International Journal of Advanced Information Science and Technology (IJAIST) ISSN: 2319:268
Vol.2, No.7, July 2013 DOI:10.15693/ijaist/2013.v2i7.29-32

Non-Real Time Testing: An Addition to Testing Process

Anshuman Singh[1], Rachit Shrivastava[2], Vairamuthu S.[3]
[1 2] B.Tech, School of Computer Science and Engineering, VIT University, Vellore, Tamil Nadu, India
[3] Assistant Professor (Sr.), School of Computer Science and Engineering, VIT University, Vellore, Tamil Nadu, India

Abstract **This paper introduces the concept of Non-Real Time (NRT) Testing, and attempts to describe why a new methodology like it was needed and was introduced. It describes the situations in which Non-Real Time Testing is preferable to Real-Time Testing and enumerates its advantages over other methods of testing. The challenges faced during the process of testing of a Flight Control System's Control Laws are also described in this paper. Due to the high costs and resources involved in testing of the Control Laws as part of the flight development program, NRT Testing was introduced to augment the verification and validation of the Digital Flight Control Laws at different stages of the testing process.**

Keywords: Testing, Non-Real Time, Safety Critical System, Ada

1. INTRODUCTION

Any system software is developed in certain steps. Development of system's software requires certain procedure or a process structure to be followed which is commonly known as Software Development Life-Cycle (SDLC). Testing is considered as an important step in the software development cycle as it helps in the validation of the final product/system. It makes the product more robust by determining the running conditions; behavior of the system upon certain user action and check the functionality of the system. Software testing can be considered as a process of verifying and validating the system if it is

- able to meet through the system requirements,
- working as expected,
- gives desired output for some user input,
- correctly designed as per the stakeholder's viewpoint

Modern day fighter aircrafts have to be highly agile in order to compete with the other fighter aircrafts of today. This agility, flexibility and maneuverability can be achieved with making the aircraft deliberately unstable and having a digital fly by wire flight control system on board which controls the stability of the aircraft. The flight control system uses the Digital Flight Control Law scheduled at every flight condition. Due to the real time requirement of the system, the flight control software goes through extensive verification and validation for a fighter aircraft program. The software goes through Unit Level Testing, Integration testing and Hardware Software Integration. After these testing procedures, the software is again tested on ground in the facilities called "MiniBird" and "IronBird".

IronBird is a ground based testing facility which is used to validate vital aircraft systems including the flight controls. All of the aircraft systems are physically integrated and each of them is laid out according to the actual configuration of the aircraft and all components are also installed on the same place as they would on the real airframe. It also has the facility to record the flight data for subsequent analysis and treatment of failures introduced in the systems. MiniBird is a less extensive form of IronBird where although flight worthy actuators are used, they are not necessarily physically oriented according to the actual configuration of the aircraft. Also, in MiniBird inspite of using two separate actuators to indicate the left and right elevons, only one actuator is used to check for functionality. They are hardware-in-loop test facilities where the real actuators, flight model and flight computer are tested. IronBird operates in a closed loop mode whereas MiniBird operates in an open loop mode.

2. WHAT IS NRT TESTING?

In Software Development Process, testing is mainly done after the design phase. Testing helps to validate the product by determining whether the system works as per the requirements and accordingly generates output. Testing has various types, some test for security, some for regression analysis, some for functional and non-functional requirements, some for performance, etc. Non-Real Time Testing can be considered as a type of testing which is used to test a system thereby reducing the cost involved in testing it in the actual environment. NRT testing is basically used for safety critical systems in which testing in the real environment in real-time may involve a lot of cost, when the same can be achieved by testing it on a cross-compiler.

A perfect example for this can be considered as the NRT testing of a Safety Critical Ada code which reduces the cost involved at testing the Ada code at the MiniBird and IronBird levels, moreover such testing also checks for minor test cases at the developers end so as to meet the requirements.

NRT Testing can be grouped under Unit Testing category as it refers to those tests that verify the functionality of a specific section of code; usually at the function level. NRT Testing uses the gray-box testing approach where the testers have some knowledge about the system and hence preparation of test cases can be done accordingly.

3. NEED FOR NRT TESTING

Safety critical systems should not give high errors at the final levels of testing in their actual environments. Questions are raised whether the systems have been coded correctly as per the required functionality, thus introducing the concept of testing the code at local level on a cross-compiler simulating real-time environment and conditions while being non real-time. NRT focuses on stressing the software to its limits i.e. each and every condition is tested covering all the boundary values. NRT Testing is introduced to those systems dealing with time constraints and where the occurrence of any error at the final level of testing is not permitted (e.g. safety critical systems). NRT Testing reduces the chances of failure of the code at the final levels by detecting the errors at the intermediate levels.

4. ADVANTAGES OF PERFORMING NRT TESTING

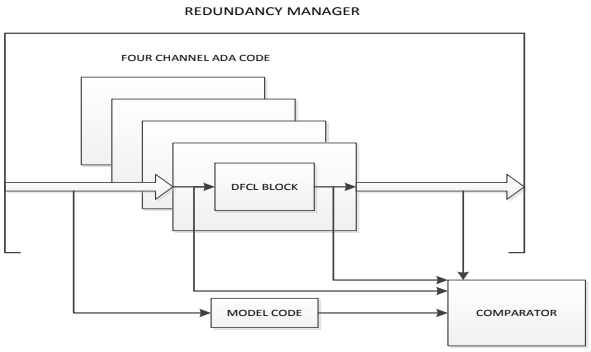
Introducing NRT Testing at the intermediate levels of software development cycle has some following mentioned advantages:

- The functional behavior and performance of the system is tested. The testability is increased as the user is able to tap out intermediate values.
- Since the test cases are based on knowledge of the system, each block has a specific number of tests associated with it and the types of input to the block is finite.
- The effect of parameters on final system performance is easy to analyze. The extent to which an error can be adjusted with can be answered which avoids costly system changes.
- NRT Testing can be started early on in the software development life cycle which saves time and cost to the project.
- NRT Testing provides the advantage during the other phases as it narrows down the places where to look for bugs.

5. EXPERIENCE/APPLICABILITY

In order to facilitate better development of CLAW code, the following points can be taken into consideration:

- Always test for transient behavior as static tests don't give much information about the system. By transit behavior, it is meant that values of signals change as time changes.
- Even though a good SQA (Software Quality Assurance) plan is followed along with good compilers and auto code generators, extensive testing is a must to ensure proper software development.



A SCHEMATIC OF THE NRT PHILOSOPHY

Fig 1 shows the schematic diagram for the concept of NRT Testing.

The concept involves testing only a part of the code on a single target machine. Only the required Ada Code is clipped and stubbed. An Ada driver is added to the clipped Ada code which helps to accept the flight input parameters and taps out the required outputs for CLAW functionality. This code with the Driver file is compiled with the same compiler and settings as the final build of the actual Flight Code.

The project involved certain steps to be taken in order to perform NRT Testing over the flight code of the aircraft. The steps have been described below:

Implementation of MATLAB driver file for running the Simulink model of the FCS

Based on the requirements, a Simulink model had been developed which is essentially the same as the Ada code which is used in the fighter aircraft. The driver file for the Simulink model provides the input parameters, initializes the parameters and provides a framework for the way in which the output is generated. Various test cases have been prepared which initialize different values to different parameters in order to check the full functionality of the CLAW module.

Implementation of Ada driver file for running the Ada code used in the aircraft

The responsibility of the Ada driver file is to take the same inputs for particular test case that are also fed into the Simulink model, process the result from the Ada code of the Fighter Aircraft. The driver file consists of the file; read and write operations so as to have all the values of input parameters assigned to the Ada variables and to generate the result in a particular orderly manner. A specification file relates all the input parameters to some dummy variables before assigning them to actual variables of Ada code. The driver file acts as an interface between the test case inputs and the variables for the Ada code. The Ada code is stubbed in order to successfully assign these variables and hence run the test cases. This code with the driver is compiled with the same compiler and settings as the final build of the critical system software.

Analyzing outputs generated and finding cause of discrepancies between them

After getting the outputs from the Simulink model and Ada code, the values of the parameters of interest are checked manually to see if there are any differences. If differences are found, the fault is in the driver file of either the model or the Ada code.

Generation of utility to compare the outputs graphically

Due to the high number of parameters that had to be checked, the approach of comparing them manually was not feasible. A utility was created with the help of MATLAB which takes the signal values from both the Simulink model and the Ada code and compares them graphically. Also, it points out the points on the graph where the signal values are not matching. The utility is useful to find out the points of difference without any need of manually going through the values. A sample graphical comparison can be seen from Figure 2.

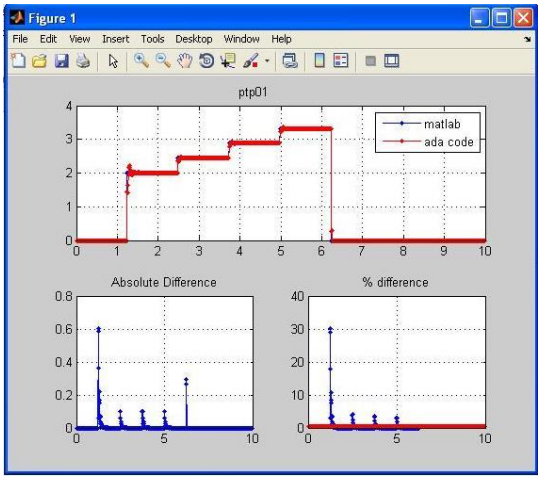


Fig 2: Graphical Comparison of a Flight Parameter (ptp01)

6. CONCLUSION

NRT Testing was successfully used to test the safety critical Ada code against the Simulink model. To conclude, we can point out a number of reasons as to why non real time testing was preferred against real time testing.

NRT testing can be done on any computer as long as it has got all the correct compilers and simulators. IronBird and MiniBird facilities are limited whilst NRT testing can be done on virtually any number of machines. This saves on the resources used to operate IronBird and MiniBird. Not only are resources saved, more extensive testing can be done before the MiniBird and IronBird level so as to ensure code is performing as per the requirement. With NRT Testing, it is relatively easier to tap out intermediate values which increase the testability.

Also, this can be started early on in the SDLC leading to savings in cost and time in the long run.

7. ACKNKOWLEDGEMENT

Special thanks to Aeronautical Development Establishment - Defense Research & Development Organization, Bangalore for providing such an environment where we were able to complete our project. We would like to thank our ADE Project Guide Ms. Asha Garg (Sc-'G', Head, Software Engineering Division – Light Combat Aircraft) for her guidance on this project. We would also like to thank all those scientists and engineers of ADE, Bangalore who constantly provided us their help regarding the project, in understanding the working environment and some terms and techniques of testing.

8. REFERENCE

[1] Srinivasan Desikan, Gopalaswamy Ramesh - *Software Testing: Principles & Practices*, Pearson Publication, Sixth Impression 2006

[2] Boris Beizer - *Software Testing Techniques*, DreamTech Press, Second Edition (2009)

[3] Richard Riehle - *Ada Distilled: An Introduction to Ada Programming for Experienced Computer Programmers*, AdaWorks Software Engineering, July 2003

[4] Steven R Rakitin - *Software Verification and Validation – A Practitioner's Guide*, Artech House (1997)

[5] Y. V. Jeppu, K. Karunakar, P. S. Subramanyam "Testing Safety Critical Ada Code Using Non Real Time Testing" - *Lecture Notes in Computer Science Volume 2655*, 2003, pp 382-393, ISBN - 978-3-540-40376-0