A Probabilistic Assessment Method for Software Reliability in the Application of Safety Standards

YAGUANG YANG
Office of Nuclear Regulatory Research
US NUCLEAR REGULATORY COMMISSION
SCOPE

• Motivation
• Existing Models for Software Reliabilities—Why A New Model
• Ideas of the New Model
  – Flow Network Model for Software Logic-Structure
  – Test-Result Estimates Logic Error Probability
• Mathematic Formulae
• A simple Example
• Automated Tool
Motivation

• Introducing DI&C in Nuclear Power Plants (NPP) has been happening and will be accelerated

• NPP operational safety requires assessing reliability of DI&C systems and embedded software
  – IAEA “Safety of Nuclear Power Plant: Design”

• There is no commonly accepted method to estimate software reliability at present
Difference between Hardware Wear Out and Software Logic Error

• Typical Hardware failure is due to slow wear out before eventual failure
  – Most important feature--failure time is uncertain
  – Traditional reliability theory is based on hardware failure mechanisms.
  – The phenomena is modeled as random failure time using Poisson distribution, exponential distribution, Weibull distribution, Gamma distribution, etc.

• Software failure is due to logic errors, problems happen whenever the error code is executed.
  – Failure is triggered by events and data inputs. It depends on which part of the software is executed and what data set is used.
  – Failure is time independent though it appears to be time related, hardware reliability theory is not applicable
  – Structure based model should be considered. Test results should be used to estimate the reliability of individual parts of the software
Existing Software Reliability Models

- Majority of software reliability assessment methods model failures using time dependent distributions
  - Traditional reliability theory for hardware failures is available
  - Failure modes of hardware and software are different
- Software reliability metrics based models
  - Metrics related to software development procedures and controls are good measure of software quality
  - Hard to establish a quantitative relationship between software reliability and the metrics
- Test coverage models
  - Using test result to quantify the metrics, such as lines of executable code, independent test path, function requirements, and hazard cases
  - Structure is not used, the estimation is less accurate
- Complexity based models
  - For example, lines of code
  - Reasonable but more detailed structure information is not used
- Patri Nets models
  - Able to model real time software
  - Difficult to implement (places, transitions, tokens, arcs, states)
- Component based models
  - Consider software component
  - Difficult to get transitional probabilities between components
Logic Structure Based Model

• Why do we use structure based models
  – Software is composed of many small logic parts and each software has its own structure
  – Given event and input data, software logic structure determines which parts of software are executed, it implicitly determines if some error will be triggered or not
  – Software failure reliability depends on software structure and the reliability of individual parts of the software
  – Structure splits software into small parts and helps to determine reliability of individual part by using test result
Flow Network Structure

- Flow network naturally matches to the software structure
  - Software has a starting point which is the source of the flow network
  - Software has an end point which is the sink of the flow network
  - Software has many logic start and end branch points (if-end) which are nodes of the flow network
  - Software is composed of many small parts between if-end which are the edges of the flow network
  - Software has parallel parts and serial parts which are parallel edges and serial edges of the flow network
Test Result Based Estimation

• Ideas of using test result to estimate the reliability of individual software part
  – The more successful runs a software part has, the higher reliability the software part has
  – Simple pass/failure behavior in a single test run can be modeled by Bernoulli distribution
  – Repeated test runs of the same software part can be modeled by binomial distribution

• Why is test based estimation good
  – Estimation is objective because test data is used and the probabilistic model makes sense
  – Expect better reliability estimation because detailed information is used
  – Test results are available before software release, so little extra effort is needed for collecting data to be used in software reliability calculation
Software Reliability Calculation

• Software partition can be obtained by using flow network model.

• Reliability of individual part of the software can be obtained by software test results
  – Bernoulli distribution is used for parts that have been tested only once
  – Binomial distributions is used for parts that have been tested multiple times

• Structure model is used to calculate the entire software reliability
  – Reliability of parallel parts and serial parts can be calculated by using traditional reliability calculation method
  – Repeat the simplification process will lead to the entire software reliability
Main()                                                              //node 1
{
  Data initialization;                                   //edge e11
  If condition A holds                                   //node 2
  {
    Process data;                                   //edge e21
    If data processing success               //node 3
    {
      Save result;                      //edge e31
    }
    Else if data processing fail               {
      Issue a warning;               //edge e32
    }
  }
  Else if condition A does not hold     //node 4
  {
    Print “condition A does not hold”;       //edge e22
  }
  Clean memories;                                    //edge e41
}
Bayesian Estimation

• Assume a part of the software is tested \( n \) times and the \( k \) tests are passed. Let \( e_{ij} \) be the jth edge under node i, the failure probability of \( e_{ij} \), the reliability of the edge can be estimated by Bayesian method

\[
E \left( p_{ij} \mid k, n, \alpha_{ij}, \beta_{ij} \right) = \frac{\alpha_{ij} + k}{\alpha_{ij} + \beta_{ij} + n}
\]

• where \( \alpha_{ij}, \beta_{ij} \) are parameters of Beta distribution
Some Notations

- $h_{ij}$ is number of tests achieved for the edge $e_{ij}$
- $t_{ij}$ is the execution time used for running the edge $e_{ij}$
Parallel Edge Simplification

- If a block under node i is composed of parallel connected edges, the total number of executions under node i is $h_i = \sum_{j=1}^{j_m} h_{ij}$, total execution time for these parallel edges is $t_i = \sum_{j=1}^{j_m} t_{ij} h_{ij}$
- The reliability of the block composed of parallel connected edges under node i is
  \[ R_{i1} = \sum_{j=1}^{j_m} \frac{h_{ij} t_{ij}}{t_i} p_{ij} h_{ij} \]
- The execution time and number of executions for the combined edge are
  \[ t_{i1} = \frac{1}{H_{i1}} \sum_{j=1}^{j_m} t_{ij} h_{ij} \quad H_{i1} = \sum_{j=1}^{j_m} h_{ij} \]
Serial Edge Simplification

- If a block under node $i$ is composed of $i_s$ serial connected edges, the total number of executions under node $i$ is $h_i = h_{ij}$, total execution time for these parallel edges is $t_i = \sum_{i=1}^{i_s} t_{ij} h_{ij}$.
- The reliability of the block composed of parallel connected edges under node $i$ is
  \[ R_{i,j} = \prod_{i=i_1}^{i_s} p_{ij}^{h_{ij}} \]
- The execution time and number of executions for the combined edge are
  \[ t_i = \sum_{i=i_1}^{i_s} t_{ij} \quad h_i = h_{ij} \]
The Simple Example

- Assume (1) three tests are conducted. The test path are 11-21-31-41, 11-21-32-41, and 11-22-41. Assume further that the total test time is 0.00011 hours and 0.00001 hours for every edge.
Automated Tool

- The computation can be tedious when software has many small logic parts
- An automated tool as part of software development environment is proposed to calculate software reliability
  - Most software development tools have debug mode and release mode that have many features for reliability calculation.
  - Propose to add test mode with the following features
    - When software is ready for test, software is compiled in test mode, the compiler should generate the flow network for the software to be tested
    - When software test is carried out, the execution number and time for each edge should be recorded
    - After each test run, if the test result is checked and believed to be correct, an acceptance signal should be sent and upon the acceptance, the software development environment should automatically update the reliability for each edge and calculate the reliability for the software using formulae developed in this paper
    - If a defect is found, it should be fixed and the reliability for that edge should be reinitialized
    - The final reliability number should be saved and can be read at any time