LWRS Efforts on Qualitative and Quantitative Risk Assessment of Digital Software Systems

February 2024

Edward Chen, Han Bao, Tate H Shorthill
DISCLAIMER

This information was prepared as an account of work sponsored by an agency of the U.S. Government. Neither the U.S. Government nor any agency thereof, nor any of their employees, makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness, of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the U.S. Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the U.S. Government or any agency thereof.
LWRS Efforts on Qualitative and Quantitative Risk Assessment of Digital Software Systems

Identification and Estimation of Software Failures

Edward Chen
Han Bao
Tate Shorthill
Idaho National Laboratory

U.S. DOE Light Water Reactor Sustainability (LWRS) Program, Risk-Informed Systems Analysis (RISA) Pathway
• Installation or modification of existing equipment with digital systems
  o 10 CFR 50.92 Issuance of Amendment (e.g., License Amendment Request)
  o Requires licensees to meet 10 CFR 50.92; no significant hazard consideration such that modification does not:
    1. involve an increase in prob. or conseq. of a previously evaluated accident
    2. create new/different accident
    3. reduce margin of safety
• PRA and FTA is used to show that system modification meets such criteria
  o Current software failure probability quantification strategy:
    • Software basic events use bounding estimate 1E-4 (overly conservative)
      • From IEC 61508 SIL 4 safety rated equipment on demand

We want to reduce the use of bounding estimates in risk analysis to assist licensing and deployment of software DI&C systems
Objective:

1. Is there a better way to identify software common cause failure events?

2. Is there a better way to quantify these events to eliminate over conservative risk estimations?

- Drawn from conservative IEC 61508 SIL 4 estimates
Regulatory anticipation of the use of STPA for hazard analyses in support of claims related to licensing.

Qualitative method for the identification of software hazards via STPA.

Redundancy Guided Systems Theoretic Hazard Analysis (RESHHA) developed by INL for full-scope risk assessment of DI&C systems.
LWRS-developed DI&C Risk Assessment Framework

• Technical approach:
  o Develop a risk assessment framework to support modernization efforts for safety-related DI&C systems including all aspects of typical risk:
    (1) hazard analysis, (2) reliability analysis, and (3) consequence analysis
  o Evaluate the impact of digital failures at the component level, system level, and plant level
Identifying Software Failure Modes for Risk Assessment

What can go wrong?

LWRS-developed DI&C Risk Assessment Framework

- Hazard Analysis
- Reliability Analysis
- Consequence Analysis

RESHA
(Redundancy-Guided System-Theoretic Hazard Analysis)

PRA + UQ
(Probabilistic Risk Assessment + Uncertainty Quantification)

Multiscale Quantitative Reliability Analysis

- BAHAMAS
  (Bayesian and HRA-Aided Method for the Reliability Analysis of Software)
- ORCAS
  (Orthogonal Defect Classification for Assessing Software Reliability)

CCF Modeling and Estimation
Overall Workflow of RESHA

Major contributions of RESHA FT:
1. Incorporates design redundancies in the FT
2. Models software dependencies within the FT
3. Models software failures as Unsafe Control Actions or Unsafe Information Flow
Software Failures in DI&C Systems

STPA – Software ‘failure’ is due to inadequate constraints leading to ‘misbehaviors’ and loss

• Rules for operation during development were not strict enough (i.e., loophole)
• Misbehaviors = Unsafe Control Actions (UCA)

<table>
<thead>
<tr>
<th>Unsafe Control Action (UCA) Loss Context</th>
<th>Type A</th>
<th>Type B</th>
<th>Type C</th>
<th>Type D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control action not provided when required context.</td>
<td>Control action provided when not required causing hazard.</td>
<td>Control action is early, late, or out-of-order.</td>
<td>Control action is stopped too soon or applied too long.</td>
<td></td>
</tr>
</tbody>
</table>

Auto emergency braking (AEB)

• AEB brake when collision imminent and not user initiated
• How does it “know” when to brake? What could go wrong?

<table>
<thead>
<tr>
<th>Unintentional collision during autopilot on highway</th>
<th>Type A</th>
<th>Type B</th>
<th>Type C</th>
<th>Type D</th>
</tr>
</thead>
<tbody>
<tr>
<td>AEB does not detect car in front and crashes.</td>
<td>AEB brakes when squirrel detected, spurious.</td>
<td>AEB brake is late causing minor collision.</td>
<td>AEB brake is too weak causing minor collision.</td>
<td></td>
</tr>
</tbody>
</table>

UCAs are undesirable software events and may be modelled as software basic events.
Example FT

**Top event:**
- Frontal collision when operator is unaware

What software failures will lead to this event?

- Missing or incorrect environmental mapping information causes loss
- Controller behaves as it is programmed but a loss still occurs.
What are software CCFs?
- Failure that causes 2 or more components to fail simultaneously from a shared causal factor

2 division redundant w/ diverse input

RESHA is an alternative FT construction method that allows exact modeling of the control structure via software basic events as UCA/UIFs.
Quantifying Software Failure Modes for Risk Assessment

LWRS-developed DI&C Risk Assessment Framework

- Hazard Analysis
- Reliability Analysis
- Consequence Analysis

- RESHA
  (Redundancy-Guided System-Theoretic Hazard Analysis)
- PRA + UQ
  (Probabilistic Risk Assessment + Uncertainty Quantification)

Multiscale Quantitative Reliability Analysis

- BAHAMAS
  (Bayesian and HRA-Aided Method for the Reliability Analysis of Software)
- ORCAS
  (Orthogonal Defect Classification for Assessing Software Reliability)
- CCF Modeling and Estimation

How likely is it?

What can go wrong?
What information is collected during the software development process?

- Defect reports -> Github issue reports (root causes)

Title: [CASSANDRA-867] Invalid host/port parameters to cassandra-cli leaves system in unrecoverable state

Link: https://issues.apache.org/jira/browse/CASSANDRA-867

Environment: WinXP / java 1.6 / svn @ 921110 trunk

Dates: Created: Tue, 9 Mar 2010 20:44:31 +0000
Resolved: Tue, 9 Mar 2010 21:03:58 +0000

Description: bin\cassandra-cli.bat -host localhost -port 8880 (cassandra not running localhost/8880 )

Starting Cassandra Client Exception connecting to localhost/8880 - java.net.ConnectException: Connection refused: connect  Welcome to cassandra CLI.

- What are the characteristics of this defect?
- How does this defect translate to operator losses?
- How does this defect impact reliability?

Defects come in all shapes & sizes, how they impact software is difficult to determine from a case-by-case assessment.

Are there patterns that exist in the data that we can use to interpret how development bugs translate to software failures?
Orthogonal defect classification:
1. Software defects with shared characteristics are grouped together (classification)
   a) Function, checking, algorithm, etc.
2. Defects with shared characteristics cause the same software failure modes (UCA/UIF)

<table>
<thead>
<tr>
<th>Defect Class (Abridged)</th>
<th>Classification Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>Significantly affects capability, interfaces, functionality, etc. s.t a formal design change is required. The program is missing a key overlooked functionality.</td>
</tr>
<tr>
<td>Algorithm</td>
<td>Efficiency or correctness problems that affect the task performance. Equation is not completely correct.</td>
</tr>
<tr>
<td>Checking</td>
<td>Related to the correctness of conditional statements, data verification, and branch traversal.</td>
</tr>
</tbody>
</table>

Grouping allows generalized assessment on how each class impacts the code useability

1. ODC allows classification of common software defects into specific categories for numerical analysis.
2. A defect class is analogous to grammatical errors in essay writing.
Data Collection & Correlation

Defect Data Correlation:
• 3526 defects were collected from 3 database management software, 1 autonomous car software, and 1 microcontroller operating system
  • MongoDB, Cassandra, Hbase, OpenPilot, Zephyr (open source Github repositories)

Defects with shared characteristics cause the same UCA/UIF software failure mode
1. The relationship between defect class detected and UCA caused holds about true for 5 different pieces of software.

2. A global relationship table is constructed for all defect classes & UCA types
   a) If a Timing defect exists, it has a 62.2% chance to cause a UCA-C;

<table>
<thead>
<tr>
<th>Defect Class</th>
<th>UCA-A</th>
<th>UCA-B</th>
<th>UCA-C</th>
<th>UCA-D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm Defect</td>
<td>0.331</td>
<td>0.137</td>
<td>0.339</td>
<td>0.194</td>
</tr>
<tr>
<td>Assignment Defect</td>
<td>0.284</td>
<td>0.648</td>
<td>0.057</td>
<td>0.011</td>
</tr>
<tr>
<td>Checking Defect</td>
<td>0.478</td>
<td>0.262</td>
<td>0.241</td>
<td>0.191</td>
</tr>
<tr>
<td>Function Defect</td>
<td>0.385</td>
<td>0.24</td>
<td>0.24</td>
<td>0.135</td>
</tr>
<tr>
<td>Timing Defect</td>
<td>0.189</td>
<td>0.054</td>
<td>0.622</td>
<td>0.135</td>
</tr>
</tbody>
</table>

A relationship between UCA/UIF type and defect class exists. If we know the prob. of a defect class, we can determine the prob. it will cause a UCA/UIF.
**Conclusion**

**LWRS-developed DI&C Risk Assessment Framework**

- **Hazard Analysis**
- **Reliability Analysis**
- **Consequence Analysis**

**Methods to identify software failure modes & losses in control AND information systems**

- **STPA + FT in redundant systems**
  - Unsafe information flow (UIFs) for control-free systems

- **How likely is it?**
  - Methods to identify software failure modes & losses in control AND information systems

  - **Orthogonal Defect Classification (ODC)**
    - Defect relationship to UCA/UIF based on data analytics

**What can go wrong?**

- **Methods to identify software failure modes & losses in control AND information systems**

  - **RESHA**
    - (Redundancy-Guided System-Theoretic Hazard Analysis)

  - **PRA + UQ**
    - (Probabilistic Risk Assessment + Uncertainty Quantification)

  - **BAHAMAS**
    - (Bayesian and HRA-Aided Method for the Reliability Analysis of Software)

  - **ORCAS**
    - (Orthogonal Defect Classification for Assessing Software Reliability)

  - **CCF Modeling and Estimation**
Sustaining National Nuclear Assets

http://lwrs.inl.gov
Failure Mechanisms Leading to UCAs

What are the causes of software failure?

- **Internal mechanisms:**
  - Inadequate control algorithm
  - Action
  - Inconsistent process model
  - Belief
  **Related to software of controller**

- **External mechanism:**
  - Incorrect/inadequate feedback
  - Awareness
  **Related to upstream dependencies**

### Control Action (UCA)

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Control action not provided when required context.</td>
</tr>
<tr>
<td>B</td>
<td>Control action provided when not required causing hazard.</td>
</tr>
<tr>
<td>C</td>
<td>Control action is early, late, or out-of-order.</td>
</tr>
<tr>
<td>D</td>
<td>Control action is stopped too soon or applied too long.</td>
</tr>
</tbody>
</table>

### Information Flow (UIF)

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Feedback is missing when required context.</td>
</tr>
<tr>
<td>B</td>
<td>Feedback is provided when not required causing hazard.</td>
</tr>
<tr>
<td>C</td>
<td>Feedback is early, late, out-of-sync, or out-of-order.</td>
</tr>
<tr>
<td>D</td>
<td>Feedback value is too low, too high, NaN, or Inf.</td>
</tr>
</tbody>
</table>

**UIFs are introduced as software failures in the information systems**
Scenario 1:
• Software on AEB contains a defect that fails when raining.

Result:
• It is raining, AEB fails to engage for oncoming car, collision occurs (UCA failure mode)

Scenario 2:
• Software on the LIDAR contains a defect that when raining obscures environment mapping.

Result:
• It is raining, LIDAR assumes car is there due to defective range finding. AEB triggers spuriously

Scenario 3:
• Optical camera is dirty, and LIDAR has height limit.

Result:
• Optical camera is blind. AEB does NOT engage, and a collision occurs.

UCAs technically cover all loss scenarios listed BUT:

a) Failures of subsystems/dependences cause failures & are exterior to controller.
   ➢ Mitigation strategies on controller does NOT resolve problem.

   -> Which component in the system failed? Who’s at fault?
Overall Lbl2Vec Pipeline for Defect-UCA Classification

Document preprocessing
- Stemming & lemmatization
- Document structuring

Training based on:
- Topic selection
- Keyword selection

Evaluation:
- Development of defect-UCA correlation table
- Kappa agreement value
Unsupervised Topic Classification

What is Lbl2vec?
- Unsupervised document topic classification and clustering technique

How does Lbl2vec work?
- Based on embedded word vectors & keywords, a topic centroid is calculated.
- From there, a similarity metric is derived for each document to the centroids.

rds, a topic each docur
Topic & Keyword Selection

<table>
<thead>
<tr>
<th>Topic</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCA-A</td>
<td>doesn't, restart, block, null, insert, disable, fail, recover, invalid, miss</td>
</tr>
<tr>
<td>UCA-B</td>
<td>remove, delete, false, correct, error, instead, exception, nullPointerException, flush, assertion</td>
</tr>
<tr>
<td>UCA-C</td>
<td>wait, lock, second, time, thread, sleep, exception, reassigned, slave, sync</td>
</tr>
<tr>
<td>UCA-D</td>
<td>long, timeout, block, sufficient, increase, impossible, load, retry, replication, trace</td>
</tr>
</tbody>
</table>
Similarity Metric for Topic Classification

Lbl2Vec output after training

<table>
<thead>
<tr>
<th>Report #</th>
<th>Likely Topic</th>
<th>UCA-A</th>
<th>UCA-B</th>
<th>UCA-C</th>
<th>UCA-D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>UCA-A</td>
<td>0.853</td>
<td>0.263</td>
<td>0.216</td>
<td>0.332</td>
</tr>
<tr>
<td>2</td>
<td>UCA-B</td>
<td>0.563</td>
<td>0.610</td>
<td>0.362</td>
<td>0.253</td>
</tr>
<tr>
<td>3</td>
<td>UCA-C</td>
<td>0.032</td>
<td>0.015</td>
<td>0.361</td>
<td>0.086</td>
</tr>
</tbody>
</table>

**Report 1:**
- Similarity metric is significant (>0.5)
- Different to next topic is significant (>0.25)

**Report 2:**
- Similarity is significant BUT
- Multiple topics are similar

**Report 3:**
- Similarity is insignificant BUT
- Topic difference is significant

Performance of Lbl2Vec on 100 randomly selected UCA classified events by Ed & Tate

<table>
<thead>
<tr>
<th>Dataset Name</th>
<th>Overall Kappa</th>
<th>UCA-A</th>
<th>UCA-B</th>
<th>UCA-C</th>
<th>UCA-D</th>
<th>Matching/Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hbase</td>
<td>0.636</td>
<td>7</td>
<td>15</td>
<td>1</td>
<td>1</td>
<td>24/33</td>
</tr>
<tr>
<td>Cassandra</td>
<td>0.667</td>
<td>4</td>
<td>11</td>
<td>3</td>
<td>0</td>
<td>18/24</td>
</tr>
<tr>
<td>MongoDB</td>
<td>0.809</td>
<td>8</td>
<td>6</td>
<td>10</td>
<td>0</td>
<td>24/28</td>
</tr>
<tr>
<td>Total</td>
<td>0.702</td>
<td>19/26</td>
<td>32/40</td>
<td>14/14</td>
<td>1/5</td>
<td>66/85</td>
</tr>
</tbody>
</table>

Substantial Kappa agreement w/ researcher labels suggests that Lbl2Vec can automate topic classification
Based of topic significance & topic difference:
• 570 of the 4096 defect reports were discarded as indetermined
• 3526 were used to construct defect-UCA correlation table (much better)
  • UCA-A: 1673

<table>
<thead>
<tr>
<th></th>
<th>UCA-A</th>
<th>UCA-B</th>
<th>UCA-C</th>
<th>UCA-D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm Defects</td>
<td>0.331</td>
<td>0.442</td>
<td>0.137</td>
<td>0.098</td>
</tr>
<tr>
<td>Assignment Defects</td>
<td>0.284</td>
<td>0.153</td>
<td>0.648</td>
<td>0.504</td>
</tr>
<tr>
<td>Checking Defects</td>
<td>0.305</td>
<td>0.478</td>
<td>0.262</td>
<td>0.338</td>
</tr>
<tr>
<td>Function Defects</td>
<td>0.385</td>
<td>0.336</td>
<td>0.240</td>
<td>0.423</td>
</tr>
<tr>
<td>Timing Defects</td>
<td>0.189</td>
<td>0.057</td>
<td>0.054</td>
<td>0.083</td>
</tr>
</tbody>
</table>