Launch Vehicle Software System Reliability Prediction based Architecture and Information Chain

Tinggui Yan [1, a], Xiaoqian Chen [1,a], Jian Bai [2, b], Li Ma [2, b], Zhifang Yang [2, b]

[1] College of Aerospace Science and Engineering, National University of Defense Technology, 410073, Deya Road, KaifuDistrict, Changsha,China

[2] Beijing Institute of Astronautical System Engineering, China Academy of Launch vehicle Technology, 100076 ,No21,10branch of 9200mailbox,Beijing,China

[a]email: qwyzfang@gmail.com, [b]email:qwyzf@163.com

Keywords: Launch Vehicle; reliability model; software testing; Information Chain

Abstract. Launch Vehicle software is the important factor in the Launch Vehicle system. Thus, a well-designed Launch Vehicle software is an important factor for Space system quality while the reliability of Launch Vehicle software is a relatively difficult research field. Existing software reliability assessment techniques based on particular assumptions and preconditions on probability distribution of cumulative number of failures, failure data observed, and form of the failure intensity function, and so on. Quality in Launch Vehicle, however, is faced with many challenges, due to the immense number of event and state interactions. In this paper, it introduces a model of Launch Vehicle software reliability, discusses its issues encountered in the modeling process. In the end, a simple case introduced guided by Launch Vehicle software Architecture and Information Chain.

Introduction

Launch Vehicle software is becoming increasingly important in Space system, a well- designed Launch Vehicle software is an important factor for Space system quality.

Many researchers have already done a lot of meaningful work on software quality. White et al [2, 3] and Belli [4] pointed that various responsibilities of the user can be specified as a complete interaction sequence (CIS) between the user and the software application under testing. A.T.Memon et al [5,6,7] proposed the model of event flow graph (EFG) models in the software testing and several approaches guiding the GUI testing, such as usage profiles, dynamic adaptive automated test generation [6] and incorporating event context.

In the test strategy , H.Hu, K.Y.Cai et al. [8,9],introduced a mechanism to handle multiple testing tasks on the same or different software under tests (SUTs) for shorten the testing time and reduce cost. Then, they proposed an extended adaptive testing strategy, namely Modified Adaptive Testing (MAT). The use of test history information allows the resulting test process to be adaptive in the selection of tests under a limited test budget. Based on this strategy, in the work of Z.F.Yang [10], an approach of software testing Guided by Bayesian model was introduced, which can guide the software testing and find more defects as soon as possible. Furthermore, in the work of L.Zhao[11],he introduce a new model which take advantage relationship of Events and code, develop a corresponding testing framework of software.

In the Launch Vehicle software testing, the above mentioned work are effective, and improve reliability of Launch Vehicle software, considering that one of the primary goals of testing is to improve reliability, since reliability is a user-centric measure[1]. In spite of the situation, the reliability of Launch Vehicle software is a relatively young research field. Existing software reliability assessment techniques attempt to statistically describe the software testing process and to determine and thus predict the reliability of the system under consideration, which based on particular assumptions and preconditions on probability distribution of cumulative number of failures, failure data observed, and form of the failure intensity function, etc. However, Launch Vehicle software reliability assessment faced several problems as follow:

First, Launch Vehicle software test cases generation and perform is complicated and costly, Test data scarcity requirements reliability assessment techniques will gain effective evaluation Even in some failure data is missing.

Second, based on the Interaction of Launch Vehicle software Event, the profile of Launch Vehicle software is quite complicated. Considering the state of Launch Vehicle software event, how to predict the affection of testing profile for the system of Launch Vehicle software is an important issue in Launch Vehicle software reliability assessment.

The last point is the most important, compared with traditional reliability assessment, which treat the software as a monolithic whole, considering only its interactions with external environment, without an attempt to model internal structure. In the Launch Vehicle software testing, it can gain a wealth of information based on Launch Vehicle software structure and Information Chain. Given the Launch Vehicle software architecture, which can be adopted to guide the testing process and establish confidence assessment of Launch Vehicle software.

The aim of this paper is to provide an overview of the architecture and Information Chain based on Launch Vehicle software approaches to quantitative assessment of software systems. The rest of the paper is organized as follows. The background is introduced in Section2. The models of Launch Vehicle software reliability assessment are described in detail in Section 3. The analysis, construction process and a case study of model are discussed in Section 4.

BACKGROUND

In the following, Launch Vehicle software and related test frameworks used for performing the experiments are outlined.

A Launch Vehicle software is a special software of Launch Vehicle system. Due to the various components of the Launch Vehicle, Launch Vehicle software shows complicated Architecture and Information Chain. In this paper, a subject programs are adopted for validation, that is, Launch Vehicle Telemetry Data Real-time Processing Software.

Telemetry Data Real-time Processing Software is an example program alone with Borland C++ Builder 6. This program is coded with C++ using Visual Component Library and contains about 10532 lines of code. The number of seeded faults relevant to this study seeded in these software are 116. In this study, the application should have passed a series of testing procedures before release, and thus “obvious” bugs should be eliminated. In this case, a static analysis tool, i.e., FindBugs, is adopted to verify that none of the reported bugs in this study can be detected by FindBugs.

Select three important modules of the software above, it is database system(Ds), Input / output system(I/Os), graphical user interface system(GUIs). The event and state interactions of three modules above contribute to Architecture and Information Chain.

Architecture and Information Chain Interaction Graph

In this paper, defined Architecture and Information Chain Interaction Graph represents real Architecture and Information Chain as follow:

$$AICIG = <H,W,E> \tag{1}$$

Where H is a set of events in software, each $h \in H$ represents a event in software, H={h}, W is a set of modules in software, each $w \in H$ represents an type of module in the graph, W ={w},and E is a set of edges in software, each $e \in E$ represents an edge in the graph, E={e}.

Then we define the node “w_i”.The model of node “w_i” represents the type of module in the graph and defined as $<W_i,RW_i,EW_i>$.where:

W_i is the i^{th} type module in the graph, RW_i is the reliability of Event based the i^{th} type module in the graph, and EW_i is its average execution time of the i^{th} type module in the graph, note that this time is how many times the event execution in process.

A directed edge e represents the execution path from one window to another and is defined by $<T_{ij},RT_{ij},PT_{ij}>$ where:

T_{ij} is the transition from node n_i to n_j ,RT_{ij} is the transition reliability, and PT_{ij} is the transition probability.

Now, we define the Architecture and Information Chain Interaction Graph for Launch Vehicle software reliability assessment showing in fig1.

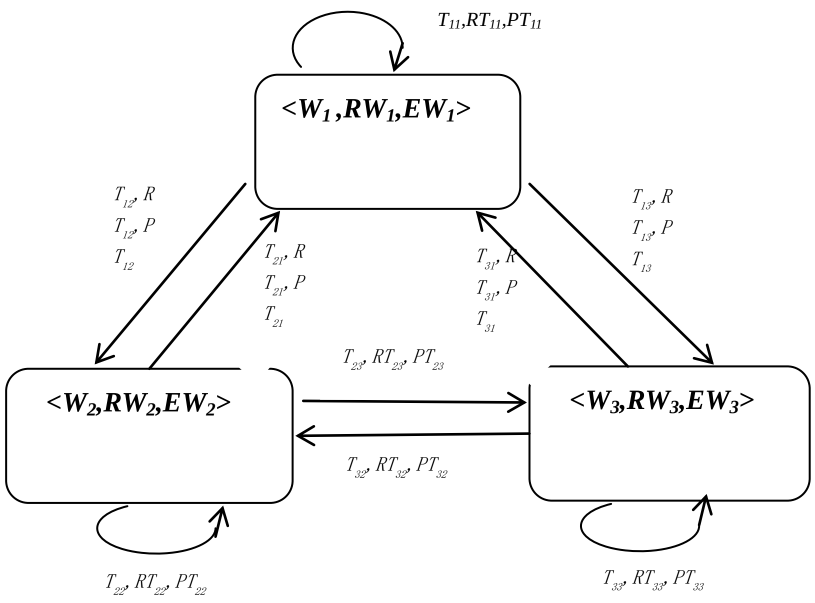

Fig.1 A real Architecture and Information Chain Interaction Graph for reliability assessment

Considering the state of software, all event have new states once an event is triggered. The changes of states of the Launch Vehicle software are called transitions, and the probabilities associated with various state changes are called transition probabilities, defined as PT_{ij} above. The process is characterized by a state space, a transition matrix describing the probabilities of transitions, and an initial state (or initial distribution) across the state space, just as Markov chain [12].

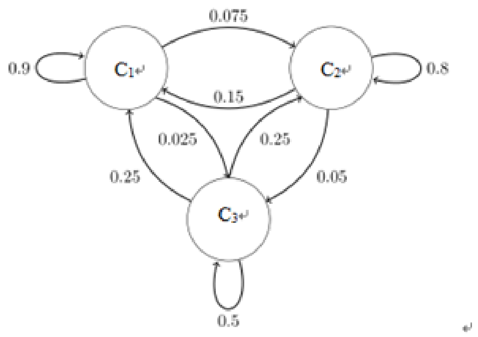

Fig.2 Launch Vehicle software states changes diagram for a simple example

Launch Vehicle software is in a certain state at a specific time and its state changes randomly within the profile. The Launch Vehicle software states that the conditional probability distribution for the system at the next step depends on the current state of the system. Since the Launch Vehicle software state changes randomly, a state diagram for a simple example is shown in the Fig. 1, using a directed graph to demonstrate the state transitions. The states represent the transition probabilities from the ith state to the i+1th state. Labelling the state space {C_1, C_2, C_3}, the transition matrix for this example is shown as follow:

$$P_{ij} = \begin{pmatrix} 0.9 & 0.075 & 0.025 \\ 0.15 & 0.8 & 0.05 \\ 0.25 & 0.25 & 0.5 \end{pmatrix} \quad (2)$$

Where, the sum of each row is 1.

Reliability assessment process

Step1. Calculate the probability of execution of each module by estimating the frequency of execution of each window，Estimate the reliability of windows (RW_i) and its average execution time (EW_i)

Step2.   Calculate the transition probability from one window to another for all functions (PT_{ij}) Estimate the transition reliability (RT_{ij}).

Step3. For each function, calculate the execution time EW_i ,build the profile of each function.

Step4. Construct the *AICIG* according to the Fig.1.

Step5. Assess the reliability of Launch Vehicle software according to the formula 3

$$R=\sum_{i=1}^{k}\sum_{j=1}^{k} RW_i * RT_{ij} * PT_{ij} * RW_j * (EW_i + EW_j) / \sum_{i=1}^{k} kEW_i \qquad (2)$$

Experimental study

In this paper, Both software above includes three module introduced in Section2. Case study as follow:

Failure detection rate of three module show in Fig. 3, it is found that the 3-rd type event has the highest failure detection rate, while the 1-st type event has the lowest failure detection rate. Apparently, the 3-rd type event handler should be paid more attention in advance to improving the reliability of Launch Vehicle software.

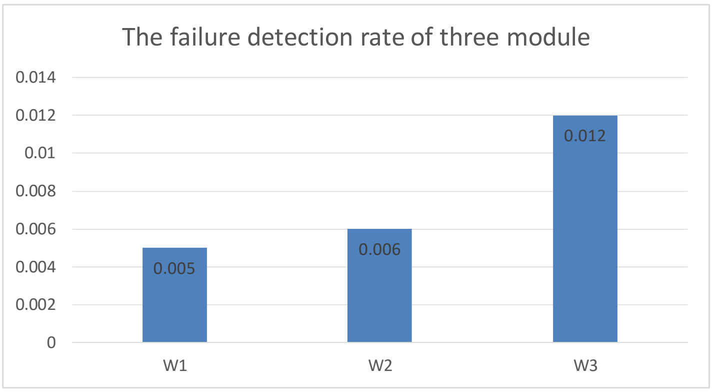

Fig.3 Failure detection rate of three module

The variation in the reliability of transitions (between different windows) shows as fig.4. The reliability of transition (one at a time) is from 0 to 1, while the reliabilities of other components and transitions are fixed (equal to 1 for the sake of comparison).

From Fig. 4, transition reliabilities can significantly affect the reliability of Launch Vehicle software. For example, the transition reliability W_2-W_2 and W_2-W_3 can significantly deteriorate the reliability of Launch Vehicle software if there are mismatches or errors in the Event interaction. This is due to the nature of Launch Vehicle software that error often found in event interaction.

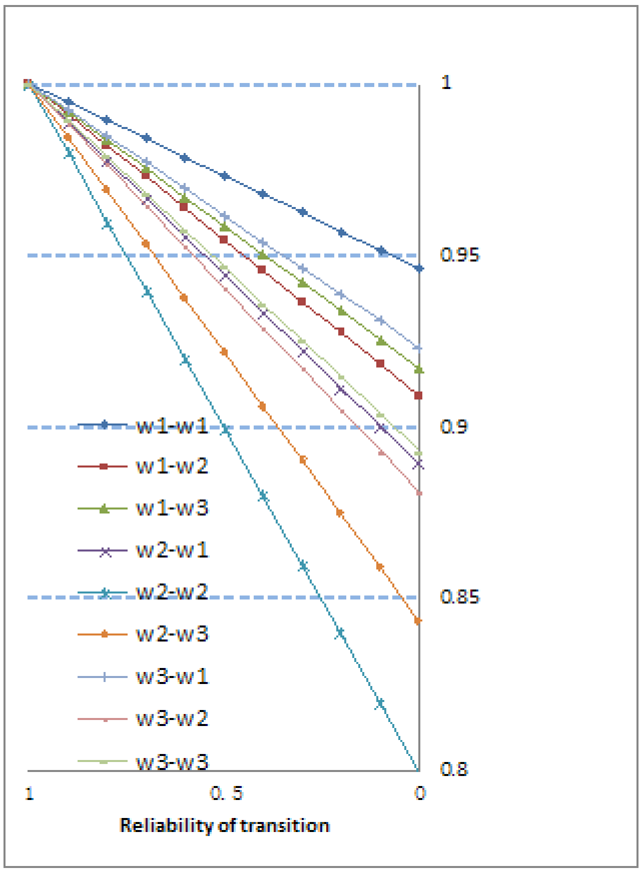

Fig.4 the reliability of transition profile shown in table.1. (one at a time)

The reliability of Launch Vehicle software: the reliability of Launch Vehicle software is derived by combning the structual of information chain. The Transition matrix is:

$$M=\begin{pmatrix} 0.20 & 0.35 & 0.45 \\ 0.37 & 0.44 & 0.19 \\ 0.29 & 0.34 & 0.37 \end{pmatrix}$$

According to function profile, operational profile and transition matrix, the reliability of Launch Vehicle software will be calculated as R=0.988

Conclusion

Summary, this paper describes an effective reliability model for Launch Vehicle software, discussed the reliability model structure and its issues encountered in the modeling process. Facing the problem of the Launch Vehicle software, we adopted Launch Vehicle software reliability assessment based on AICIG to guide the Launch Vehicle software reliability assessment process. In the end, a simple case verifies the validity of the model during the Launch Vehicle software reliability assessment process.

References

[1] F.Belli, M.Beyazit, N.Güler, Event-Based GUI Testing and Reliability Assessment Techniques -- An Experimental Insight and Preliminary Results, 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops[C] ,2011.

[2] L.White, H.Almezen, Generating Test Cases for GUI Responsibilities Using Complete Interaction Sequences, Proc.the 11th International Symposium on Software Reliability Engineering[C],2000

[3] L.White, H.Almezen, N.Alzeidi, User-Based Testing of GUI Sequences and Their Interactions,

Proc. the 12th International Symposium on Software Reliability Engineering[C] ,2001.

[4] F.Belli, Finite State Testing and Analysis of Graphical User Interfaces, Proc. the 12th International Symposium on Software Reliability Engineering [C],2001.

[5] P.A. Brooks, A.M.Memon, Automated GUI Testing Guided by Usage Profiles, Proc. the twenty-second IEEE/ACM international conference on Automated software engineering [C],2007

[6] X.Yuan, M.B.Cohen, A.M.Memon, Towards Dynamic Adaptive Automated Test Generation for Graphical User Interfaces. IEEE International Conference on Software Testing, Verification, and Validation Workshops[C],2009.

[7] X.Yuan, M.B.Cohen, A.M.Memon, GUI Interaction Testing: Incorporating Event Context [J].IEEE Transactions on Software Engineering. 2011(4): 559-574

[8] H.Hu, K.Y.Cai etc. A Parallel Implementation Strategy of Adaptive Testing, Computer Software and Applications Conference Workshops (COMPSACW), 2010

[9] H.Hu, K.Y.Cai etc. Enhancing software reliability estimates using modified adaptive testing. Special Section: Component-Based Software Engineering (CBSE)[J], 2011(55) 2:288–300

[10] Z.F.YANG, Z.X.YU,C.G.BAI, The approach of graphical user interface testing guided by bayesian model , Proc. the 2013 International Conference on Computer Engineering and Network [C] ,2013

[11] L.Zhao,GUI Software Testing based on Event Handlers[D] Beijing:Beihang University,2010

[12] http://en.wikipedia.org/wiki/Markov_chain