Estimating Software Reliability In the Absence of Data

Joanne Bechta Dugan (jbd@Virginia.edu)
Ganesh J. Pai (gpai@Virginia.edu)
Department of ECE
University of Virginia, Charlottesville, VA
Research Motivation

- Estimate of reliability of systems containing software

- How do we do this early during design?

- Many quantitative approaches to estimate software reliability rely on test data
  - Test data may not be available till late into the project
  - Process information is available which is usually not considered in the reliability estimate

- Develop a reasonable “first-pass” prediction when little or no data is available

- Provide confidence in the reliability estimates
Proposed Approach

• Develop a generic bayesian model (BBN) based on software development lifecycle
  ▪ Capture the influence of development processes on software reliability
  ▪ Provide a “first pass reliability estimate”
  ▪ Refine BBN & Reliability Estimate as testing data / lifecycle / process information is available

• Inputs to the bayesian network would be
  ▪ Metrics available early during design
  ▪ Insights from the software architecture
  ▪ Expert insights/ engineering judgment
  ▪ Knowledge of module quality from quality classification
  ▪ Other insights i.e. Were formal methods used?, etc.

• Possible outputs
  ▪ A probability that the software reliability lies in a certain range
  ▪ Confidence value that the software reliability has an acceptable value
  ▪ An estimate of # of residual faults
Proposed Approach – A Bigger Picture

- Use a point estimate from the predicted range in the BE for a fault tree of the system.
- The fault tree provides an estimate for system unreliability, taking software into account.
- Related work by Smidts et al. at UMD:
  - We are not developing a FT for software.
  - Rather develop FT for the entire system, where software is a basic event.
  - No enumeration of process failure modes.
  - If available, can be incorporated.
  - They note that a bayesian framework incorporating history and combining qualitative and quantitative information will be valuable.
Bayesian Belief Networks and Fault Trees

- Fault Trees are a graphical representation of logical relationship between
  - Basic failure events (BE) and
    System failure (Top event)
  - Have Static / Dynamic gates
  - Computes P(Top Event) as a function of P(BE)
- Can express combinatorial and non-combinatorial failures

- BBNs are a directed acyclic graph with nodes and edges
  - Nodes represent random variables with probability distributions
  - Edges represent weighted causal relations between nodes
  - Graphical representation of probability propagation using Bayes’ formula
Example

- Front end loader with a software controlled weight-monitor system
- System reports weight of load to the operator
  - Incorrect reporting can cause failure of hydraulic system
  - Loader may tip over if (weight of load) > Max. permissible load

System Fault Tree
Hypothetical BBN for Software

- Node probability tables associated with each node
- Each node can have multiple states
- NPT of a Node A with parents B, C → P(A_i | B_j, C_k)
- NPTs define the “weight” of the causal relations between nodes

### Deterministic NPT for Quality Policy

<table>
<thead>
<tr>
<th>QA</th>
<th>QC</th>
<th>QP</th>
<th>HigherR</th>
<th>MiddleR</th>
<th>LowerR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Superior</td>
<td>Superior</td>
<td>Superior</td>
<td>100.00</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Superior</td>
<td>Medium</td>
<td>Superior</td>
<td>80.000</td>
<td>20.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Superior</td>
<td>Mediocre</td>
<td>Superior</td>
<td>15.000</td>
<td>70.000</td>
<td>5.000</td>
</tr>
<tr>
<td>Medium</td>
<td>Superior</td>
<td>Medium</td>
<td>5.000</td>
<td>90.000</td>
<td>5.000</td>
</tr>
<tr>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>5.000</td>
<td>80.000</td>
<td>15.000</td>
</tr>
<tr>
<td>Medium</td>
<td>Mediocre</td>
<td>Mediocre</td>
<td>0.000</td>
<td>25.000</td>
<td>75.000</td>
</tr>
<tr>
<td>Inferior</td>
<td>Superior</td>
<td>Mediocre</td>
<td>15.000</td>
<td>70.000</td>
<td>5.000</td>
</tr>
<tr>
<td>Inferior</td>
<td>Mediocre</td>
<td>Mediocre</td>
<td>0.000</td>
<td>5.000</td>
<td>95.000</td>
</tr>
<tr>
<td>Inferior</td>
<td>Mediocre</td>
<td>Mediocre</td>
<td>0.000</td>
<td>0.000</td>
<td>100.00</td>
</tr>
</tbody>
</table>
Hypothetical priors are computed
  - After gathering data, instantiating parent nodes
  - Transition weights are determined by expert judgment
  - *Reliability Estimate* states represent $R > 0.95; 0.95 \leq R \leq 0.9; R < 0.9
Computation of Priors

- For a variable $A$ with states $a_1, a_2, \ldots, a_n$; (Root nodes in the BBN)
  - $P(A) = \text{Probability distribution over the states} = P(a_1, a_2, \ldots, a_n)$;
  - $a_i \geq 0; \sum_{i=1}^{n} a_i = 1$

- For another variable $B$ with states $b_1, b_2, \ldots, b_m$; (Child/Target node)
  - $P(A/B)$ is an $n \times m$ table containing numbers $P(a_i/b_j)$
  - This is the NPT in the BBN
  - Also $P(a_i/b_j)P(b_j) = P(a_i, b_j)$ and $P(a_i) = \sum_{j=1}^{n} P(a_i, b_j)$

- Finally $P(A) = \sum_{B} P(A, B)$

- The BBN is an elegant graphical abstraction for this computation
FTA of the System

Weight Monitoring System Failure

Control Lever Failure
Dump Cylinder Fails
Metal Fatigue
Operator Error
Display Monitor Failure

Lever Lock Failure
Operator Error
Lever Shifts
Monitor HW Failure
Monitor SW Failure

Developement Quality
- High: 35.4
- Medium: 42.2
- Low: 22.3

Code Quality
- High: 55.8
- Medium: 16.2
- Low: 28.0

Design Quality
- High: 47.4
- Medium: 20.4
- Low: 32.2

Architecture
- Superior: 32.1
- Medium: 46.5
- Inferior: 21.4

Quality Assurance Policy
- Strict: 55.0
- Medium: 25.0
- Mediocre: 20.0

Quality Control
- Strict: 63.2
- Poor: 36.8

Quality Policy
- Superior: 50.6
- Medium: 42.1
- Mediocre: 7.35

Reliability Estimate
- Higher Range: 29.7
- Middle Range: 56.3
- Lower Range: 14.0

Convert to Unreliability
## Analysis – Result of FTA

<table>
<thead>
<tr>
<th>Basic Event</th>
<th>Failure Rate (λ)</th>
<th>Failure Probability (P)</th>
<th>System Unreliability (Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lever Lock Failure</td>
<td>0.0045</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Operator Error</td>
<td>-</td>
<td>0.03</td>
<td></td>
</tr>
<tr>
<td>Lever Shifts</td>
<td>0.00065</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Metal Fatigue</td>
<td>0.0035</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Monitor HW Failure</td>
<td>0.00000565</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Monitor SW Failure</td>
<td>-</td>
<td>0.1</td>
<td></td>
</tr>
</tbody>
</table>
Analysis - Effect of Evidence

Quality Assurance Policy
- Strict: 0
- Medium: 100
- Mediocre: 0

Code Quality
- High: 55.8
- Medium: 16.2
- Low: 28.0

Design Quality
- High: 47.4
- Medium: 20.4
- Low: 32.2

Development Quality
- High: 35.4
- Medium: 42.2
- Low: 22.4

Architecture
- Superior: 32.1
- Medium: 46.5
- Inferior: 21.4

Quality Policy
- Superior: 63.2
- Medium: 36.8
- Mediocre: 0

Reliability Estimate
- HigherRange: 32.1
- MiddleRange: 60.0
- LowerRange: 7.86

Source of BE probability in Fault Tree
Comparison

Reliability Estimate before evidence

<table>
<thead>
<tr>
<th>Reliability Estimate</th>
<th>HigherRange</th>
<th>MiddleRange</th>
<th>LowerRange</th>
</tr>
</thead>
<tbody>
<tr>
<td>HigherRange</td>
<td>29.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MiddleRange</td>
<td>56.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LowerRange</td>
<td>14.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reliability Estimate after evidence

- Evidence has increased probability of $0.95 \leq R \leq 0.9$
- Point estimate is unchanged $\Rightarrow$ Result of FTA is the same
- **However**, we now have more confidence in this estimate

- Future work – quantifying the confidence value
Summary of the Methodology

1. Develop generic BBN to estimate software reliability
2. Plug value into Basic Event in Fault Tree
3. System Reliability Estimate considering Software
4. Evidence becomes Available
5. Refine estimate
6. Refine BBN to Incorporate evidence
7. Qualitative factors, Expert judgment
Conclusions

• Process and product information / evidence is incorporated as they become available.
• Reliability estimates can be refined
• The Bayesian framework provides confidence in the reliability estimates
• An early estimate of reliability - by blending qualitative data, expert opinion/ engineering judgment with quantitative data

Future Work

• Developing a generic BBN based on software lifecycles
• Identifying nodes of the BBN
• Determining NPT values
• Define a stopping criterion for adding BBN nodes

Especially interested in getting real data to validate methodology
Please let us know if you’d be willing to share data!

Contact
Joanne Bechta Dugan (jbd@Virginia.edu)
Ganesh J Pai (gpai@Virginia.edu)
Project

Modeling the “Safe Enough to Release” Decision

Principal Investigator

Joanne Bechta Dugan, Ph.D.
jbd@virginia.edu

Susan K. Donohue
susand@virginia.edu
Validation of System Safety

Validation of system safety is the process by which it is determined that the system, as designed, can be expected to operate without incident for a given time period within the specified requirements for safety.
The Problem

Current methods of validating safety may lead to the certification of a system as “correct,” “valid,” and “safe” when all known failure states have not been observed or tested in significant numbers.
The Problem

- How can an assessor extend the validation process to gain a greater confidence that the system is “safe enough”?

- What support is available to allow an assessor to weigh and review both quantitative and qualitative evidence in a systematic, repeatable, and auditable fashion?

- How can uncertainty be factored explicitly into the assessment?
In general...

How do we go from... to...

Is the system safe enough?

Test Results
Quality Assurance
Requirements Review
FTA

Personal and Team CMM
Prototype Performance
System Design

How can we bridge the gap between what we know and the requirements we must meet with the information that’s available? ESPECIALLY WITH A UNIQUE SYSTEM?

I have an acceptable level of belief that the system will operate as specified.
One Possible Answer

Bayesian Belief Networks are a modeling formalism that support the proposed extension of the safety validation process.

A BBN from the Halden Project
Modeling Component Sources

Evidence comes primarily from:

- Process Evidence
- Product Evidence
- Engineering Judgment
Modeling Components

Variables to investigate and assess include:

- Dependability
- Quality (e.g., system, process, and supplier)
- Hazard / Risk Analysis
- System Design
- Compliance to Standards
- Results of V&V Activities
- “Safeness” of System Components and Interfaces
- Support Materials (e.g., training materials and project documents)
- Behavior of Prototypes or Simulation
Proposed BBN (2)

- Project Management
- Process Maturity
- Personnel
- Process Quality
- Safe Enough?
- Model
- Performance-Verified
- Correctness-Validated
- System Materials
- Hardware
- Software
- Interfaces
- Requirements
- Test Plan
- Training
- Project Documents
Conclusions

Our examples represent the first step in modeling the assessment of the safety validation process for a generic, unique ultrareliable computer-based system.

Results will vary depending on the system being modeled, BBN components, and the expert opinion elicitation methods.
Future Work

- Elicit expert opinion to populate BBN nodes’ NPTs
  - Develop elicitation instrument
  - Evaluate usefulness of approximate reasoning and fuzzy intervals in elicitation

- Provide importance factors for nodes and paths
  - Adapt FTA importance factors
  - Develop new importance factors

- Develop case studies
  - Validate models against past projects
  - Apply models to current projects
Publications / Conferences
