Simulation-Based Boundary Testing of Software with Its Applications

HU Shaolin[1, a], KARL Meinke[2,b] and WANG Xinfeng[1,c]

[1] Key Laboratory of Spacecraft Fault Diagnosis and Maintenance, Xi'an,P.O.Box 508-17,China

[2] Royal Institute of Technology, Stockholm, Sweden

[a]hfkth@126.com, [b]kmeinke@nada.kth.se，[c]huf@mail.xjtu.edu.cn

Keywords: Software Testing; Boundary Testing; Model Based Approach.

Abstract. To test the boundary values of input variables is very useful but difficult when designing and debugging a software system for scientific computations. In this paper, the model-based method and the simulation-based method are integrated to form a practical approach to deal with this troublesome problem of boundary testing. After selectively building a model library, the testing algorithms are designed in detail are used in designing and evaluating the systems for exterior tracking and post processing of measurement data.

Introduction

Recently, software testing technologies have been widely used to achieve reliability and validity of software systems [1,2]. As one of the most important testing approaches, boundary testing, which mainly focuses on the boundary as well as limit conditions of the software being tested, can be used in or integrated into the testing procedure of software systems.

Boundary testing is to test find out boundary values of input and output of software. In order words, the software boundary testing is the process of detecting and localizing salient boundaries of multi-input data series as well as multi-output results. Boundary testing is very important to make sure the software safely running. Generally, test cases are generated using the extremes of the input domain, e.g. maximum, minimum, just inside/outside boundaries, typical values, and error values [3]. It is similar to equivalence partitioning but focuses on "corner cases"[4,5].

In this paper, we investigate how to design a practical boundary test approach. By integrating the model-based method and the simulation-based method into a testing framework, a new and practical test procedure is set up in section 2. In section 3, an example is analyzed, which shows that this test algorithm can be directly used to validate the exterior tracking and measurement data post processing (ETMPP for short) system.

Model Based Simulation Evaluation

In order to test the boundary of input variables and to evaluate the correctness and precision of computational software, we assume that the input variables and output variables are $X \in R^p$ and $Y \in R^q$ respectively and that the system is a black box [6], as illustrated in Fig 1.

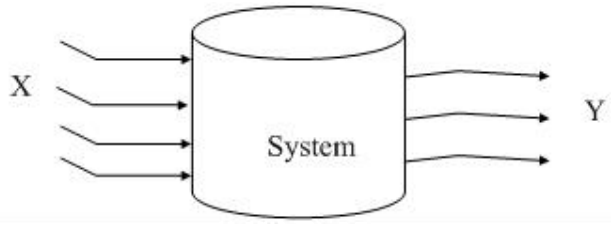

Fig. 1 Description of Software Model

This input-output model described in Fig 1 can be abstracted as the following network model in Fig 2, which has three layers: the input node layer, the output node layer and the transact node layer.

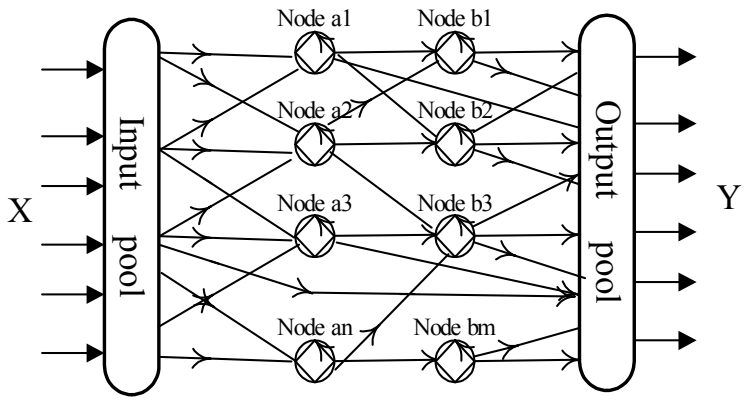

Fig.2 Network Model

Considering all kinds of computing errors from data processing when the software is used, we may set up the following mathematical model:

$$Y = f(X,\theta) + \varepsilon \tag{1}$$

where the stochastic variable ε is the modeling error.

Assuming that there are n normal samples, which are computed using the software, different inputs were given. From reference [7], if a library of models Θ is selectively built to approximate the network model described in Fig 2, then the identification algorithm of parameter θ can be obtained as follows

$$(\hat{f},\hat{\theta}) = \arg\min_{\theta\in R^m}\left\{\min_{f\in\Theta}\sum_{i=1}^{n}\{\|Y_i - f(X_i,\theta)\|\}\right\} \tag{2}$$

So, we get the best estimator $\hat{\theta}$ of the model parameters θ and the best fitted model $\hat{f}$ with parameters $\hat{\theta}$ as well as summation of the fitted residual of the model as

$$Rss = \sum_{i=1}^{n}\{\|Y_i - \hat{f}(X_i,\hat{\theta})\|\} \tag{3}$$

Based on this selectively fitted model as well as the fitting residual, we can set up the following procedure to test the software:

Step1: Computation of prediction output: setting input values X_{n+s}, we use a selectively model described in equation (2) from the library Θ to compute theoretical output:

$$\hat{Y}_{n+s} = \hat{f}(X_{n+s},\hat{\theta}) \tag{4}$$

Step2: Real output from software computation: putting the same input values X_{n+s} into the software to be tested, we get a series of practical output Y_{n+s};

Step3: Residual generation: denote the difference between the practical and the predictive output by

$$\hat{\varepsilon}_{n+s} = Y_{n+s} - \hat{f}(X_{n+s},\hat{\theta}) \tag{5}$$

Step4: Judgment: if $\|\hat{\varepsilon}_{n+s}\|^2 \le 9\cdot RSS/n$ the input values X_{n+s} are reasonable; otherwise, the input values X_{n+s} is beyond the boundary of input variables in this software.

Step5: Modification of the estimators of edges: If the input X_{n+s} is normal, we extrapolate the edge of input in the software and set a modified estimator of boundary values as follows

$$\hat{X}_b = \hat{X}_{n+s} + (\hat{X}_{n+s} - \hat{X}_n)/2 \tag{6}$$

Otherwise, we interpolate and estimate the boundary value as follows

$$\hat{X}_b = \hat{X}_n + (\hat{X}_{n+s} - \hat{X}_n)/2 \tag{7}$$

Step6: Set $\hat{X}_b \Rightarrow \hat{X}_{n+s}$ and go return to step 1) and do steps 1) to 5) repeatedly till the renewed $\hat{X}_b$ is approximately equal to the previous $\hat{X}_b$ one step behind.

As for the model library Θ, we may set the library as an open frame and firstly put some typical models into the library, such as the linear regression model, the non-linear model, the time series model[5] (AR, MA, ARMA, ARX, ARXMA, etc), the polynomial model and the dynamic-measurement model[8] for continuous variable dynamic system and the Petri net model, the automaton model, and Markov chain model for discrete event dynamic model. When the library being used, open frame can be helpful to update and to add new models into the library Θ on-line.

Application to Test the ETMDPPS Software

The ETMDPPS system is the practical computation software used to determine the trajectories of spacecraft. The ETMDPPS system is composed of five parts: all three databases which provide the system with all kinds of tracking & measurement data, data preprocessor and data cleaner which are used to interpolate missing data and to modify outliers in usable sampling series, correction and filtering modules which are used to correct all kind of systematic errors and to smooth random errors in sampling series, trajectories computation modules which are used to compute location as well as velocity parameters of rocket piece to piece, joining & precision analysis modules which can be used to link up all pieces of trajectories and to analyze the precision of the integrated trajectory.

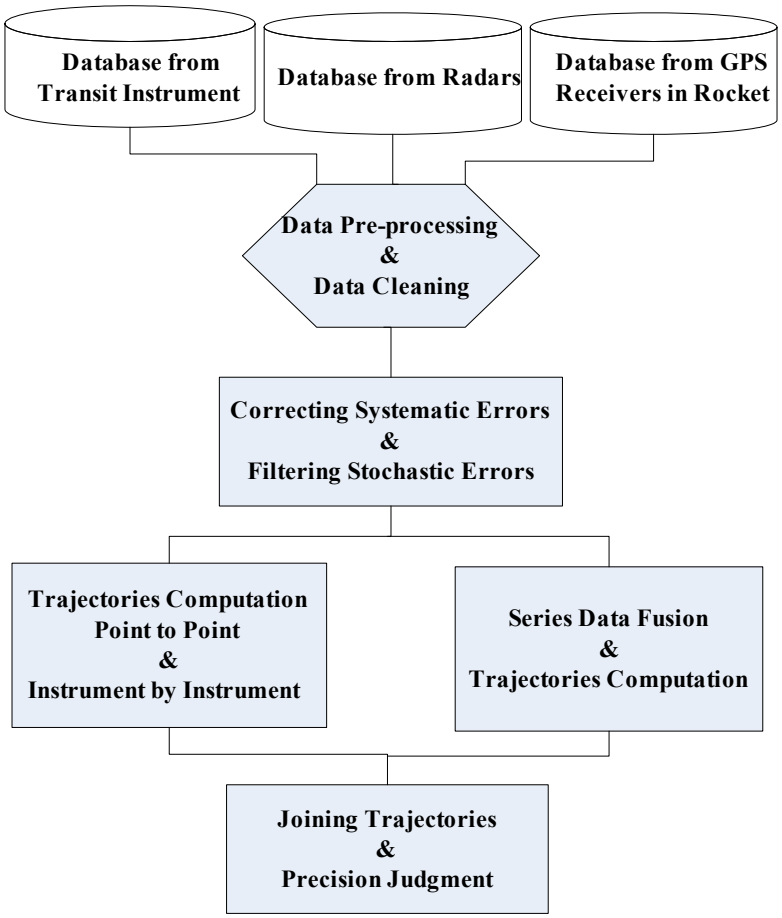

Fig. 3 Structure of the Software ETMDPPS

In order to validate this software, we design two different kinds of new simulation based testing and evaluation scenario based on the procedure described in section 2. These two procedures are used

to evaluate functions of four subsystems and boundaries of input variables. Simulation results show that the testing algorithm described in section 2 is valid and usable.

Acknowledgements

This work was financially supported by the National Nature Science Foundation of China (61473222) and the Sweden Institute Grant (SI 05483/2005-210). The first author would like to thank Dean Ingrid Melinder of the Computer Science and Communication (CSC) of the Royal Institute of Technology (KTH) for her friendly help when he visited the CSC of KTH of Sweden.

References

[1] Cem Kaner, et al. *Testing Computer Software*. NY:Van Nostrand Rienhold, 1993.

[2] McCarthy Jim. *Dynamics of Software Development*. WA:Microsoft Press, 1995.

[3] Ian R McDonald. *Black Box Test Techniques-Equivalence Partitioning and Boundary Value Analysis*. http://slideshare.net, 2013

[4] Software Testing Glossary, *Software Testing and Quality Assurance Glossary*. http://aptest.com, 2013

[5] Murnane,K Reed,R Hall.*On the Learnability of Two Representations of Equivalence Partitioning and Boundary Value Analysis*, in Software Engineering Conference,ASWEC2007,18th Australian, 2007,pp:274-283

[6] Meinke Karl. *Automated Black-Box Testing of Functional Correctness Using Function Approximation*. Proc ACM SIGSOFT Int. Symp on Software Testing and Analysis, ISSTA 2004.

[7] K Romanik. *Approximate Testing and Its Relationship to Learning. Theoretical Computer Science,* 1997,188, pp:79-99

[8] Jianqing Fan,Qiwei Yao. *Nonlinear Time Series-Nonparametric and Parametric Methods*. Springer Science+ Business Media, Inc. 2005,pp:1-373.