ISSN 2229-5518

Quantitative Metrics for validating the effectiveness of the Model based approach for indigenously developed SWS/AIC system

Manju Nanda, Chinmayi S Jamadagni

Abstract— The aim of this paper is to validate the effectiveness of model-based approach for the indigenously developed stall warning and aircraft interface computer system (SWS/AIC) by generating the software engineering process metrics and the development of the empirical relationship between the conventional and the model-based approach. The quantitative metrics for software analyzability, changeability, testability, stability, traceability, safety compliance, reliability, design time, debug time, upgrade time, reusability, readability, maintainability, modularity, reachability and availability is derived and generated for the two approaches to demonstrate the effectiveness of the model-based approach. The empirical relationship developed helps in analyzing the reduction in effort for development of safety critical software using model-based approach.

The metrics generated and the empirical relationship derived between the two approaches proves the effectiveness of the model-based approach over the conventional approach. The results of this work are encouraging for incorporation of the model-based approach for the design, development and verification and validation of safety critical systems.

Index Terms— Formal methods, Model based approach, Verification and Validation, Safety critical systems, Metrics, Stall Warning System.

1 INTRODUCTION

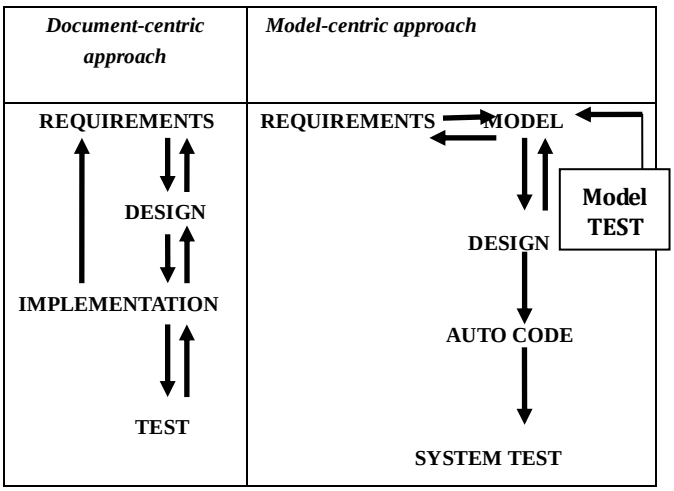
Development of critical systems for i.e. the aeronautics or automotive industry requires a strict interdisciplinary approach and conformance to standards and specifications in order to ensure safe systems, since failures are often catastrophic and with loss of life as a consequence.

The development of embedded systems with real-time and other critical constraints raises distinctive problems. In particular, development teams have to make very specific architectural choices and handle key non-functional constraints related to, for example, real-time deadlines and to platform parameters like energy consumption or memory footprint. The last few years have seen an increased interest in using model-based engineering (MBE) techniques to capture dedicated architectural and non-functional information in precise (and even formal) domain-specific models in a layered construction of systems. MBE techniques are interesting and promising because they allow to capture dedicated architectural and non-functional information in precise (and even formal) domain-specific models, and they support a layered construction of systems, in which the (platform independent) functional aspects are kept separate from architectural and non-functional (platform specific) aspects, where the final system is obtained by combining these aspects later using model transformations.

Model based engineering approach is the formalized application of modeling support system requirements, design, analysis and V&V. Mathematical rigor enables users to analyze and verify these models at any part of the program life-cycle: requirements engineering, specification, architecture, design, implementation, testing, maintenance and evolution. The use of mathematical techniques reduces the possible personal interpretation.

The paper discusses a pioneering framework in making the engineering process effective. The framework includes model based approach for the design life cycle and demonstrates the effectiveness of the approach by generating metrics. The approaches adopted for the comparative case study are conventional and model based. The design cycle for the conventional and model based approaches is as shown in Fig 1.

Fig 1



Model-Based Design [3] with automatic code generation is an important and established technology for developing aerospace embedded control systems. Early verification, validation, and test of models and generated code using software tools with accompanying workflows are increasingly used. Model-based design provides numerous advantages over the traditional design approach. Using the model-based approach, you reduce the risk of mistakes and shorten the development cycle by performing verification and validation testing throughout the development instead of only during the final testing stage. Design evaluations and predictions can be made much more quickly and reliably with a system model as a basis. This iterative approach results in improved designs, both in terms of performance and reliability. The cost of resources is reduced, because of reusability of models between design teams, design stages, and various projects and the reduced dependency on physical prototypes. Development errors and overhead can be reduced through the use of automatic code generation techniques. These advantages translate to more accurate and robust control designs, shorter time to market, and reduced design cost.

2 STALL WARNING SYSTEM AND AIRCRAFT INTERFACE COMPUTER SYSTEM

The system under consideration is the Stall Warning System used in aircrafts. The purpose of the SWS/AIC system is to provide stall warning whenever the aircraft approaches stall angle of attack, display continuously the angle of attack information on the primary display, provide interface between Caution warning panel (CWP) and systems which require an interface for CWP and provide pitch trim function and monitoring. The **stall warning system** is designed and modeled using both conventional process and model process and the metrics obtained are compared and analyzed in order to obtain the footprints and figure of merit.

The model based approach [4] allows engineers to design embedded systems and simulate them on their desktop environment for analysis and design. Model-Based Design provides a variety of code generation capabilities that teams use to generate source code for many purposes including simulation, rapid prototyping and hardware-in-the-loop testing. Model-Based Design promotes a requirements-oriented project view and greater integration and reuse between conceptual and detailed modeling and design work. The block diagram of the SWS/AIC is depicted in Fig 2.

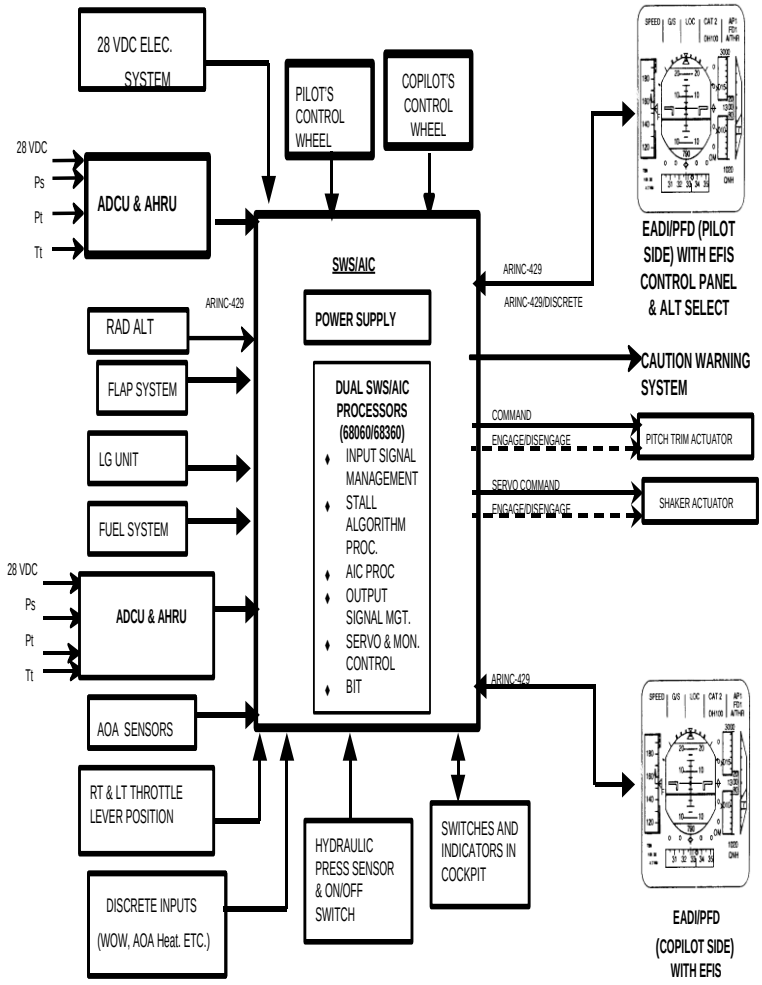


Fig 2

The model based formal implementation of the stall warning system is done using Mathworks toolset (R2010a)[1]. After creating the model, it has to be tested extensively to ensure that model is identical to the legacy source code. So the model validation and comparison of its outputs with the legacy source code becomes an important task in MBSE. MBSE uses a V- Model / Life Cycle for the model creation and its validation The Matlab/Simulink model of SWS is depicted in Fig 3. The SWS modeled in Simulink is simulated to check for functionality and then auto code is generated for the model. The auto code generated is compared with the manual code, thus highlighting the advantages of model based approach over the conventional approach. The model is then verified using Simulink Design Verifier. The SDV generates Auto test cases for coverage of the model.

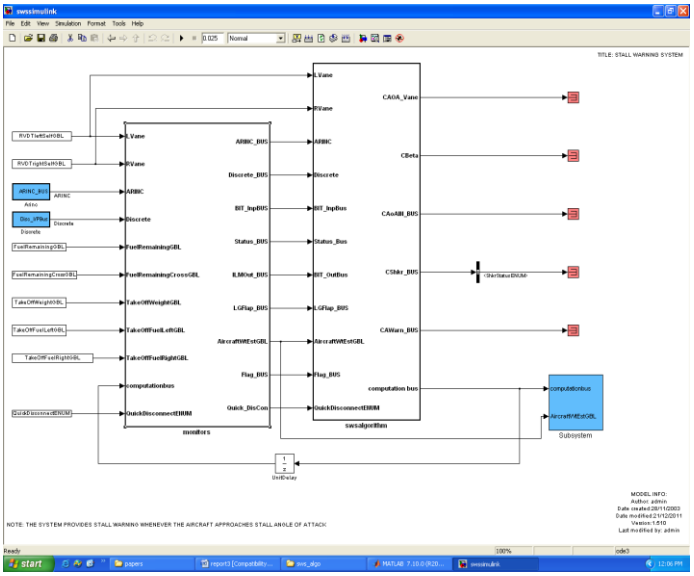


Fig 3

The Stall warning system module is subjected to Simulink Design Verifier which checks the model for compatibility and generates test cases for the functionality. The SDV includes **formal prover engine** which proves the properties of the model. The SDV report for the Landing gear module along with coverage metrics and test cases are discussed below. Fig 4 depicts the test unit for the landing gear module. The inputs from the harness unit are given to the test unit in the form of signal builder as shown in Fig 4. Depending upon the module, the SDV log gives **compatible**, **partially compatible** and **incompatible** results. The landing gear module taken into consideration is compatible with the SDV. The SDV generates a test unit (module that is compatible with the SDV) for verification purposes to which inputs are provided through signal builder block. The signal builder block serves as a tool for generation of test cases i.e. auto test cases for coverage analysis. The Test case explanation in document format can also be obtained from the tool. Once the auto test cases are obtained, they are run to generate the coverage report for further analysis.

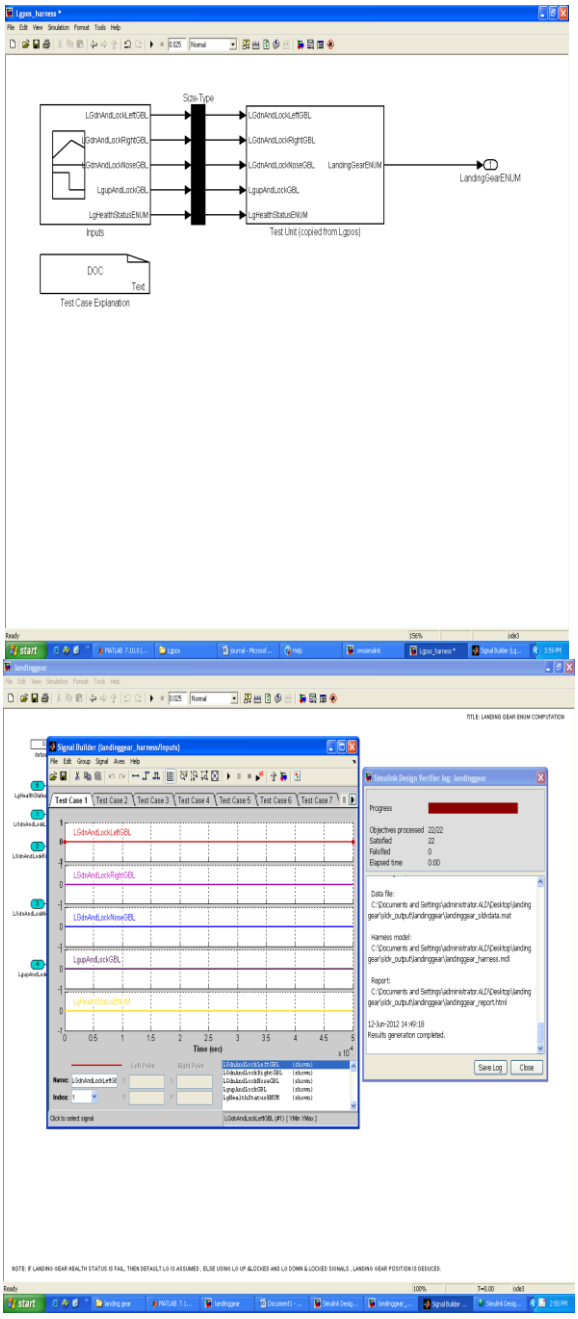


Fig 4

3 SYSTEM PROPERTY METRICS

The performance metrics [2] for analyzing the system design and the process are carried out on the SWS/AIC system. The system is first developed using the conventional document centric approach and later the model based formal approach. The SWS/AIC system is modeled using the Simulink 2010a toolset. The auto code generated is compared with the manual code generated using the conventional approach. The two processes are compared and the metrics ob-

tained are shown in TABLE I. Descriptive computation and the comparisons of the two approaches are shown in TABLE II. The system property metrics proposed and analyzed for the case study are defined as follows and their interrelation is described formally as empirical formulae. The relationship between the System Property Metrics and the performance metrics can be described in the tree diagram with weighted methodologies in Fig 5.

Metrics Definition:

Reliability of a system can be defined as its ability to perform a given trial or probability that an item will last for a given period of time.[5]

Reachability can be attributed to analyzability and traceability of a system .Hence an empirical relationship exists between the former and later metrics.

Availability is attributed to analyzability and stability of a system.

Maintainability of a system is dependent on its changeability, modularity, traceability, design time and upgrade time.

Safety is the most critical metric and cannot be compensated for in any approach. Safety critical systems are defined by this metric based on certain standards namely DO178B, DO178C etc... Safety metric is attributed to the testability and modularity towards fault tolerance of a system.

Reusability defines the no of files that are being reused during simulation and code generation.

Readability includes clarity in interfaces, uniformity in appearance, coding and documentation.

A system is said to possess ***changeability*** if it is flexible, adaptable, scalable and modifiable [15].

Analyzability of a system is defined from the effort required to detect deficiencies and to modify it [6][7].

Testability of a system is satisfied if it is controllable, observable, isolatable, understandable, and automatable and offers heterogeneity [8].

Stability is defined as the ability of the system not to hang, not to lose data, not to disrupt system functionality and be predictable [16].

Modularity of a system defines its level of independence [11][12][13][14].

PARAMETER (Metrics)	CONVENTIONAL APPROACH	MODEL BASED FORMAL APRROACH
Executable Size	259 kb	128 kb
Lines of code	53 (+ 332)	17 (+1615)
Commented lines	26(+103)	41(+721)
Analyzability	20%	60%
Changeability	50%	75%
Testability	66.6%	83.3%
Stability	Yes (99.9%)	Yes (99.9%)
Traceability [9][10]	75% (document centric)	100% (model centric)
Safety compliance	DO178 B (77.2%)	DO178B (94.1%)
Reliability	Yes	Yes
Design time	330 man hours approx.	160 man hours approx
Debug time	3 man-hours (for a complex functionality)	1 man-hour (for a complex functionality)
Upgrading time	2 man-hour for a single functionality upgrade	Maximum of 10 minutes for a single functionality upgrade
Reusability	0%	20%
Readability	90%	100%
Maintainability	68.33%	91.35%
Modularity	80%	100%
Reachability	50%	80%
Availability	78.2%	90%
Development Effort	22.6%	12.7%

TABLE I

PARAMETER	CONVENTIONAL APPROACH	FORMAL APPROACH
Approach	Document based	Model based
Readability	Textual; Interface comply coding standards and design styles; documentation complies in house documentation standards	Modular ; Interface comply coding standards and design styles; MAAB style
Changeability	Modification done at all levels of design ; Scalable : more effort; Manual	Modification done at top level of the design ;Scalable by tool ; Automated
Testability	Driven by impact analysis; Code review; Functional isolation ; Manual	Driven by functionality of the model; Module isolation; Auto code generation ; Tool dependent
Analyzability	Manual ; Code reviews; Impact analysis	Automated; Tool dependent
Traceability [9]	Requirements → Design →Code →Report ; traceability matrix generated manually	Requirements→Model→design→code→Report ; traceability matrix generated by tool
Stability	Stable	Stable
Modularity	Level of independence based on architecture ; Interaction by means of drivers; Specification based on configuration files; Coupling on control and data coupling	Level of independence based on modules; Interaction by means of functionalities of models; Specification based on inputs ;control and data coupling
Safety	Safe	Safe
Design time	More ; manual	Less ; automated
Verification &Validation	Done at the end of the cycle	Can be done at the start of the cycle
Test cases	Manual ; depends on functionality	Tool generated ; optimized

TABLE II

From the case study implemented, an empirical relationship is deduced between the two approaches.

- ***Size of manual code ≥ 2 * (size of auto code generated)***
- ***Commented lines = exp (executable lines of code)***
- ***System property metrics for formal approach = 1.258 * (conventional approach)***
- ***Development Effort*** is calculated from the statistical model formula:

E = 5.2 * L^ 0.91;

L is the no of lines of code in thousands. (The formula is derived by Walston and Felix with **a =5.2** and **b= 0.91**, constants obtained by regression analysis)

There is a 32 % reduction in the number of lines of code compared to the conventional approach. This is a considerable reduction factor when the complexity of the system increases. The effort for programming also reduces; hence designers can concentrate more on the other phases of the life cycle. From the data obtained from the case study, comparison chart is developed for the property metrics in order to obtain the improvement in the approach involving the formal methods as compared to the conventional approach.

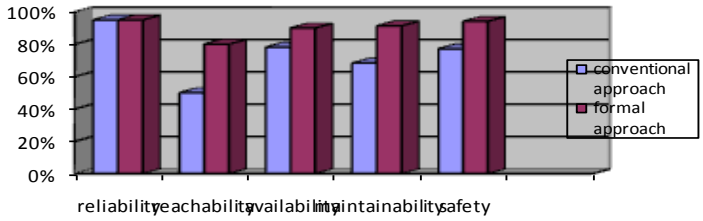


FIG 6

From the above chart (Fig 6) it can be observed that the model-based approach provides an improvement of 16.34% in the proposed system property metrics keeping the reliability factor intact. This further helps in obtaining an effective systems engineering framework that integrates formal methodologies. The effect of the weighted methodologies on the system property metrics for both the approaches is depicted in the charts given below. (Fig 7 a and Fig 7 b)

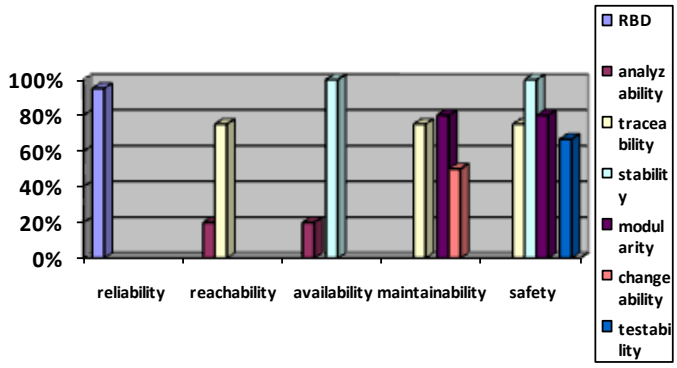


Fig 7(a)

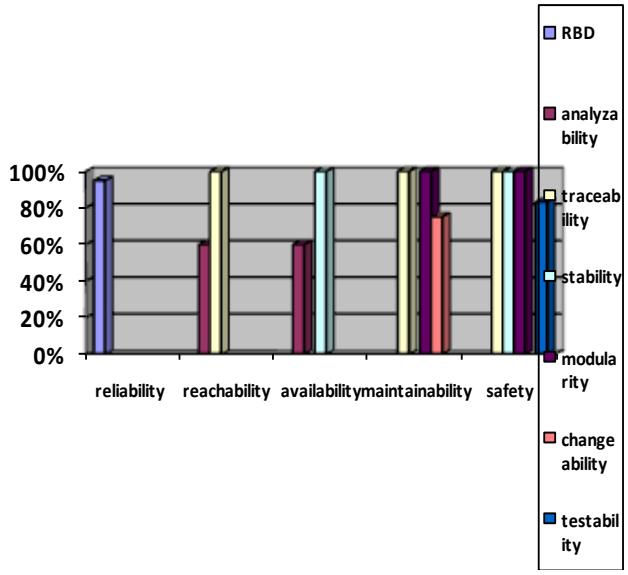


Fig 7(b)

4 CONCLUSION AND FUTURE WORK

Using Model-Based Design, verification and validation activities occur throughout development. A number of new technologies have been introduced that assist with early model verification such as requirements traceability, model checking, model coverage, formal methods, and test case generation. Continuous verification and validation of requirements throughout the design life cycle reduced errors and development time.

The results obtained from the work not only helped in deriving an empirical relationship between the model-based approach and the conventional approach but also highlighted its advantages over the conventional approach. The application of the model based approach in safety critical domain has proven to be effective and can be extended to more critical functionalities in the domain. The same approach can be implemented at design level which encourages V&V at the top most level of the design life cycle, thus ensuring correctness of the system right at the start of the life cycle. Also there are other commercially available tools that support model based development involving formal techniques apart from Mathworks. The other available tools can also be used for implementing the approach and a comparative analysis of tools can be done in order to find an effective tool for the particular application.

ACKNOWLEDGMENT

We acknowledge Director CSIR-NAL to carry out this work.

REFERENCES

1. CONTROL ALGORITHM MODELING GUIDELINES USING MATLAB®, Simulink®, and Stateflow® Version 2.1 MathWorks Automotive Advisory Board (MAAB) July 27th, 2007.
2. "Comparison of software metrics tools" by Rüdiger Lincke, Jonas Lundberg and Welf Löwe. Software Technology Group, School of Mathematics and Systems Engineering, Växjö University, Sweden.
3. "Measuring productivity and quality in model based design": excerpt from MATLAB digest; March 2006
4. "Model Based Design for DO178B with qualified tools" by Tom Errkinen and Bill Potter; Mathworks Inc.
5. "A new approach to system reliability" by Gopal Chaudhri, Kuolong Hu and Nader Afshar. (IEEE transactions on Reliability : March 2001)
6. "Quantifying the analyzability of Software Architectures" by Eric Bouwers, Jos´e Pedro Correia, Arie van Deursen and Joost Visser (Delft University of technology ,Delft , Netherlands)
7. www.arisa.se/compendium/analyzability
8. IEEE standards glossary of software engineering technology
9. "On traceability for safety critical systems engineering" by Dr Paul Mason (Shinawatra University, Thailand)
10. "Creating Safety Requirements Traceability for Assuring and Recertifying Legacy Safety-Critical Systems" by Janice Hill and Scott Tilley (IEEE International Requirement Engineering Conference 2010)
11. "Modularity in Design of Products and Systems" by Chun-Che Huang and Andrew Kusiak (IEEE 1998)
12. Using Software Architecture Techniques to Support the Modular Certification of Safety Critical Systems" by Tim Kelly (University of York, UK)
13. "Meaning , Memory and Modularity" by Stephen Crain , Weijia Ni , Donald Shankweiler , Laura Conway and David Braze (University of Maryland and University of Connecticut)
14. "Measuring Software Design Modularity" by Yuanfang Cai and Sunny Huynh (Drexel University , Philadelphia)
15. "Defining Changeability: Reconciling Flexibility, Adaptability ,Scalability, Modifiability and Robustness for maintaining Systems Lifecycle value" by Adam M Ross , Donna H Rhodes and Daniel E Hastings (Massachusetts Institute of technology,Cambrige)

16. "Stability Monitoring and Analysis of Learning in Adaptive Systems" by Edgar Fuller, Bojan Cukic,Martin Mladenovski and Sampath Yerramalla (West Virginia University)