flight test. The data collected during flight is being analyzed. The main objective of this analysis is to fine-tune the signal thresholds and verify the functional correctness of the system. The analysis helped in fine-tuning the system behavior since we have the total control on the process and the product. One such instance is the detection of mounting error in the angle of attack sensors. Figure 4 shows the plot which enabled to detect the mounting error.

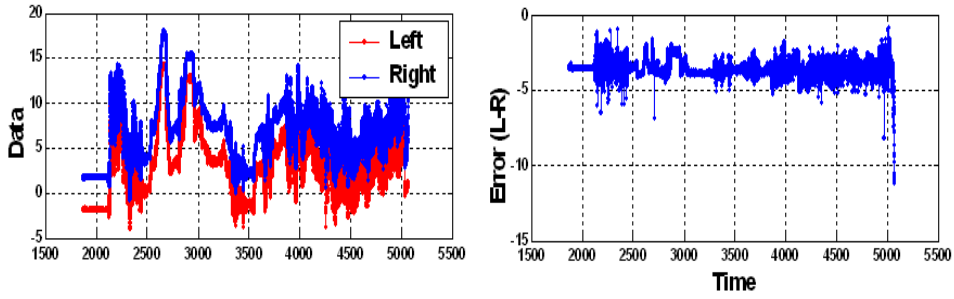

Figure 4: AOA Plot generated from Flight Data

3.4 METRICS

The software metrics for the SWS/AIC software is given in the following table.

Activity	Metrics
Total number of modules developed	137
Average number of lines in each module	100 lines
Source lines of code	12000
Number of software problems raised	789
Low level testing phase	450
HSI testing Phase	339
Number of Functional Tests	22
Total number of test cases for HSI including all sub tests	5000 plus
Size of executable code	336Kb
Total memory usage	16.80%
Time duration	30 months
Number of persons	6
No. of lines of code/day	2.2 sloc

4. CONCLUSION

The indigenous development effort has given the team at NAL a very good exposure and expertise of DO 178B software development process for level A. The available expertise is being used for SARAS Autopilot and EICAS design and development leading to lot of cost saving. The expertise can be used at institution level to provide a premier base for achieving self reliance in software development for civil aviation.

5. ACKNOWLEDGMENTS

The authors wish to acknowledge the Director, Head and the INCAST for providing an opportunity to share the experience of indigenous software development.

REFERENCES

[1] WIKIPEDIA, http://en.wikipedia.org/wiki/DO178B
[2] "Software Considerations for Civil Certifications of Stall Warning System", Technical Seminar on Advances in Aerospace Sciences held in December 2003.
[3] SYSTEM REQUIREMENTS SPECIFICATIONS FOR SARAS SWS / AIC SYSTEM, DR-19A, June 2005
[4] Software Design Description for SARAS SWS/AIC system, DR-22, June 2005
[5] Software Development Plan for SARAS SWS/AIC system, June 2005, TB-04A
[6] Plan for Software Aspects of certification for SARAS SWS/AIC, Sep 2003, TB-03
[7] RTCA DO-178-B, Software Considerations in Airborne systems and Equipment Certification, Dec 1992