A Frame Work on Early Reliability Assessment

Debasis Mohapatra, Sasmita Rani Behera and Ranumayee Sing

Lecturer (Dept. of CSE) P.M.E.C. (*A constituent college of BPUT*), Berhampur, India

Summary

Reliability is the probability of fault-free operation of software over a particular time limit. Most of the work done in this area generally focus on finding the reliability at the later stage, which can answer what is the reliability of the system, but it can't ensure what the reliability should be. Finding reliability is a major task for providing the user appropriate level of services with less fault-rate, so it is mandatory to predict the reliability before implementation and let the system to be modeled accordingly to achieve certain predetermined reliability. This paper focuses on the above mentioned requirement. In this paper we have generated the use case diagram and component diagram for each use cases. The use case diagram represents the sequence of operations and the component diagram represents the involvement of the components to give rise to particular use case. A fault tree is generated that is used to find out the fault rate associated with each component of the software system that leads to failure in the upper-level operation.

Key words:

Use Case, Fault Tree, Composed Tree, Cut-Set.

1. Introduction

Reliability is the probability of successful survival of certain systems for a particular time interval. Most of the research work is based on finding reliability after the implementation, but our concerned objective is based on the priory estimation of reliability by fixing the initial fault-rate. The operations of the system and the accumulation of components to produce certain behavior are represented by use case diagram and component diagram respectively. First the fault rate of the system is determined and by using the fault-tree the fault-rates of all the components are measured, which in turn is used to find out the reliability of all the components. Also we can derive the reliability of each use cases from this.

2. Related work

Boland[1] had proposed to divide the software reliability models into two types: Type 1 describes the model as numbers of faults extracted under time T and Type 2 describes the model as the time gap between two consecutive failures. Man Cheol, Seung and Jaejoo[2] explored the possibilities and corresponding limitations of applying the software reliability growth models to safety-critical software. Xiang and Futatsugi [3] proposed software reliability allocation by using Software Fault-Tree Analysis(SFTA). Ganesh and Joanne [4] proposed a method that combines BBNs (Bayesian Belief Networks) and Fault-Tree to improve software reliability analysis in complex systems. Appukkutty, Ammar, Katerina[5] presented a methodology that assesses software risk at the requirements level using UML specifications of the software at the early development stage. Smith [6] proposed how to annotate different UML diagrams with reliability measures.

3. Annotating Use Case Diagram

A Use Case Diagram describes system behavior and provides a high level view of how the system interacts with external entities (actors)[6,10]. In annotating a Use Case Diagram, we look at the two parameters.

(i) Probability that an actor will use the system denoted by q(i). Where $\sum q(i)=1$.

(ii) Probability of an actor using a selected system behavior, denoted by P(i,x). Where $\sum p(i,x)=1$.

The Probability of system behavior x occurring is given by.

$$p(x)=\sum q(i)*p(i.x)\ldots\ldots\ldots(1)$$

Here we have passed the use case interaction of actors down to the module level for finding module importance I(m).

$$I(m)=(1/k)\sum\theta(m,i)\ldots\ldots\ldots.(2)$$

Where $\theta(m,i)$=Importance of module m according to actor i.
K=Total no. of actors.

And we formulate the utility function as.

$$\text{Max } U=\sum I(m)*R(m)\ldots\ldots..(3)$$

Where R(m)=Reliability of module 'm'.

Manuscript received September 5, 2011
Manuscript revised September 20, 2011

4. Proposed Work

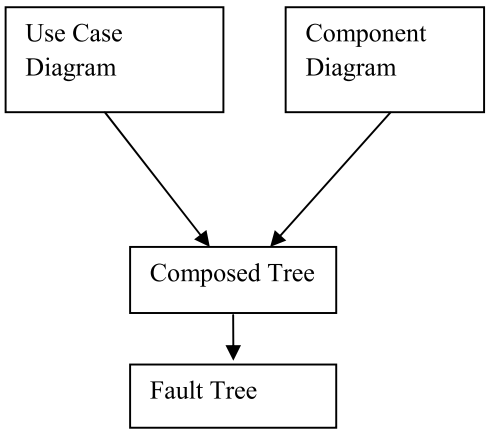

Fig. 1 Overall Structure

The four important tools that are used to assess reliability of each components of the system are Use Case Diagrams, Component Diagrams, Composed tree, Fault tree. The behavioral aspects are traced out from the use cases present in the Use Case Diagram. The Component Diagram provides the view of finding the involvement of components under certain Use Case. The Composed Tree is designed from the Use Case Diagrams and Component Diagrams where the Use Cases represents middle level events, Components represent basic lower level events. The Fault Tree is used to assign the fault rate to all the components according to their involvement under particular Use Case.

4.1 Use Case Diagram

A Use Case Diagram shows a set of Use Cases and actors and their relationships. Use Case Diagram addresses static use case view of a system. Each Use Case in the Use Case Diagram represents certain sequence of operations.

4.2 Component Diagram

A component diagram shows the organization and dependencies among a set of components. Component diagrams address the static implementation view of a system.

4.3 Composed -Tree

Composed Tree is formed by taking Use Cases and Components as the events. Here, each upper level event is considered as successful only when the successful executions of the lower level events take place. The leaf nodes of the tree are filled with successful execution of components.

There are four major steps to form Composed - Tree.

- Define the system, its boundaries, and the top event.
- Construct the composed tree, which symbolically represents the system and its relevant events.
- Perform a qualitative evaluation by identifying those combinations of events that will cause the top event.

Here Composed –Tree is used to provide ease in design the fault tree . Because composed tree is the representation of successful execution from where it is easy to find out the fault in component level .

4.4 Fault -Tree

Fault Tree is based on allocating the fault rate to each component present in the system. Fault Tree Analysis is required to analyze the fault rate of each component. Fault-Tree is the reverse of the Composed –Tree.

Two major steps to form Fault-tree.

- Each node is represented with unsuccessful execution of event.
- Perform a quantitative evaluation by assigning failure probabilities or unavailability to the basic events and computing the probability of the top event.

5. Proposed Algorithm

5.1 Reliability allocation Algorithm

Step1:-Draw Use Case diagram of the System. UC={set of all Use Cases}

Step2:-Draw the component diagram for each Use Case. CM={set of all components}

Step 3:-Construct the composed-Tree CT(E,G,C)

Where
E is the set of events of Successful execution of Use Cases.

G is the set of gates(AND,OR).

C is the set of basic events of Successful execution of component under the Use Case.

Step 4:-Derive Fault Tree FT(E',G',C') from the composed tree, Where all the elements represent reverse of the elements present in the composed diagram.

Step 5:-Deciding the maximum fault-rate of the system. FR<=F(sys)

So, the minimum reliability is R(sys)=1-F(sys).

Step 6:-Finding maximum fault-rate of each component. (i)Formation of different cut-sets present in the system. Call *CUTSETEVAL()*

(ii)Derive maximum fault-rate of each component present under certain cut-set. Using

$F(m(cut(i)))\leq\left| F(sys)/(\text{Total no. of cut sets})\right| **(1/|cut(i)|)$

Where $F(m(cut(i)))$=Fault rate of component(m) under cut set(i).

If one component is repeated in more than one cut-set then consider the maximum fault-rate of that component. Here minimum reliability of each component can be evaluated by using $R(m)=1-F(m)$.

Step 7:-Finding component reliability

(i)Finding module importance $(I(m))$.

$I(m)=(1/k)\sum \theta(m,i)$

Where $\theta(m,i)$=importance of module m according to actor i. k=Total no. of actors.

(ii)Formulation of L.P.P.

Objective Function

$Max\ U=\sum I(m)*R(m)$

Constraints

$R(m)\leq Upp(m)$
$R(m)\geq Low(m)$
$h(m)+c(m)*R(m)\leq\alpha*v(m)$
$\sum(h(m)+c(m)*R(m))\leq C*$

Where

$Upp(m)$=Upper limit reliability of module(m).
$Low(m)$=Lower limit reliability of module(m).
$h(m)$=cost given to module m by constraint $R(m)$.
$c(m)$=Adjustable cost
$\alpha=(PR-1)$(PR=profit rate extended by vendor)
$v(m)$=selling price
$\alpha*v(m)$=Estimated developing cost of module(m) and actual cost should not exceed it.
$v(m)=I(m)*V$
[$I(m)$=importance of module(m),V=Total cost of the software]
$C*$=Available developing resource, the whole cost of reliability should be less than it.

(iii)Implementing GA to solve the above L.P.P

The above L.P.P can be solved by using GA to obtain optimize reliability of the components.

Where fitness function is

$U=\sum I(m)*R(m)$

And suppose the optimized reliability of each module of the system is (R(1),R(2),……..,R(n)).

Step 8:-Evaluating reliability of each Use Case

From the composed diagram derive the ordering structure of the components to give successful execution of use case(i).

If OR gate → parallel connection

AND gate → Series connection

For parallel connection use the formula$(1-\prod(1-R(i)))$and for series connection use $\prod R(i)$.

5.2 Cut-Sets Formation Algorithm

A Cut-set is a collection of basic event that gives rise to top event. The Cut-sets are evaluated from the Fault Tree FT(E',G',C'). Cut-sets are the sets that causes the system level failure.

```
CUTSETEVAL()
Input:-FT(E',G',C')
Output:- Cut Sets.
1. For Top level event.
   If middle level events are
   ORed-place the middle level events in separate rows
   ANDed-place the middle level events in same row.
2. For each row derived from step-1.
      For each column
      If lower level events are
         ORed-place the lower level events in
         separate rows.
         ANDed-place the lower level events in
      same row.
3. Take the unique Cut Sets.
```

6. Description of algorithm

The above algorithm generally based on the static reliability allocation because here we are using use case diagram along with the component diagram, which represents the static nature of the component involvement. It is a fault-rate allocation method for reliability assessment. In the Reliability allocation algorithm step 1-6 is based on fault-rate allocation and step7-8 is based on reliability assessment.

7. Conclusion and Future work

It is a convenient method for modeling static software reliability assessment. The two important tools used here are the composed tree and the fault tree. It is a fault allocation based modeling. UML is used here as a modeling tool, which assists the composed tree and fault tree for reliability measurement. Also from the component reliability we have designed the static use case reliability. In software it is not easy to formulate the L.P.P. and get the reliability. Evolutionary approaches can be used for optimization that can estimate the approximate reliability. Like in Genetic Algorithm the parameters can be used to formulate a fitness function and optimal reliability can be estimated.

References

[1] Philip J. Boland,"Challenges In Software Reliability and Testing".

[2] Man Cheol Kim, Seung Cheol Jang and Jaejoo Ha,"Possibilities and Limitations Of Applying Software Reliability Growth Models To Safety Critical Software" NUCLEAR ENGINEERING AND TECHNOLOGY,VOL.39 NO.2 APRIL 2007.

[3] 3. Jianwen XIANG, Kokichi FUTATSUGI, "Fault Tree Analysis of Software Reliability Allocation".

[4] Ganesh J Pai, Joanne Bechta Dugan,"Enhancing Software Reliability Estimation Using Bayesian Networks and Fault Trees".

[5] 5. K. Appukkutty, Hany H. Ammar, Katerina Goseva Popstajanova, "Software Requirement Risk Assessment Using UML", IEEE 2005.

[6] 6. William B. Smith V, "Early Component-Based Reliability Assessment using UML Based Software Models" ,Thesis submitted in west Virginia University,Lane Department of computerscience and Electrical Engineering,2002.

[7] 7. M.Thangarajan, "Software Reliability Prediction Model, Whitepaper" , TATA ELXS Engineering Creativity.

[8] 8. Deepak Pengoria, Saurabh Kumar, "A Study on Software Reliability Engineering Present Paradigms And Its Future Consideration" , ISSRE 2009.

[9] 9. Machael R. Lyu ,"Software Reliability Engineering: A Roadmap".

[10] 10. Debasis Kundu, Monalisha Sarma, Debasis Samanta,"A Novel Approach to System Testing And Reliability Assessment Using Use Case Model ", ISEC 2008.

Mr. Debasis Mohapatra is a Lecturer in Parala Maharaja Engg. College (A Constituent College of BPUT), Odisha. His research interest is Automated Test case Generation, Software Reliability Engineering, Program Slicing, Soft Computing.

Ms. Sasmita Rani Behera is a Lecturer in Parala Maharaja Engg. College (A Constituent College of BPUT), Odisha. Her research interest is Automated Test case Generation using UML diagrams.

Ms. Ranumayee Sing is a Lecturer in Parala Maharaja Engg. College (A Constituent College of BPUT), Odisha. Her research interest is Automated Test case Generation using UML diagrams, Wireless Ad-Hoc Networks.