INL/MIS-24-76654-Revision-0

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

February 2024

Changing the World's Energy Future

Edward Chen, Han Bao, Tate H Shorthill

Idaho National Laboratory

INL is a U.S. Department of Energy National Laboratory operated by Battelle Energy Alliance, LLC

INL/MIS-24-76654-Revision-0

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

Edward Chen, Han Bao, Tate H Shorthill

February 2024

Idaho National Laboratory
Idaho Falls, Idaho 83415

http://www.inl.gov

Prepared for the
U.S. Department of Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

February 15, 2024

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

Identification and Estimation of Software Failures

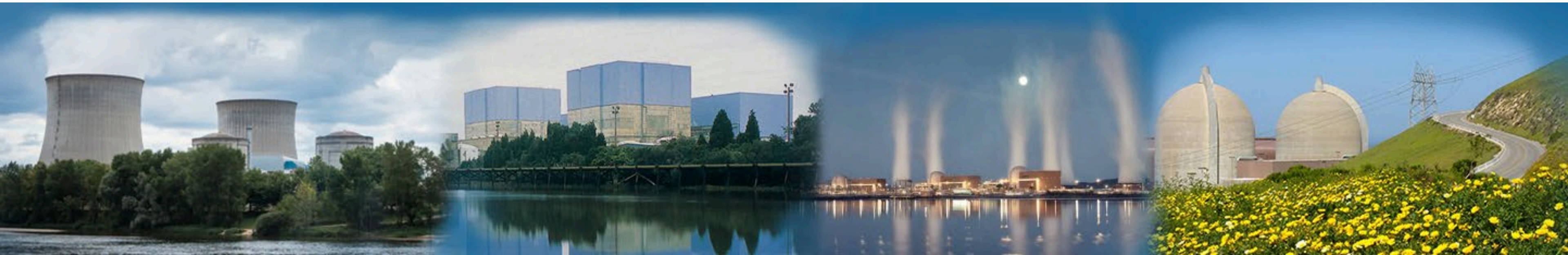
Edward Chen

Han Bao

Tate Shorthill

Idaho National Laboratory

U.S. DOE Light Water Reactor Sustainability (LWRS) Program, Risk-Informed Systems Analysis (RISA) Pathway

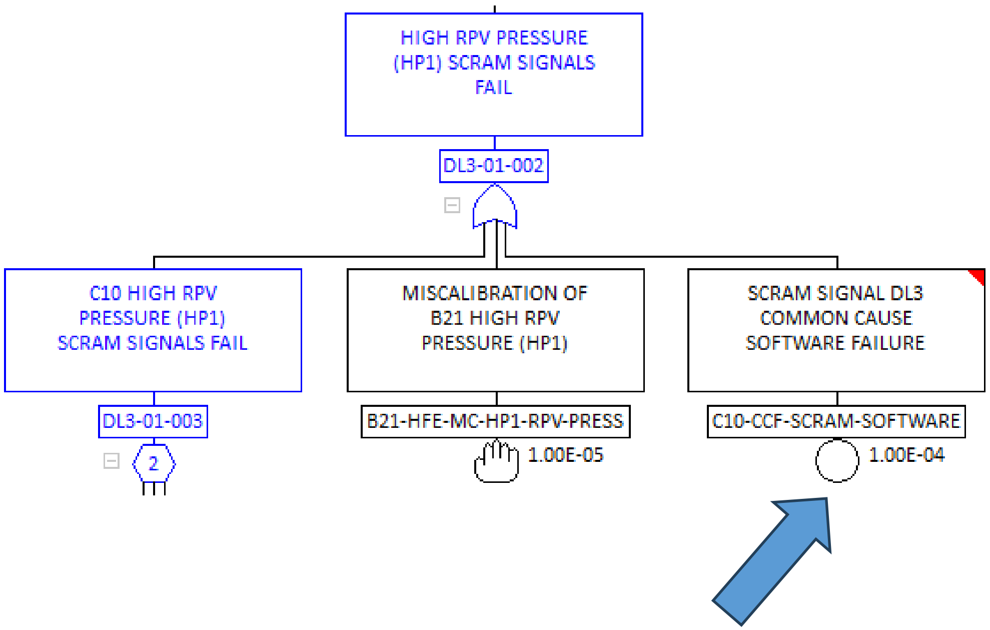


Background

- **Installation or modification of existing equipment with digital systems**
  - 10 CFR 50.92 Issuance of Amendment (e.g., License Amendment Request)
  - 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**
  - 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

Current Industry Method



- Drawn from conservative IEC 61508 SIL 4 estimates

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?

State of Technology

LWRS LIGHT WATER REACTOR SUSTAINABILITY

U.S.NRC
United States Nuclear Regulatory Commission
Protecting People and the Environment

TECHNICAL LETTER REPORT
TLR-RES/DE-2022-006

June 17, 2022

HAZARD ANALYSIS: AN OUTLINE OF TECHNICAL BASES FOR THE EVALUATION OF CRITERIA, METHODOLOGY, AND RESULTS

Regulatory anticipation of the use of STPA for hazard analyses in support of claims related to licensing

EPRI | ELECTRIC POWER RESEARCH INSTITUTE

2021 TECHNICAL REPORT

HAZCADS: Hazards and Consequences Analysis for Digital Systems – Revision 1

Qualitative method for the identification of software hazards via STPA

Common Cause Failure Evaluation of High Safety-significant Safety-related Digital Instrumentation and Control Systems

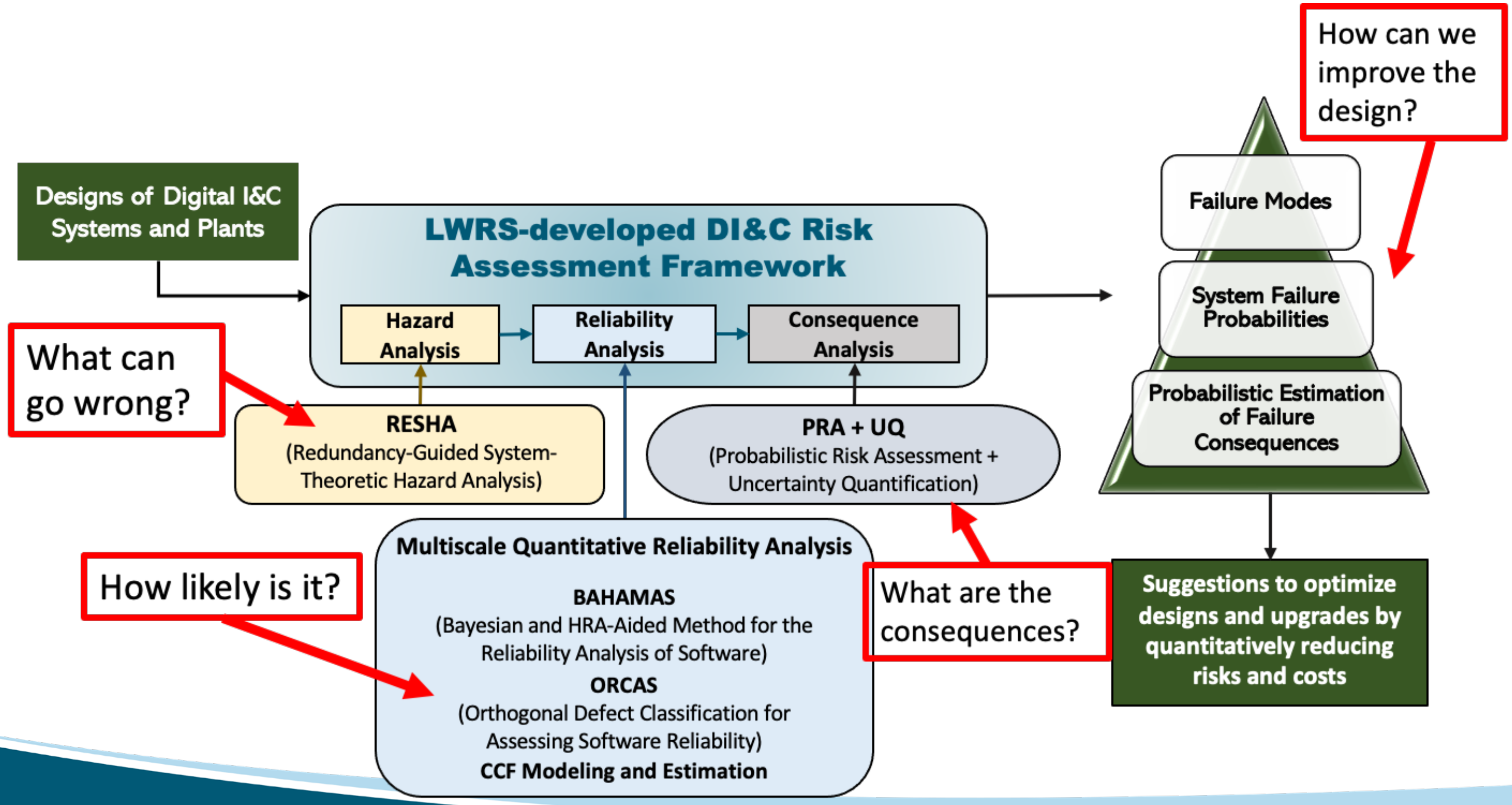
June 2022

Redundancy Guided Systems Theoretic Hazard Analysis (RESHA) developed by INL for full-scope risk assessment of DI&C systems

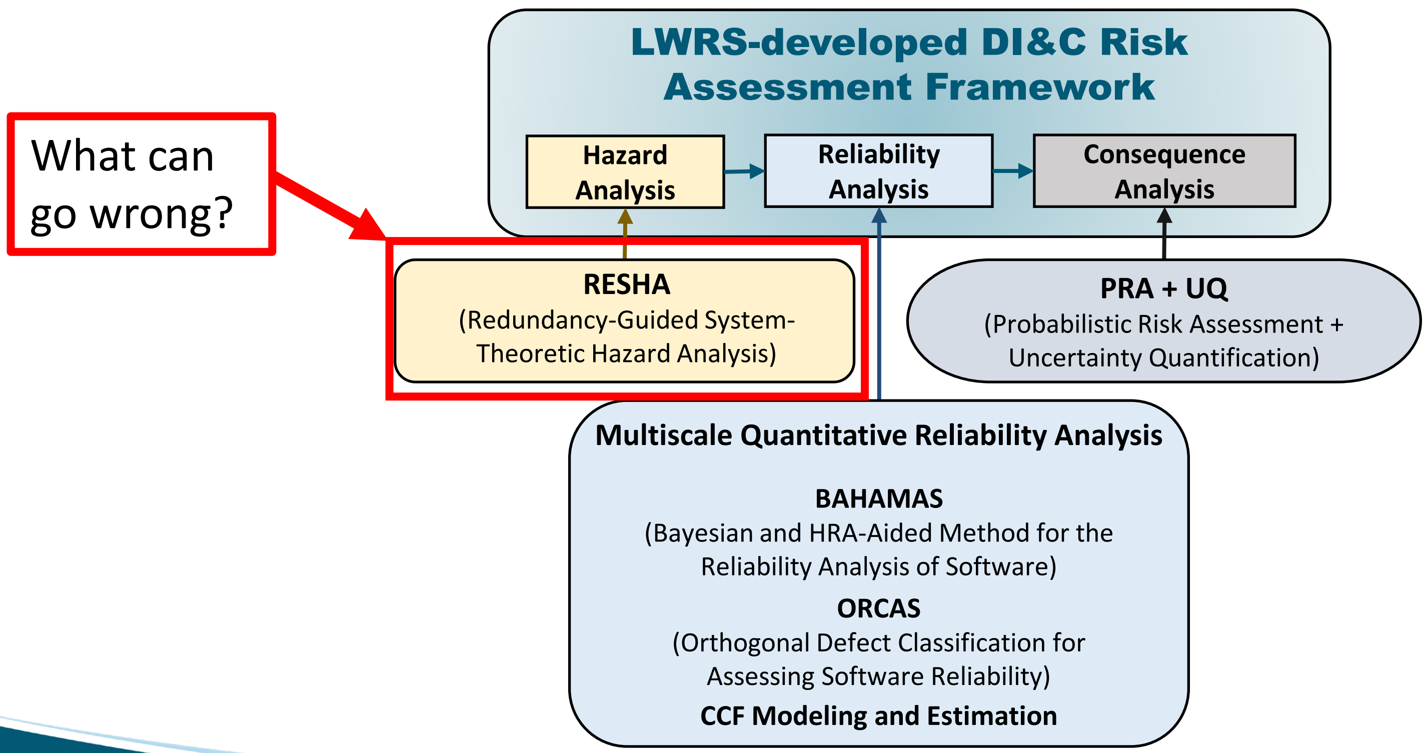
LWRS-developed DI&C Risk Assessment Framework

- **Technical approach:**
  - Develop a risk assessment framework to support modernization efforts for safety-related DI&C systems including all aspects of typical risk:

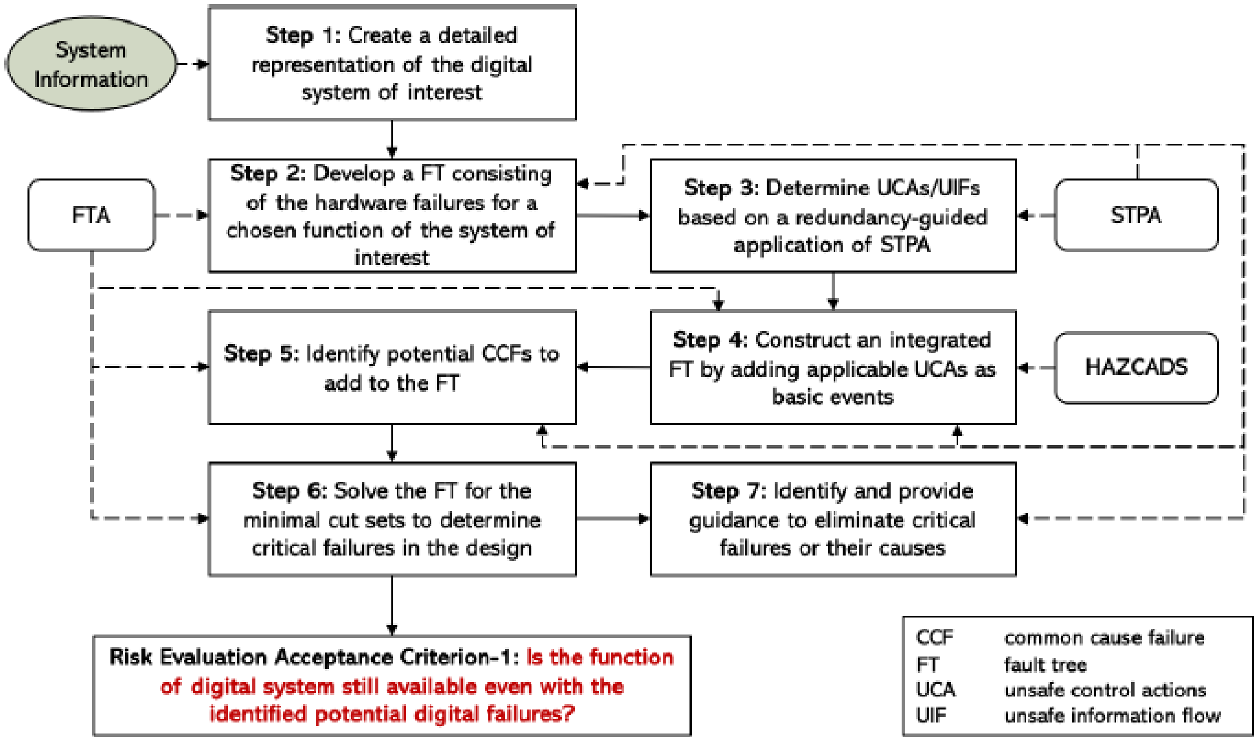
    (1) hazard analysis, (2) reliability analysis, and (3) consequence analysis
  - Evaluate the impact of digital failures at the component level, system level, and plant level



Identifying Software Failure Modes for Risk Assessment



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

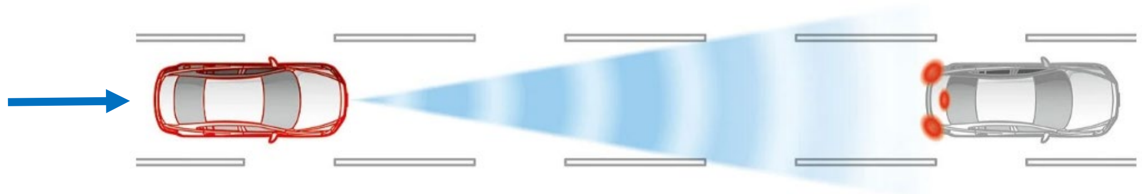
Software Failures in DI&C Systems

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)

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

Auto emergency braking (AEB)



- AEB brake when collision imminent and not user initiated
- How does it "know" when to brake? What could go wrong?

	Type A	Type B	Type C	Type D
Unintentional collision during autopilot on highway	AEB does not detect car in front and crashes.	AEB brakes when squirrel detected, spurious.	AEB brake is late causing minor collision.	AEB brake is too weak causing minor collision.

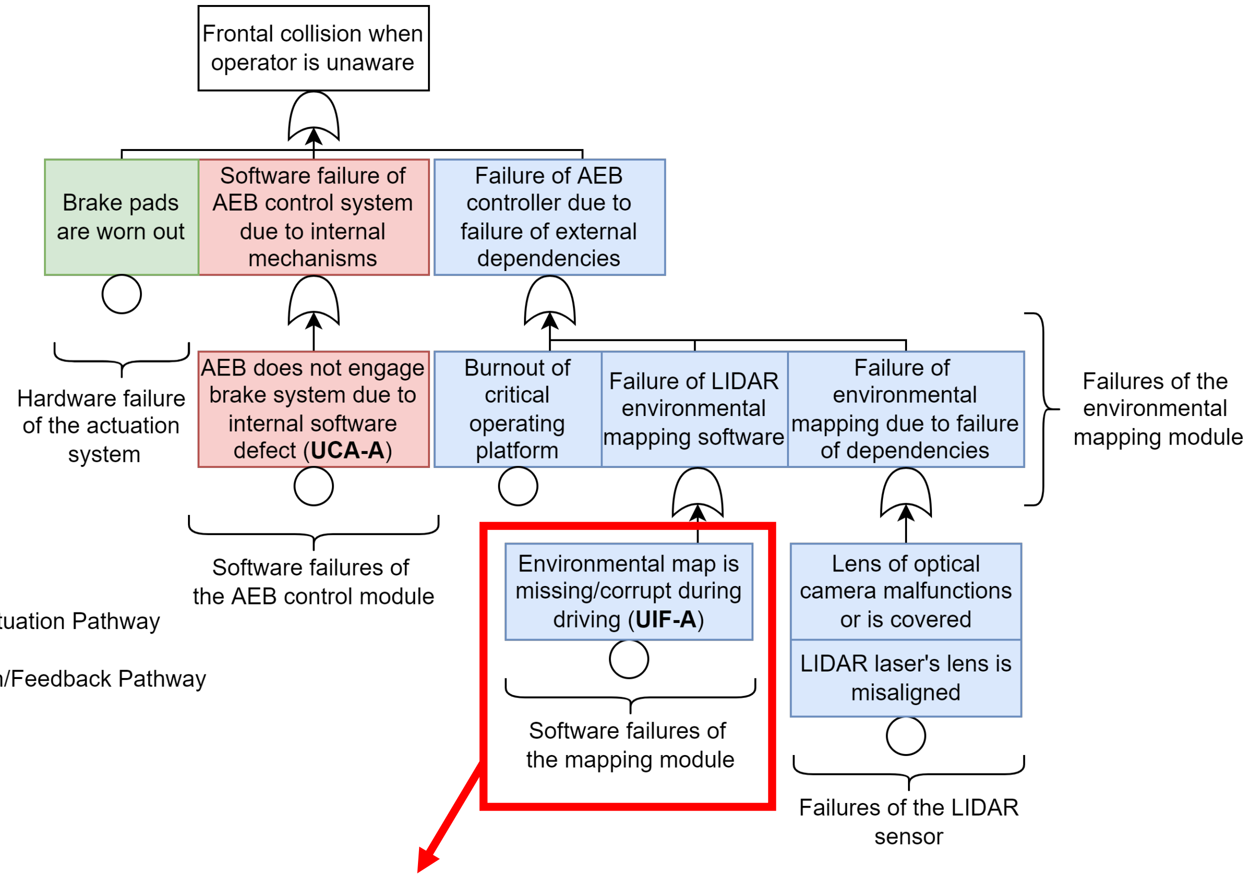
UCAs are undesirable software events and may be modelled as software basic events.

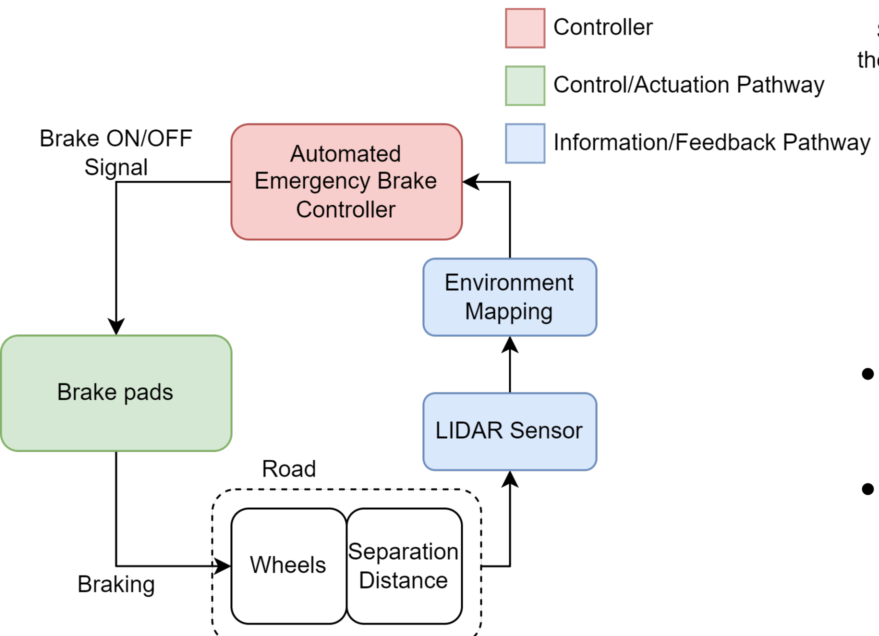
Example FT

<u>Top event:</u>

- Frontal collision when operator is unaware

What software failures will lead to this event?





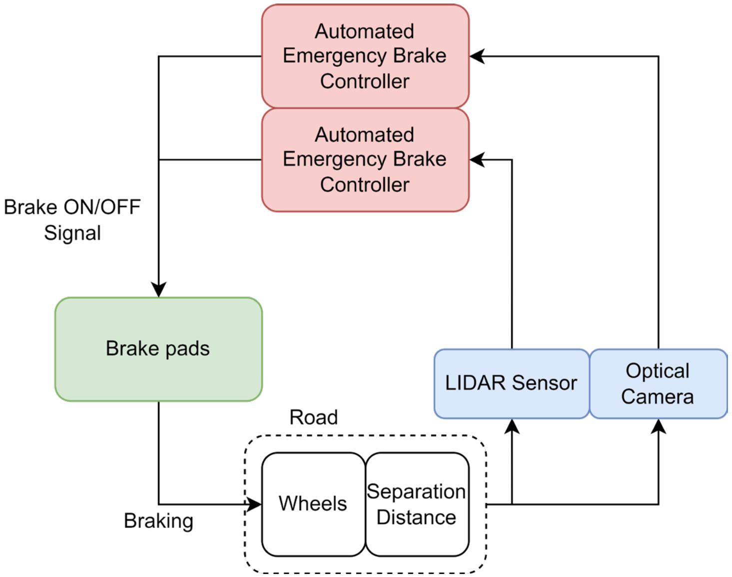
- Missing or incorrect environmental mapping information causes loss
- Controller behaves as it is programmed but a loss still occurs.

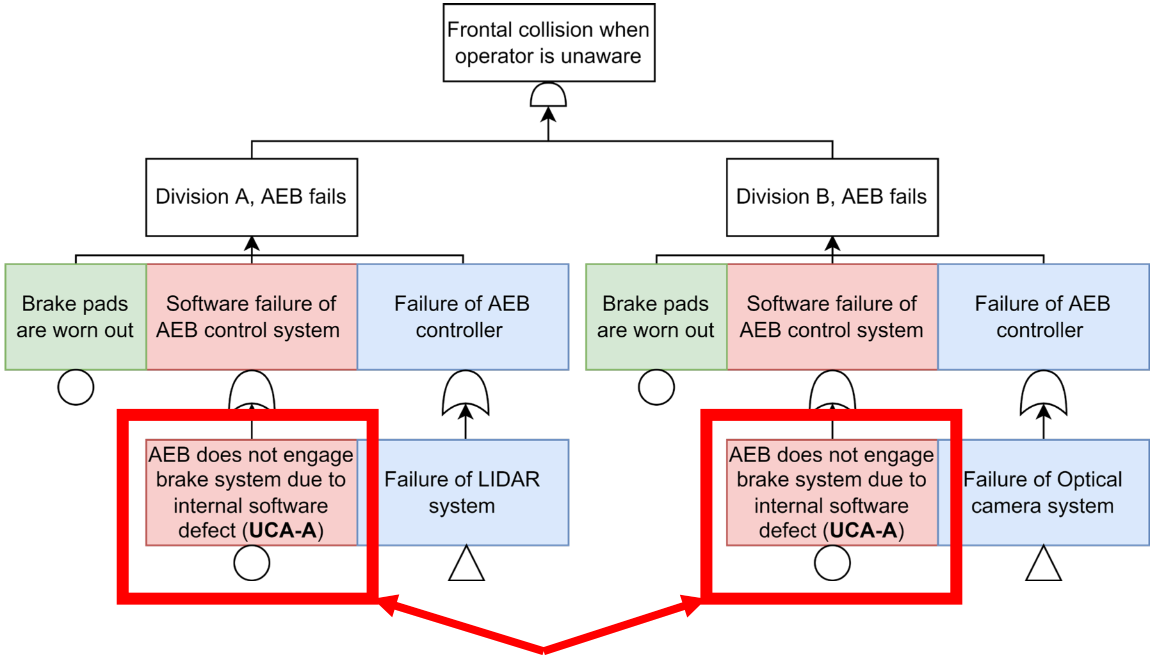
CCF in I&C Systems

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

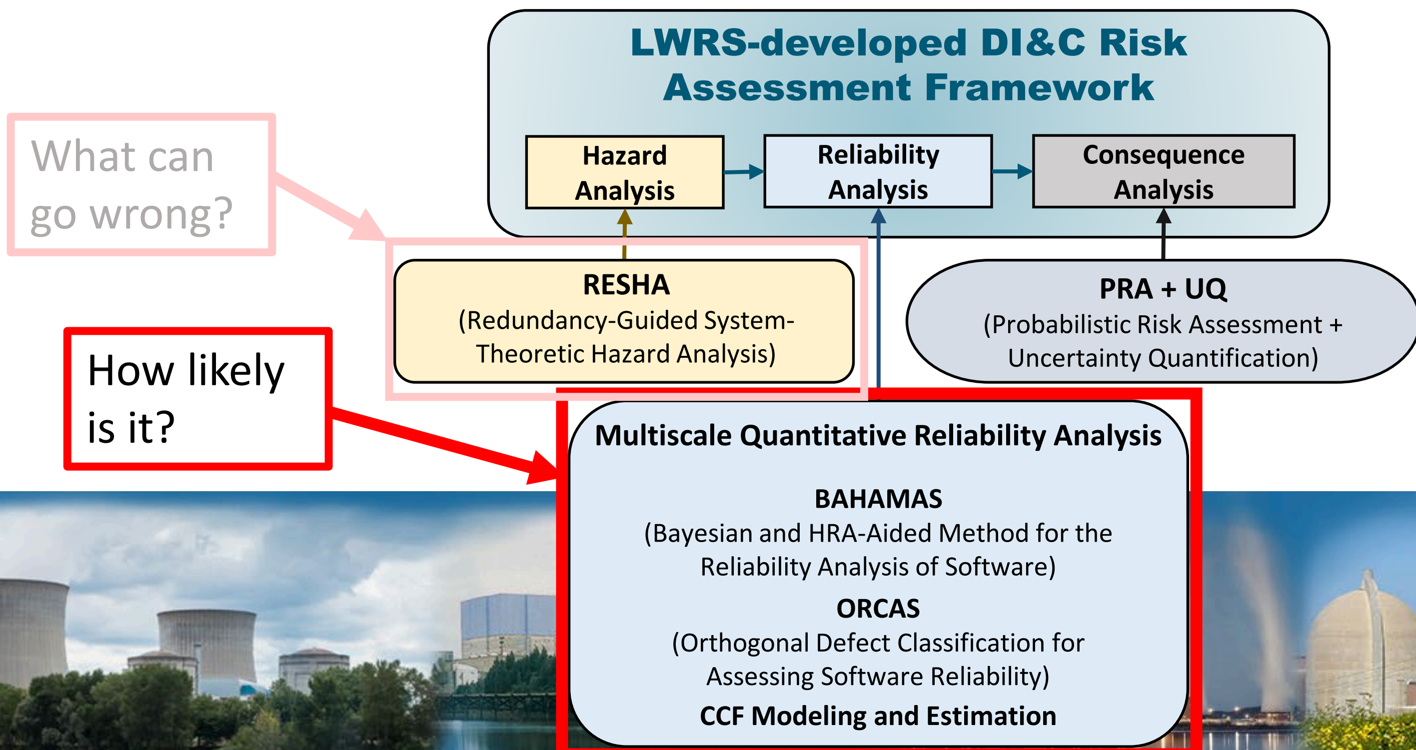




Potential software CCF

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



Discovered Defects & Relationship to UCA/UIF

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:            <p>WinXP / java 1.6 / svn @ 921110 trunk</p>Dates:          Created: Tue, 9 Mar 2010
20:44:31 +0000          Resolved: Tue, 9 Mar 2010 21:03:58 +0000

Description:bin\cassandra-cl.bati -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?

Discovered Defects & Relationship to UCA/UIF

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)

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

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.**

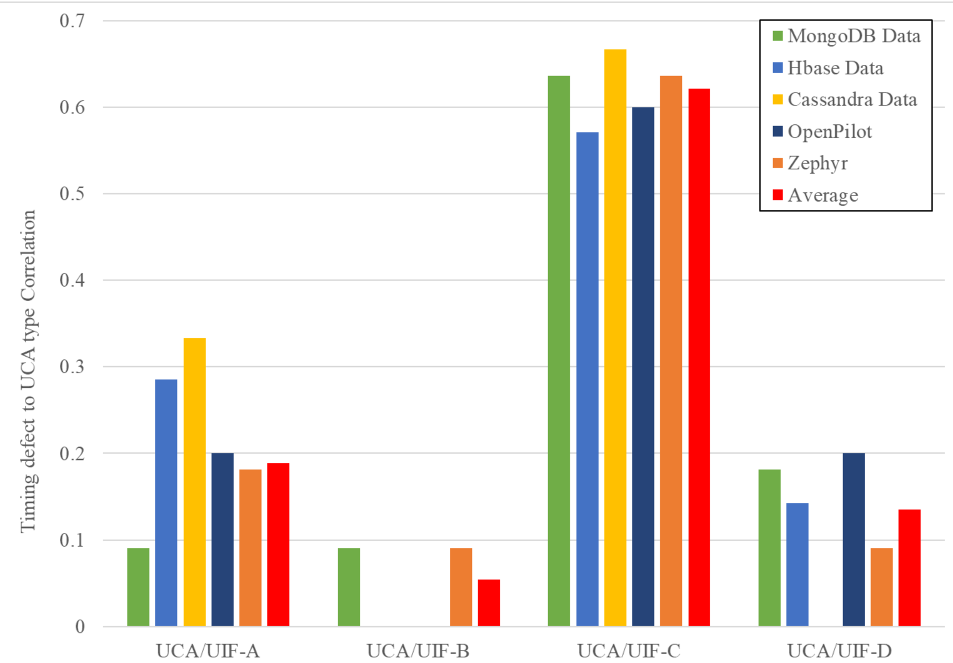
Data Collection & Correlation

Defect Data Correlation:

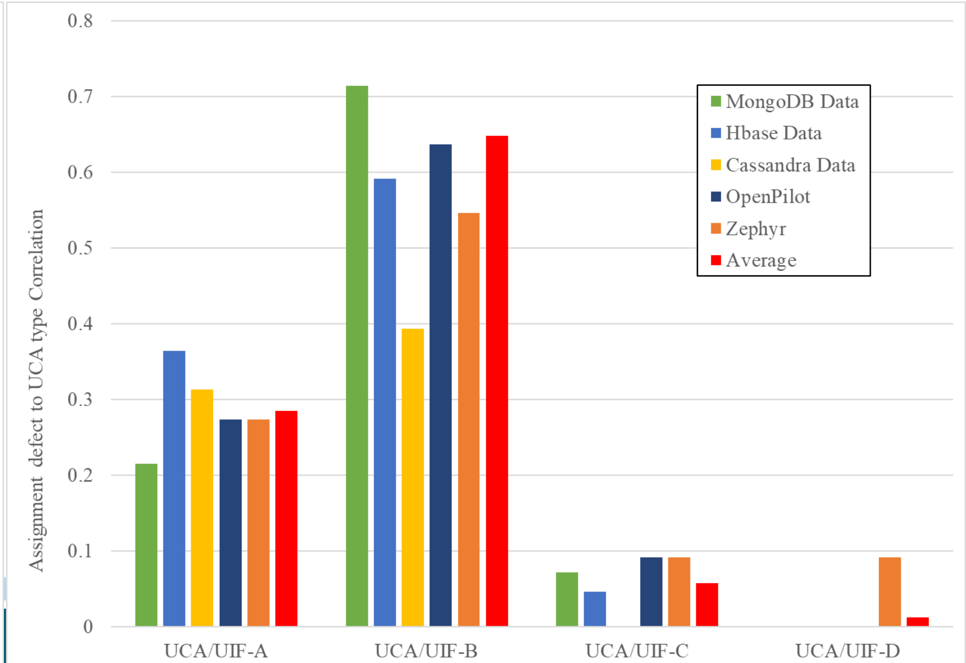
- 3526 defects were collected from 3 database management software, 1 autonomous car software, and 1 microcontroller operating system
  - MongoDB, Cassandra, Hbase, OpenPilot, Zephyr (open source Github repositories)

Defects with <u>shared</u> characteristics cause the same UCA/UIF software failure mode

Timing Defect Class & UCA/UIF Correlation by Dataset



Assignment Defect Class & UCA/UIF Correlation by Dataset



Data Collection & Correlation

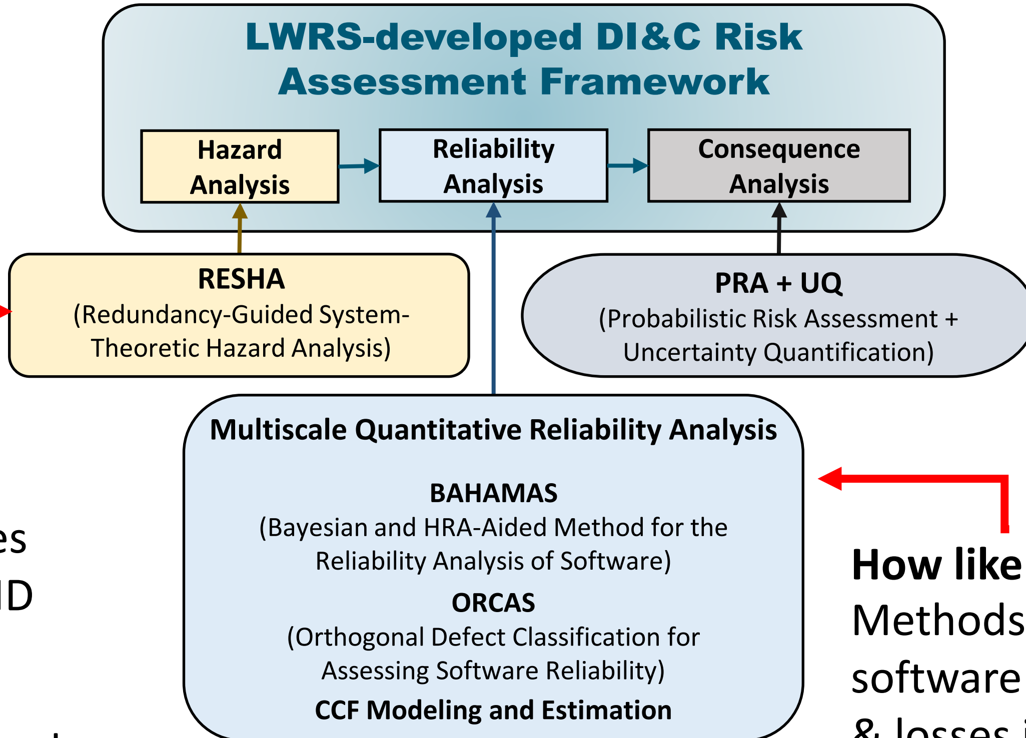
1. The relationship between defect class detected and UCA caused holds about true for 5 different pieces of software.

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;

Defect Class	**UCA-A**	**UCA-B**	**UCA-C**	**UCA-D**
Algorithm Defect	0.331	0.137	0.339	0.194
Assignment Defect	0.284	0.648	0.057	0.011
Checking Defect	0.478	0.262	0.241	0.191
Function Defect	0.385	0.24	0.24	0.135
Timing Defect	0.189	0.054	0.622	0.135

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



What can go wrong?
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

Sustaining National Nuclear Assets

http://lwrs.inl.gov

Failure Mechanisms Leading to UCAs

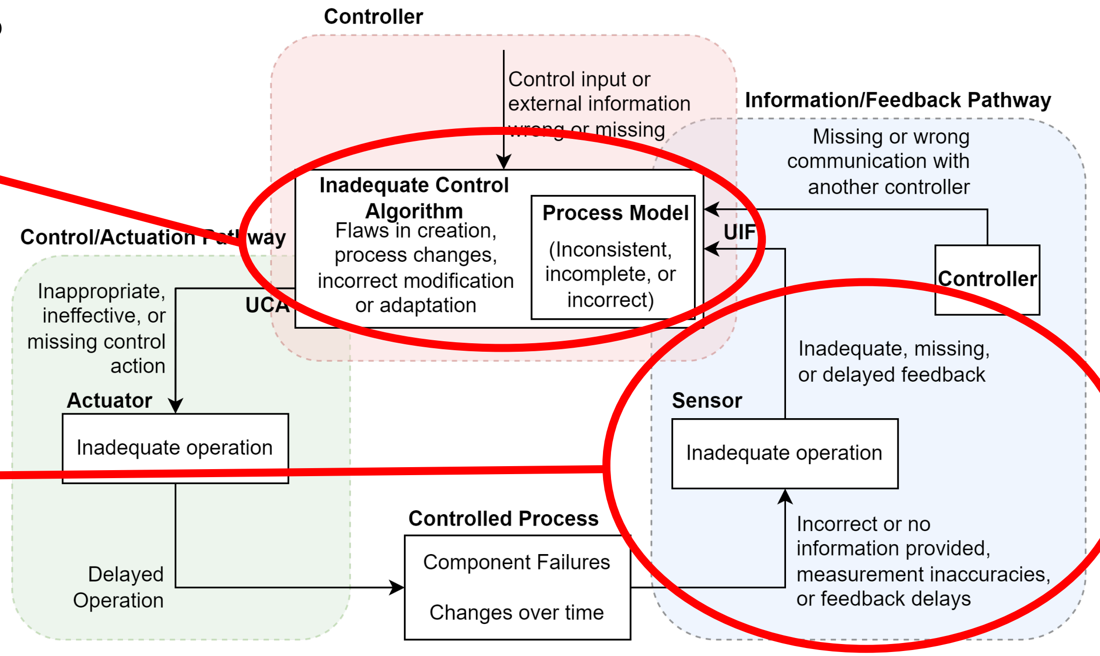
What are the causes of software failure?

- <u>Internal mechanisms:</u>
  - Inadequate control algorithm
    - Action
  - Inconsistent process model
    - Belief

Related to software of controller

- <u>External mechanism:</u>
  - Incorrect/inadequate feedback
    - Awareness

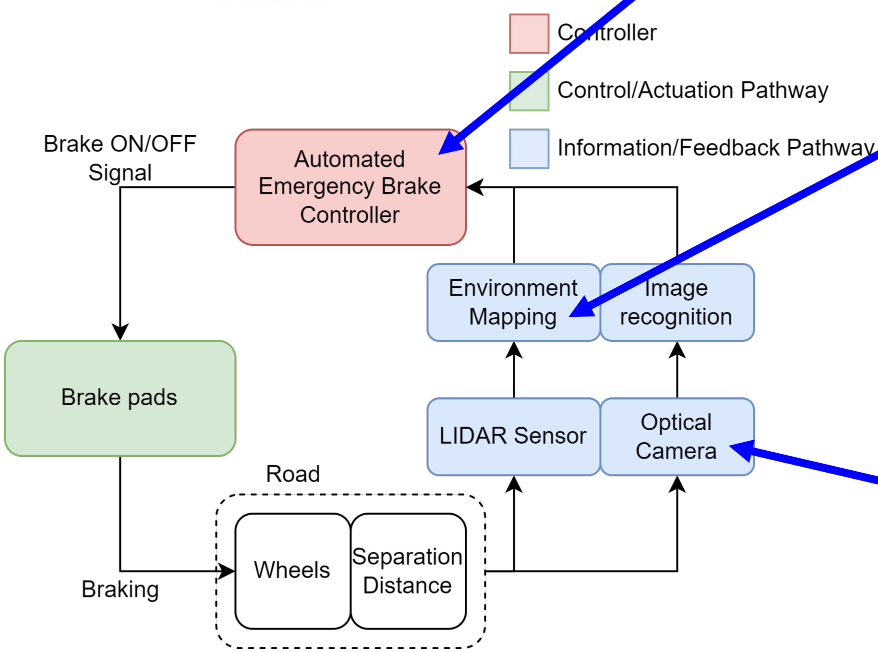
Related to upstream dependencies

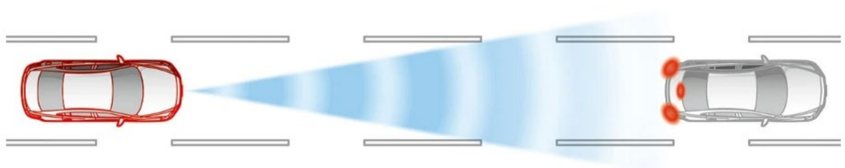


	Type A	Type B	Type C	Type D
Unsafe Control Action (UCA)	Control action not provided when required context.	Control action provided when not required causing hazard.	Control action is early, late, or out-of-order.	Control action is stopped too soon or applied too long.
Unsafe Information Flow (UIF)	Feedback is missing when required context.	Feedback is provided when not required causing hazard.	Feedback is early, late, out-of-sync, or out-of-order.	Feedback value is too low, too high, NaN, or Inf.

UIFs are introduced as software failures in the information systems

Example





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 <u>spuriously</u>

Scenario 3:

- Optical camera is dirty, and LIDAR has height limit.

Result:

- Optical camera is blind. AEB does NOT engage, and a collision occurs.

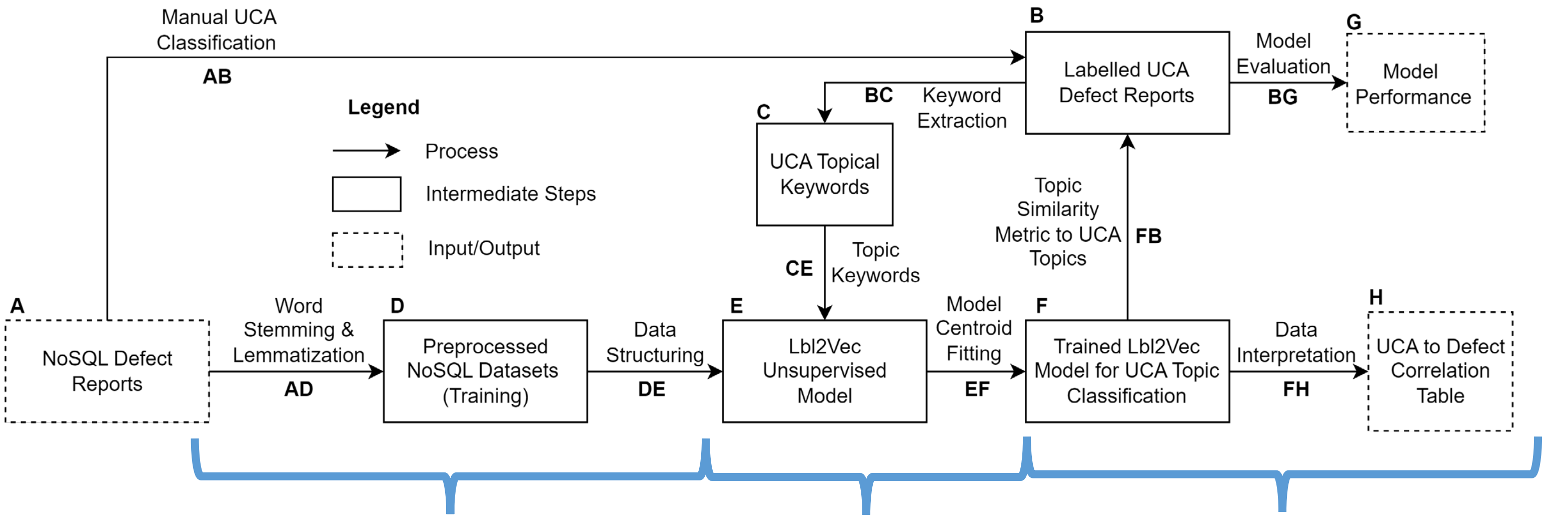
UCAs technically cover all loss scenarios listed BUT:

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

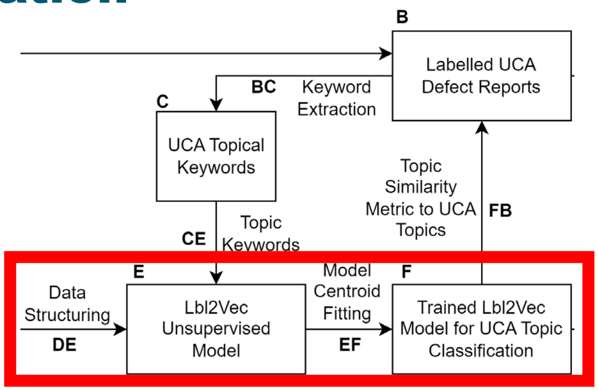
Training based on:

- Topic selection
- Keyword selection

Evaluation:

- Development of defect-UCA correlation table
- Kappa agreement value

Unsupervised Topic Classification

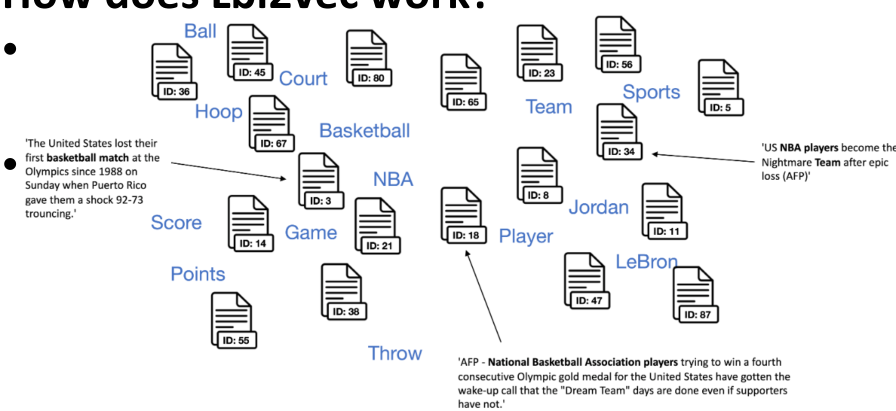


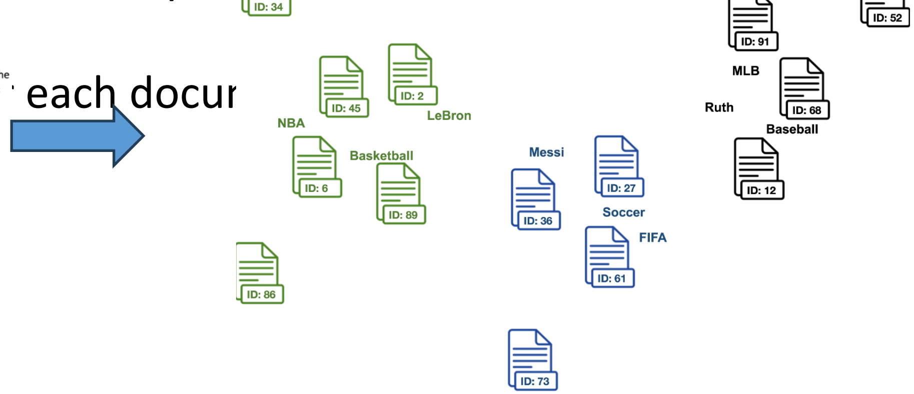
What is Lbl2vec?

- Unsupervised document topic classification and clustering technique

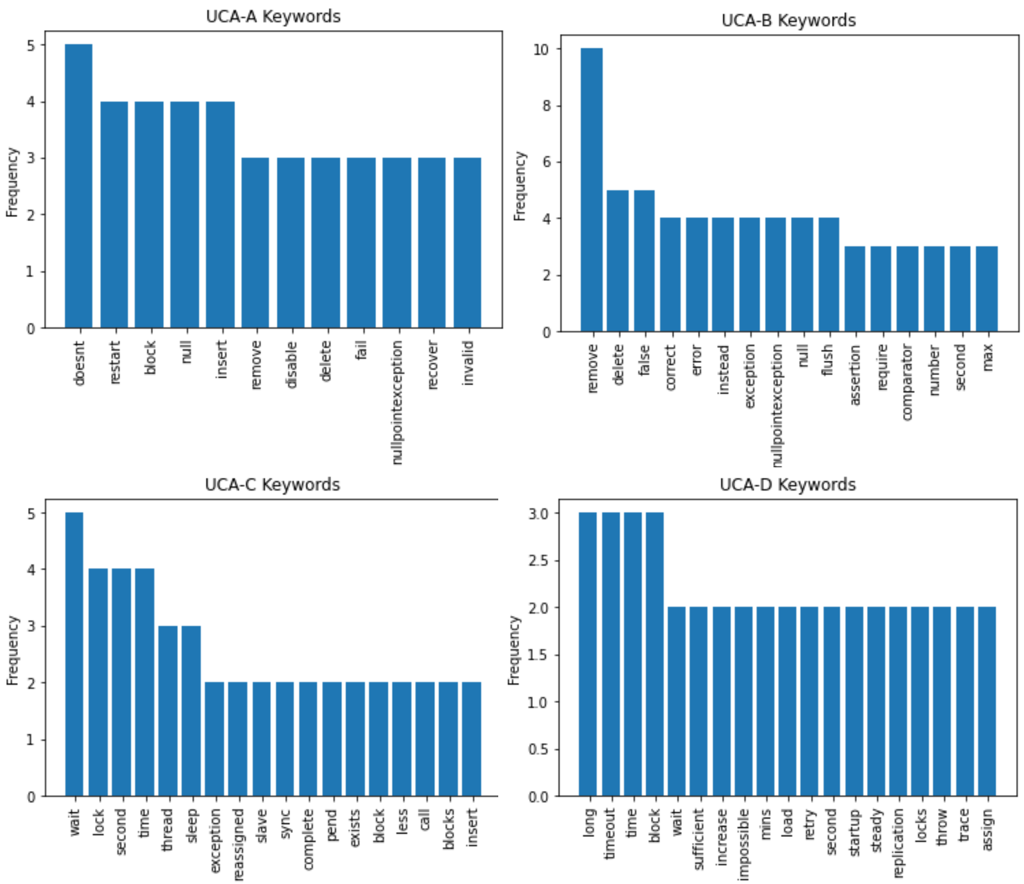
How does Lbl2vec work?

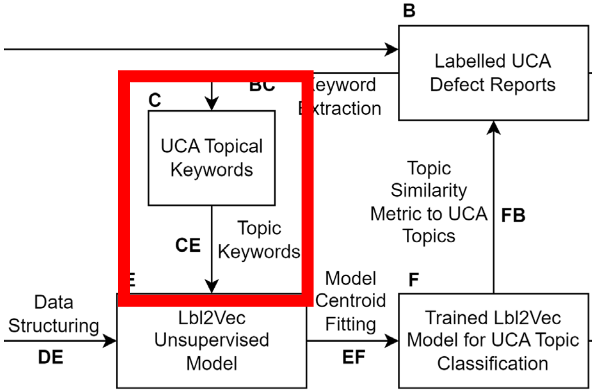
- … **rds**, a topic
- … each docur





Topic & Keyword Selection



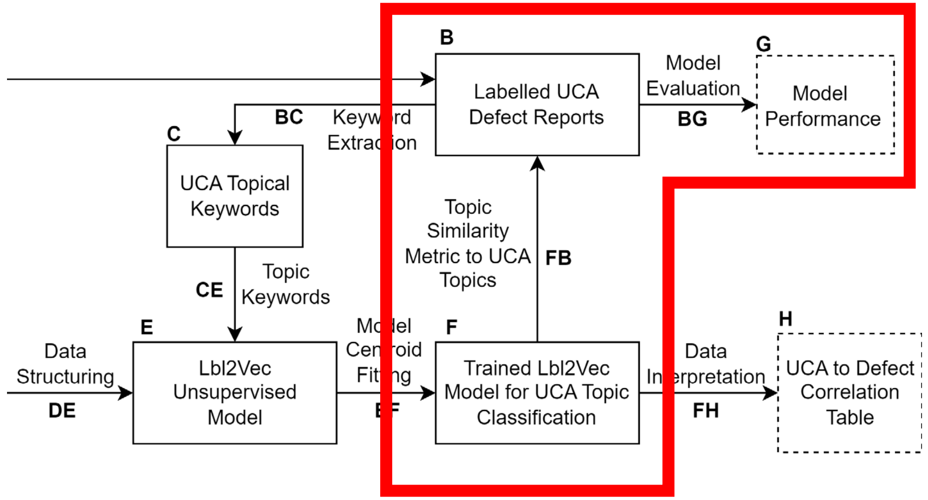


Topic	Keywords
UCA-A	doesnt, restart, block, null, insert, disable, fail, recover, invalid, miss
UCA-B	remove, delete, false, correct, error, instead, exception, nullpointexception, flush, assertion
UCA-C	wait, lock, second, time, thread, sleep, exception, reassigned, slave, sync,
UCA-D	long, timeout, block, sufficient, increase, impossible, load, retry, replication, trace

Similarity Metric for Topic Classification

Lbl2Vec output after training

Report #	Likely Topic	UCA-A	UCA-B	UCA-C	UCA-D
1	UCA-A	0.853	0.263	0.216	0.332
2	UCA-B	0.563	0.610	0.362	0.253
3	UCA-C	0.032	0.015	0.361	0.086



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

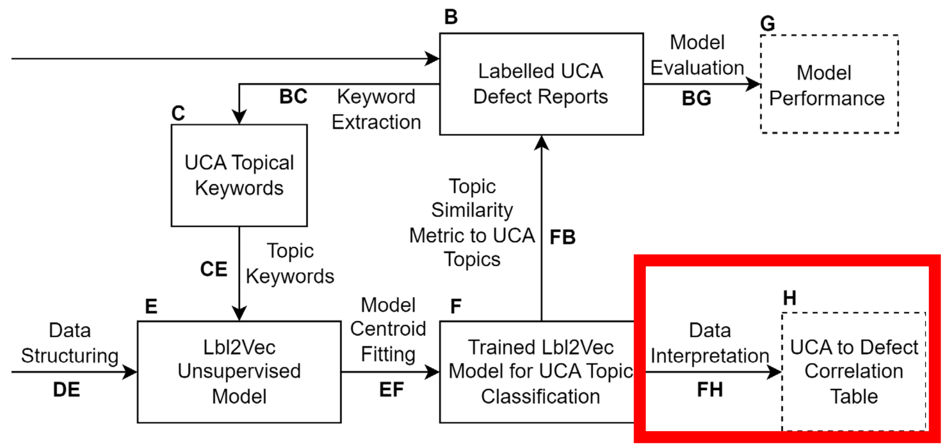
Dataset Name	Overall Kappa	UCA-A	UCA-B	UCA-C	UCA-D	Matching/Total
Hbase	0.636	7	15	1	1	24/33
Cassandra	0.667	4	11	3	0	18/24
MongoDB	0.809	8	6	10	0	24/28
Total	0.702	19/26	32/40	14/14	1/5	66/85

Substantial Kappa agreement w/ researcher labels suggests that Lbl2Vec can automate topic classification

Correlation Table Developed

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



	UCA-A		UCA-B		UCA-C		UCA-D	
Algorithm Defects	0.331	0.442	0.137	0.098	0.339	0.321	0.194	0.138
Assignment Defects	0.284	0.153	0.648	0.504	0.057	0.101	0.011	0.039
Checking Defects	0.305	0.478	0.262	0.338	0.241	0.139	0.191	0.043
Function Defects	0.385	0.336	0.240	0.423	0.240	0.109	0.135	0.132
Timing Defects	0.189	0.057	0.054	0.083	0.622	0.759	0.135	0.101

Manually derived correlation

Lbl2Vec derived correlation

- Within ± 0.05
- Greater > 0.05
- Lesser < 0.05