NTNU
Norwegian University of Science and Technology

Finite-state automata modeling pattern of systems-theoretic process analysis results

RAMS Seminar

Date: Thursday 18.03.2021

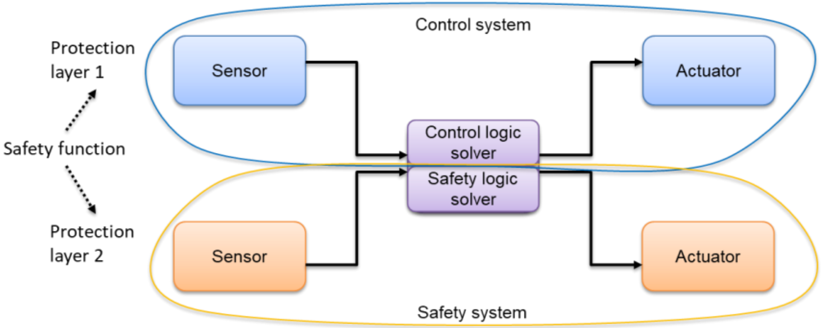
By Nanda Anugrah Zikrullah

Outline

1. Introduction
2. Study case
3. Discussion

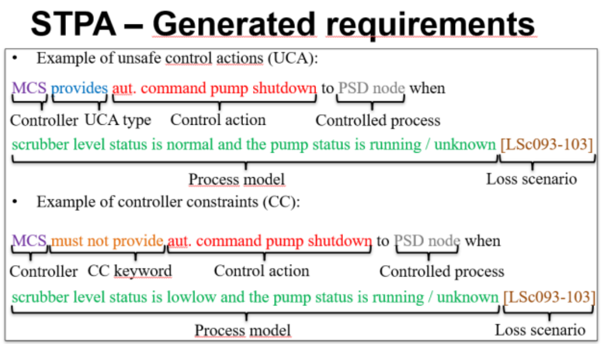
Introduction

From previous work



Integration concept

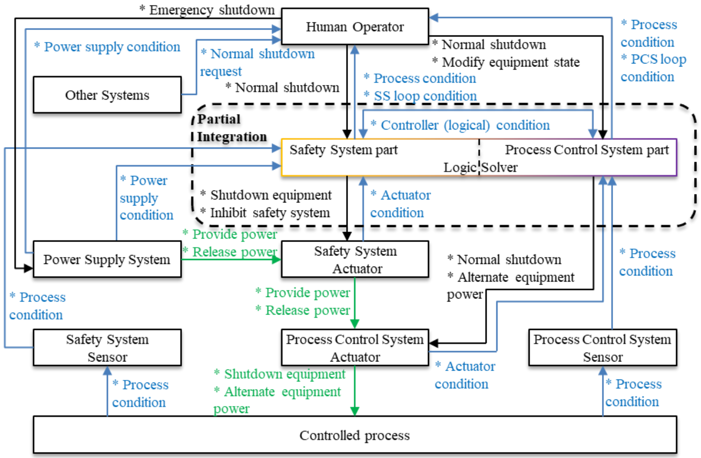
Systems-Theoretic Process Analysis



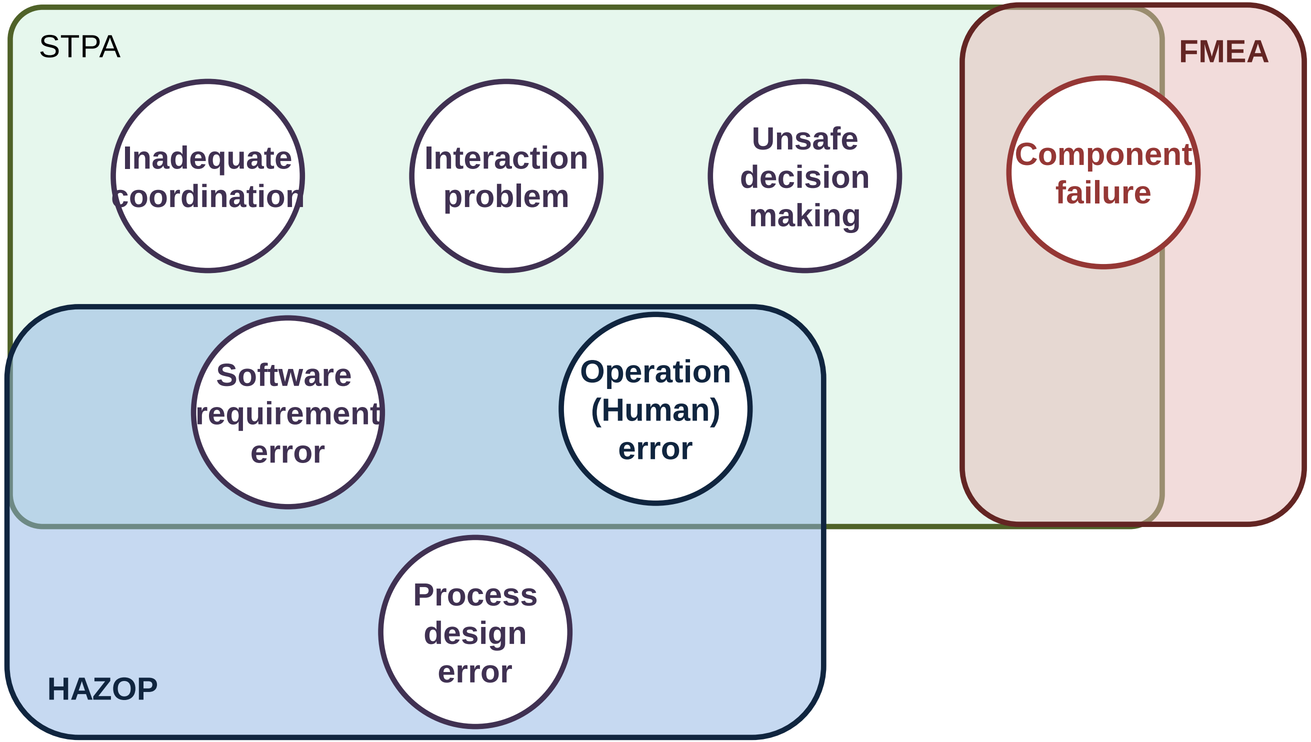
Unsafe control actions (Hazards) & Safety requirements

Systems-Theoretic Process Analysis

- Hazard analysis technique developed by Leveson
- Based on systems theory and systems thinking
- Utilize a control structure model



Why STPA?



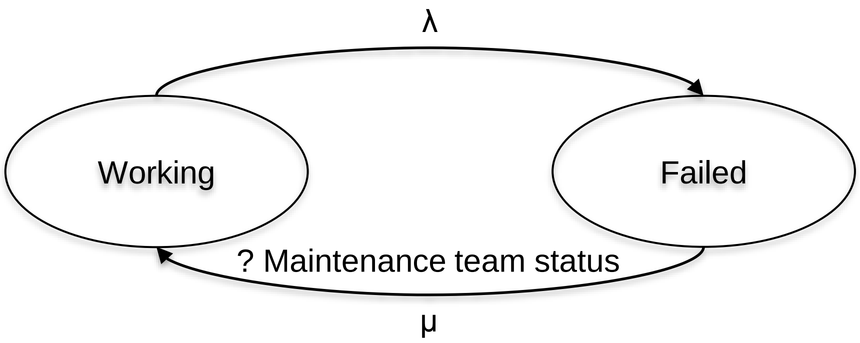
STPA
FMEA
Inadequate coordination
Interaction problem
Unsafe decision making
Component failure
Software requirement error
Operation (Human) error
Process design error
HAZOP

Problem formulation and available contributions

- How can STPA results be used in a decision-making context?
  - Zikrullah et al. (2021) – Generate high-level safety requirements
  - Kim et al.(2020) – Risk-based prioritization of safety measures
  - Zhang et al.(2019) – Incorporating results from STPA into availability calculation
  - Our contribution (under progress) – Incorporating results from STPA to support safety demonstration

Finite state automata (FSA)

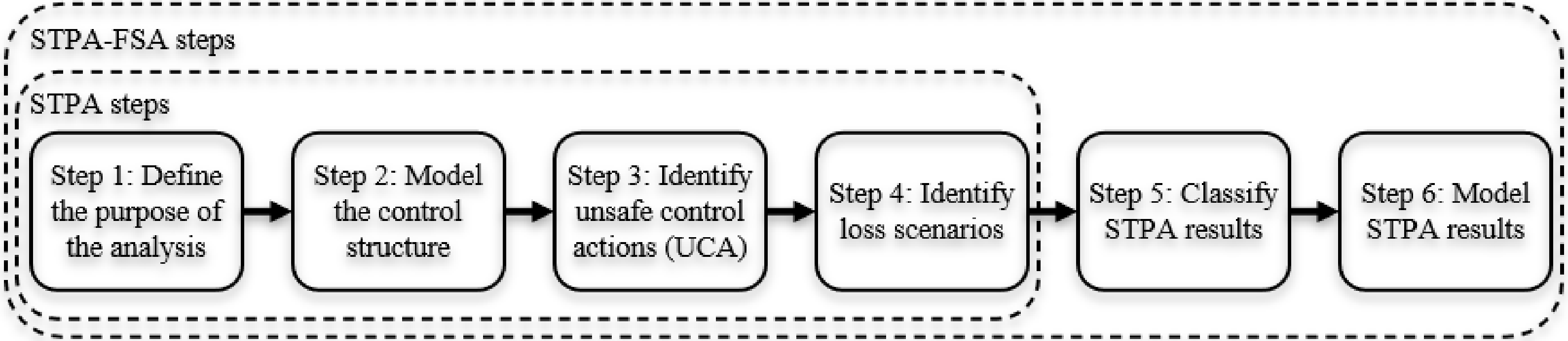
- An approach to model the system as a set of finite states
- Used to quantify system availability or mean time to failures
- Example techniques:
  - Markovian
  - Petri nets
  - Textual-based formal language (e.g., Altarica 3.0)




```
class RepairableComp
    Boolean vsWorking (init = true);
    parameter Real pLambda = 2.3e-4;
    parameter Real pMu = 8.3e-2;
    event evFailure (delay = exponential(pLambda));
    event evRepair (delay = exponential(pMu));
    transition
        evFailure: vsWorking -> {vsWorking := false; pLambda := pLambda * 0.8;}
        evRepair: not vsWorking -> vsWorking := true;
    Boolean input, output (reset = false);
    assertion
        output := if vsWorking then true else false;
end

block System
    RepairableComp C1 (pLambda = 2.3e-4, pMu = 8.3e-2);
    RepairableComp C2 (pLambda = 1.7e-4, pMu = 4.2e-2);
    observer Boolean P3 = C1.output and C2.output;
    observer Boolean P2 = not C1.output and C2.output;
    observer Boolean P1 = C1.output and not C2.output;
    observer Boolean P0 = not C1.output and not C2.output;
end
```

STPA-FSA modeling approach



Examples of STPA result

UCA example:

- UCA001. Controller xxx *does not provide* control action xxx to the controlled process during the condition xxx [H1]

UCA classification

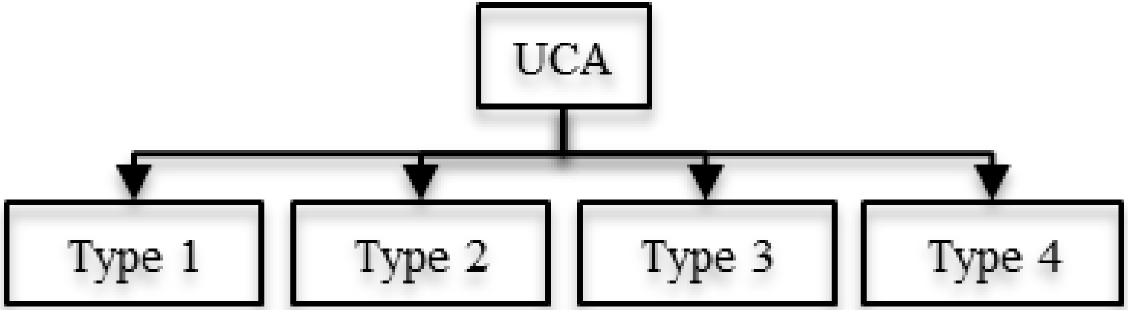
Classification of UCAs:

1.) not providing the control action during a specific condition,

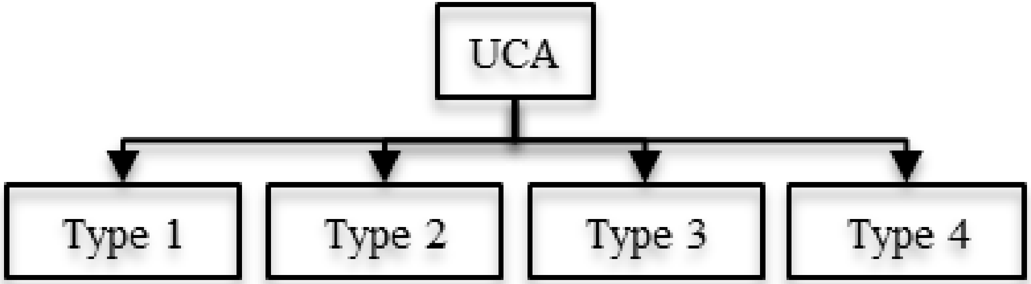
2.) providing unnecessary control action (leading to hazard),

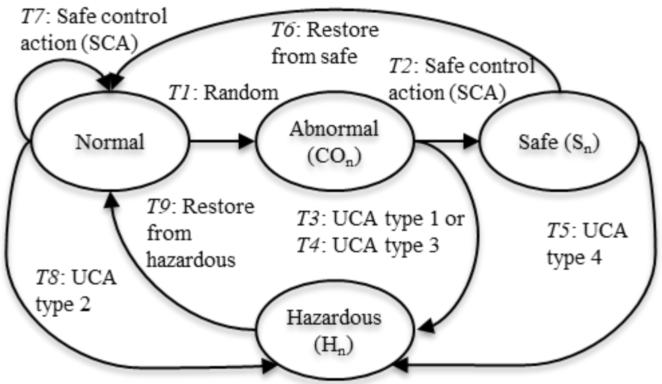
3.) providing a potentially safe control action but too early, too late, or in the wrong order,

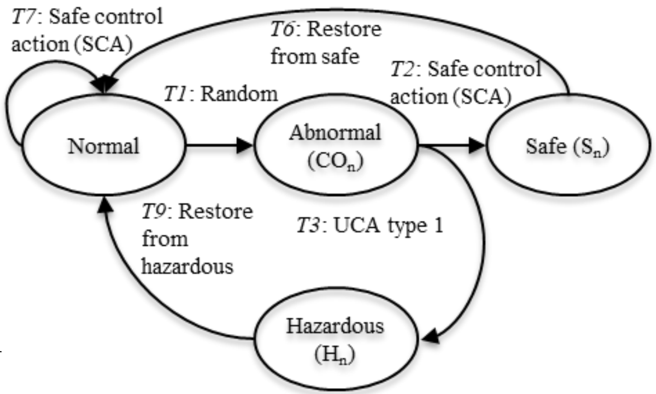
4.) the (continuous) control action lasts too long or is stopped too soon.



(Generic) Controlled process model







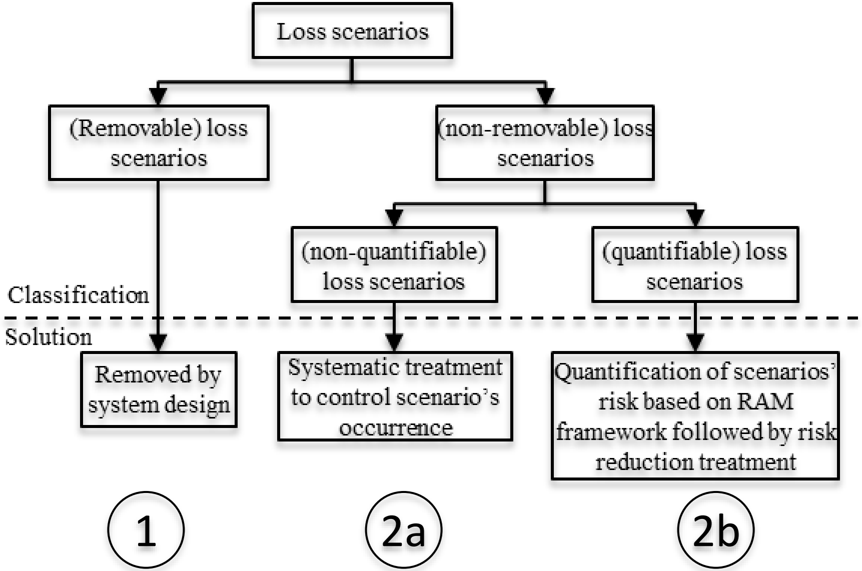
UCA Type 1

Examples of STPA result

Loss scenario example:

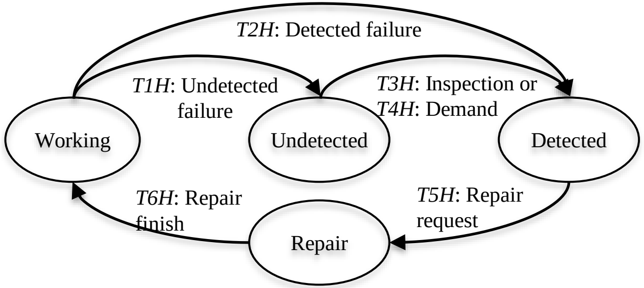
- LSc001. Coupling of *hardware failure* in component xxx and *systematic failure* in component xxx results into UCA001.

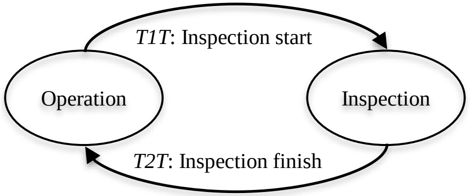
Loss scenario classsification

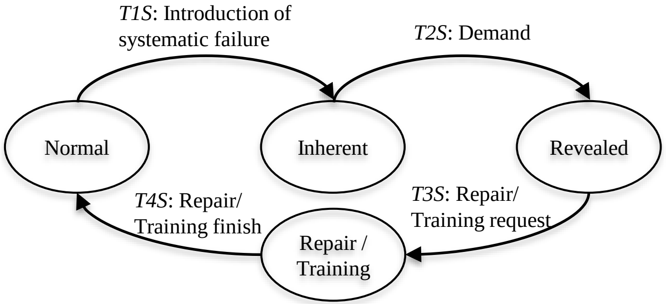


(Generic) Control element model for single failure type

I. Single failure
   1. Random hardware failure (RHF)
      a) Detected
      b) Undetected
   2. Systematic failure
      a) Software
         i. Multiple occurrence. Reappearance follows a stochastic behavior
         ii. Single occurrence. Can be removed by system design *(cannot be modelled)*
      b) Human (the occurrence follows a stochastic behavior)

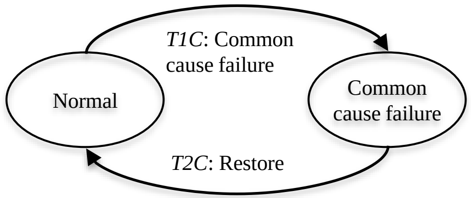






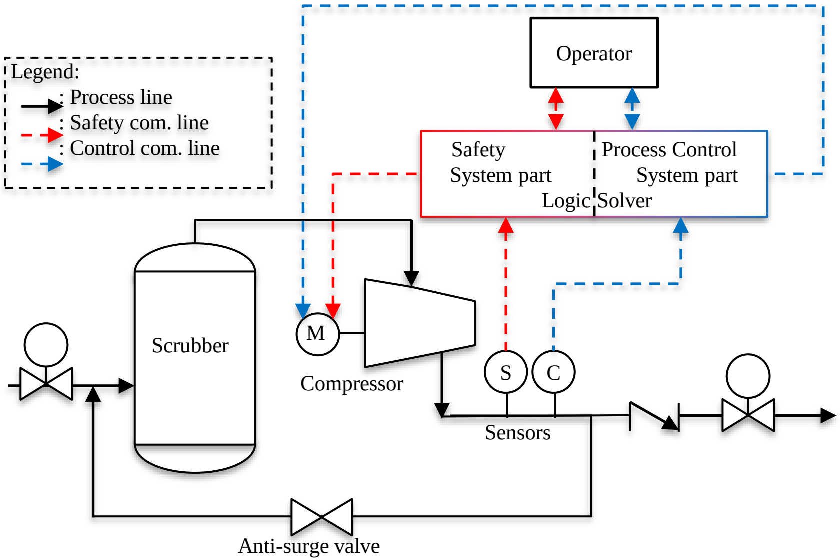
Control element model for multiple failure type

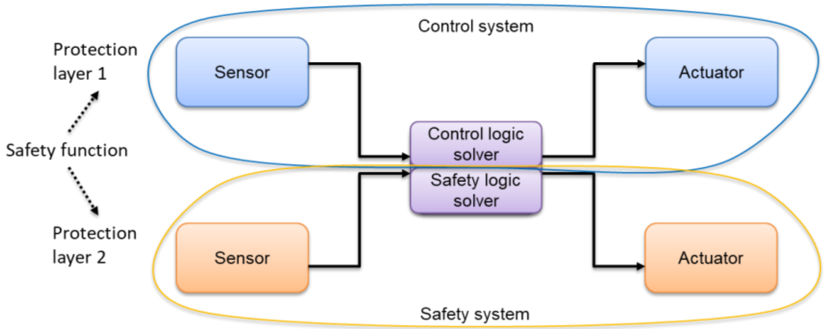
I. Multiple failure
   1. Common cause failure
   2. Cascading failure -> Utilize combination of single failure type model



Study case

Subsea compression system schematic





Example of STPA results

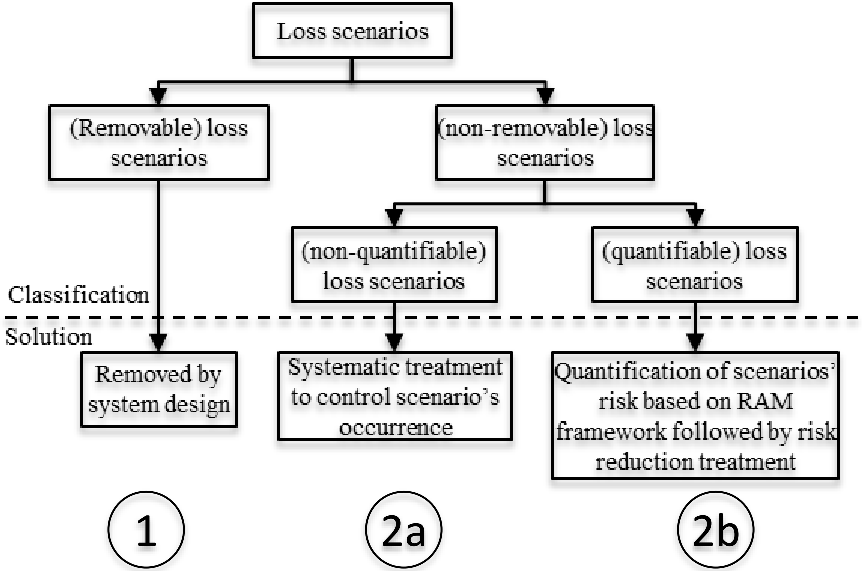
- UCA22. SS part of the logic solver must provide Shutdown equipment command to SS actuator when The gas temperature is very high and the compressor is running [H2]

- Loss scenarios list:

LSID	Scenario	Treatment
LS104	Erroneous information from the SS sensor results in inaccurate information processed at the controller	2b
LS105	Component failure of the SS actuator results in system inability to process the control command	2b
LS106	Component failure of the SS sensor results in inaccurate information processed at the controller	2b
LS107	Problem in the transmitted information (e.g., erroneous, delay) results in inability to transfer information/command in the control loop	2a
LS108	Component failure of the communication transmission system results in inability to transfer information/command in the control loop	2b
LS109	Algorithm flaw on the SS part of the logic solver is a design problem that cause unintended functionality at the controller	1
LS110	Component failure of the PCS/SS logic solver (shared) results in incorrect administration of control action	2b
LS111	Unintended overwrite from PCS to SS in the logic solver is a design problem that cause unintended functionality at the controller	2a/2b*
LS112	Resource sharing problem between PCS and SS in the logic solver is a design problem that cause unintended functionality at the controller	1

* Depending on data availability

Loss scenario classsification



NTNU

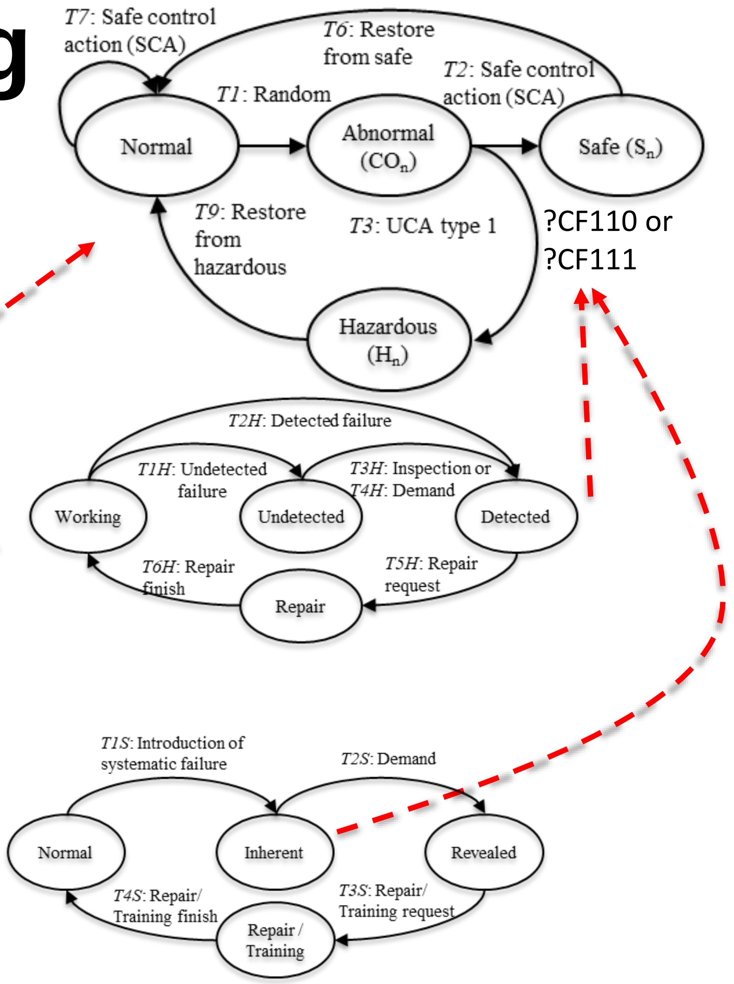
Scenario's modeling

UCA example:

- UCA22. SS part of the logic solver must provide Shutdown equipment command to SS actuator when The gas temperature is very high and the compressor is running [H2]

Loss scenario example:

- LS110. Component failure of the PCS/SS logic solver (shared) results in incorrect administration of control action
- LS111. Unintended overwrite from PCS to SS in the logic solver is a design problem that cause unintended functionality at the controller



Source code for implementation

Demand.alt

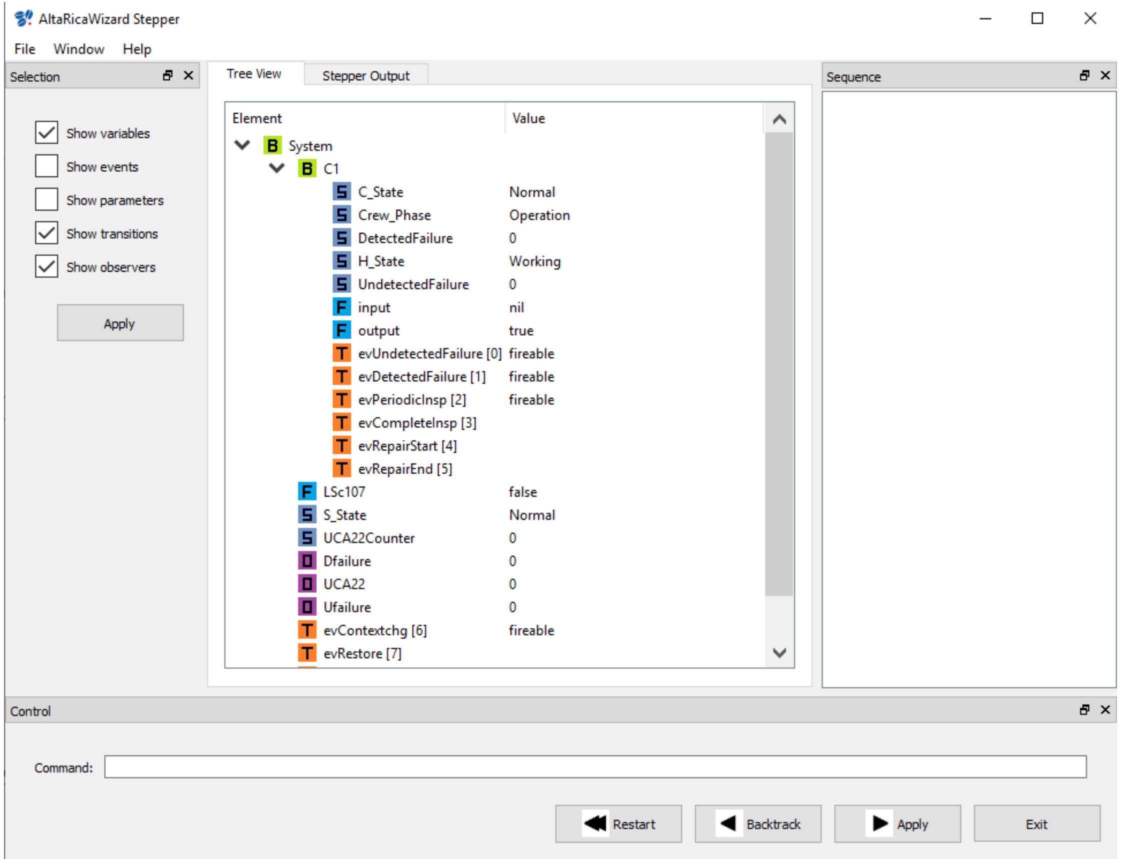
```
domain SystemState {Normal, Context, Safe, Hazardous, Loss}
domain HardwareState {Working, Undetected, Detected, Repair}
domain SystematicState {Working, Systematic, Revealed, Repair}
domain ElementState {Normal, Demand}
domain HumanState {Normal, Systematic, Learning}
domain Phase {Operation, Inspection}

class CompWRHF
    HardwareState H_State (init = Working);
    ElementState C_State (init = Normal);
    Phase Crew_Phase (init = Operation);
    parameter Real pLambdaU = 1e-6;
    parameter Real pLambdaD = 1e-4;
    parameter Real pMu = 1.25e-1;
    parameter Integer pInspectionPeriod = 4380;
    parameter Integer pInspectionDuration = 0.0;
    event evUndetectedFailure (delay = exponential(pLambdaU));
    event evDetectedFailure (delay = exponential(pLambdaD));
    event evUndDemand, evDetDemand (delay = 0, hidden = true);
    event evPeriodicInsp (delay = pInspectionPeriod);
    event evCompleteInsp (delay = pInspectionDuration);
    event evRepairStart (delay = 0);
    event evRepairEnd (delay = exponential(pMu));
    transition
        evUndetectedFailure:    H_State == Working and Crew_Phase == Operation -> H_State := Undetected;
        evDetectedFailure:      H_State == Working and Crew_Phase == Operation -> H_State := Detected;
        evUndDemand:            H_State == Undetected -> {H_State:= Detected; C_State := Demand;}
        evDetDemand:            H_State == Detected -> C_State := Demand;
        evPeriodicInsp:         Crew_Phase == Operation -> Crew_Phase := Inspection;
        evCompleteInsp:         Crew_Phase == Inspection
                            -> {Crew_Phase := Operation ; if H_State == Undetected then H_State := Detected;}
        evRepairStart:          Crew_Phase == Operation and (H_State == Detected) ->  H_State := Repair;
        evRepairEnd:            Crew_Phase == Operation and H_State == Repair
                            -> {H_State := Working; C_State := Normal;}
    Boolean input, output (reset = false);
    assertion
        output := if H_State == Working then true else false;
end

class Human
    HumanState Hu_State (init = Normal);
    ElementState C_State (init = Normal);
    Integer SystematicFailureCounter (init = 0);
    parameter Real pLearningRate = 0.8;
    parameter Real pLambdaS = 1e-2;
    parameter Real pMu = 1.25e-1;
    event evSystematicFailure1 (delay = exponential(pLambdaS));
    event evSystematicFailure2 (delay = exponential(pLambdaS * pLearningRate));
    event evHumDemand (delay = 0, hidden = true);
    event evLearningStart (delay = 0);
    event evLearningEnd (delay = exponential(pMu));
    transition
        evSystematicFailure1:   Hu_State == Normal and SystematicFailureCounter == 0
                            -> {Hu_State := Systematic; SystematicFailureCounter := SystematicFailureCounter + 1;}
```

- Altarica 3.0
  - Library module for **controlled process** model and **control element** models

Results (Stepwise simulation)



For verification of system behavior

Parameters for simulation (from PDS (2021) and experts judgment

Parameter	Value	Probability distribution
SS Sensor failure rate	DU = 2e-7 /hour	Exponential
SS Sensor erratic reading rate	DD = 4e-7 /hour	Exponential
SS Actuator failure rate	DU = 5e-7 /hour	Exponential
Communication equipment failure rate	DD = 1e-8 /hour (assumption, need discussion)	Exponential
PCS/SS logic solver failure rate	DU = 1.1e-6 /hour; DD = 1.5e-6 /hour	Exponential Exponential
SS software systematic fault introduction rate	Sys= 1e-8 /hour (assumption, need discussion)	Exponential
Repair time	8 hour	Exponential
Repair delay	8 hour	Exponential
Inspection period	once every 6 months	Dirac
Inspection duration	24 hour	Exponential
Frequency of context change	once per year	Exponential
System restoration time	8 hour	Exponential
Simulation time	87,600 hour	n/a
Number of simulations	500,000	n/a

NTNU

S afety 4.0

Results (Stochastic simulation)

meta-data										
	number-of-runs	500000								
	seed	12345								
	mission-time	87600.0								
	model-name	System								
	file-name	C:/Users/nandaa/Google Drive/RAMS/PHD/Research/Altarica/Juntao/System.gts								
	start-time	Tue Feb 16 12:48:19 2021								
	end-time	Tue Feb 16 12:48:22 2021								
	steps min	1								
	steps mean	22.6083								
	steps max	87								
	tool version	1.0.0								
	compiler version	1.0.0								
observer	SCA22	type	Boolean							
	indicator	SCA1	type	number-of-occurrences	value	TRUE				
		date	87600.0							
			sample-size	500000						
			mean	2.03938						
			standard-deviation	1.49259						
			confidence-interval	0.95	size	0.0041372	low	2.03524	high	2.04352
observer	UCA22	type	Boolean							
	indicator	UCA1	type	number-of-occurrences	value	TRUE				
		date	87600.0							
			sample-size	500000						
			mean	0.474194						
			standard-deviation	0.620325						
			confidence-interval	0.95	size	0.0017194	low	0.472475	high	0.475913
observer	Ufailure	type	Integer							
	indicator	UFailure	type	value						
		date	87600.0							
			sample-size	500000						
			mean	0.717696						
			standard-deviation	0.70462						
			confidence-interval	0.95	size	0.0019531	low	0.715743	high	0.719649

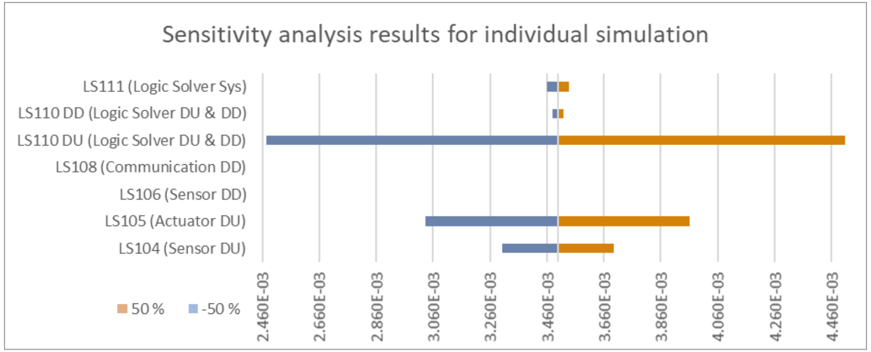
UCAxxx frequency

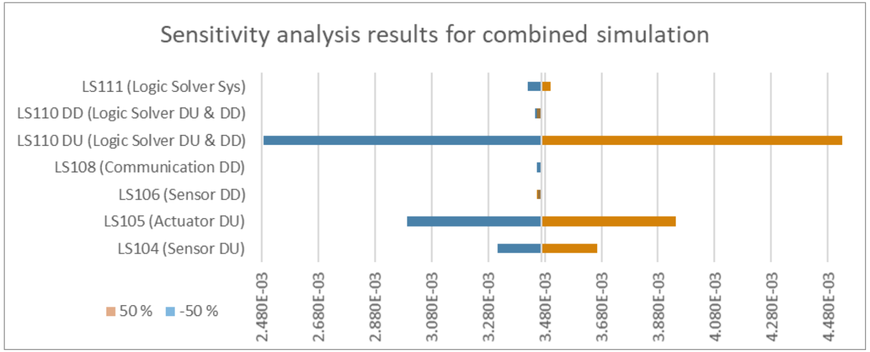
Result tabulation

Loss scenario ID (Causal factor)	LS frequency (/year) (Individual simulation)	Simulation time	LS frequency (/year) (Combined simulation)	Simulation time
LS104 (Sensor)	3.780E-04	9 s	3.778E-04	-
LS105 (Actuator)	9.342E-04	9 s	9.198E-04	-
LS106 (Sensor)	8.200E-06	8 s	1.260E-05	-
LS108 (Communication)	6.000E-07	8 s	4.000E-07	-
LS110 (PCS/SS logic Solver)	2.102E-03	18 s*	2.073E-03	-
LS111 (PCS/SS logic Solver)	7.600E-05	9 s	8.280E-05	-
Total UCA frequency & simulation time	3.499E-03	avg. 60.86 s	3.460E-03	avg. 27.27 s

* two simulations are performed due to contribution from several causal factors (undetected & detected failure)

Sensitivity analysis



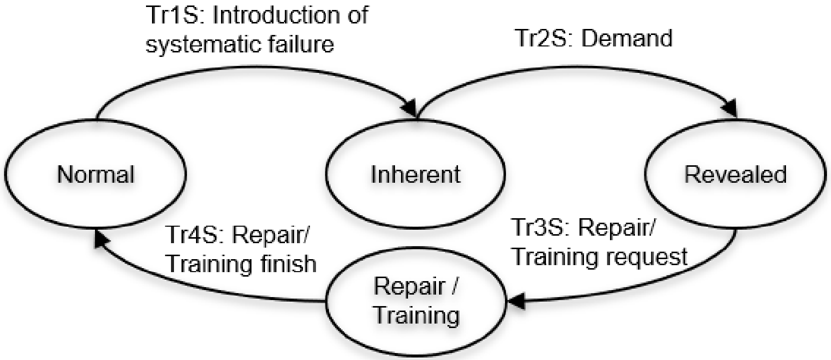


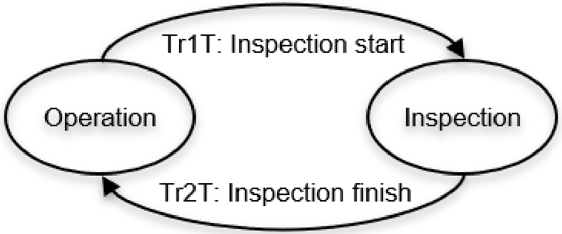
Discussion

Contribution of the new approach

- Capability to model systematic faults
- Aggregation of multiple scenarios into one model (for LSs)
- Improved simulation time ?
- Comparison with traditional quantitative modeling approach
- Prioritization based on quantified value
- Reduction of model uncertainty
- Input for risk assessment method using STPA

Capability to model systematic faults





Aggregation of multiple scenarios into one model (for LS)

Loss scenario ID (Causal factor)	LS frequency (/year) (Individual simulation)	Simulation time	LS frequency (/year) (Combined simulation)	Simulation time
LS104 (Sensor)	3.780E-04	9 s	3.778E-04	-
LS105 (Actuator)	9.342E-04	9 s	9.198E-04	-
LS106 (Sensor)	8.200E-06	8 s	1.260E-05	-
LS108 (Communication)	6.000E-07	8 s	4.000E-07	-
LS110 (PCS/SS logic Solver)	2.102E-03	18 s*	2.073E-03	-
LS111 (PCS/SS logic Solver)	7.600E-05	9 s	8.280E-05	-
Total UCA frequency & simulation time	3.499E-03	avg. 60.86 s	3.460E-03	avg. 27.27 s

* two simulations are performed due to contribution from several causal factors (undetected & detected failure)

Combined simulation results										
	Failure rate value	Individual Contribution	Effect to UCA	Individual frequency (with base value)						Simulation Time (seconds)
				LS104 (2E-7)	LS105 (4E-7)	LS106 (5E-7)	LS108 (1E-8)	LS110 (DU = 1.1E-6 & DD = 1.5E-6)	LS111 (1E-08)	
LS104 (Sensor DU) -50%	1.000E-07	1.810E-04	3.315E-03	n/a	9.330E-04	1.300E-05	2.000E-07	2.103E-03	8.900E-05	27.00
LS104 (Sensor DU) base	2.000E-07	3.778E-04	3.460E-03	3.778E-04	9.198E-04	1.260E-05	4.000E-07	2.073E-03	8.280E-05	27.00
LS104 (Sensor DU) +50%	3.000E-07	5.620E-04	3.667E-03	n/a	9.460E-04	1.500E-05	2.000E-07	2.071E-03	7.900E-05	27.00

Improved simulation time (?)

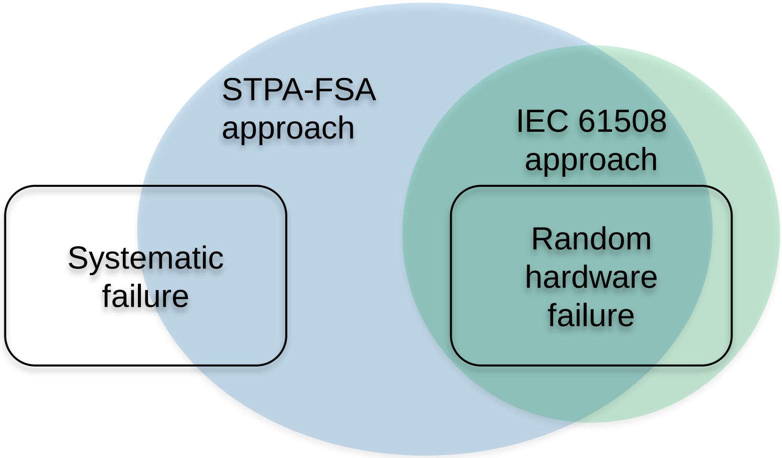
Comparison with Zhang et al. (2019)

Failure rate (/hour)	Juntao's result (UCA freq./year)	Simulation time	My result (UCA freq./year)	Simulation time
5e-6	3.3e-4	~44 minutes	2.5e-2	3 seconds
1e-5	5.7e-4	~44 minutes	4.7e-2	3 seconds
1.5e-5	7.9e-4	~44 minutes	6.6e-2	3 seconds

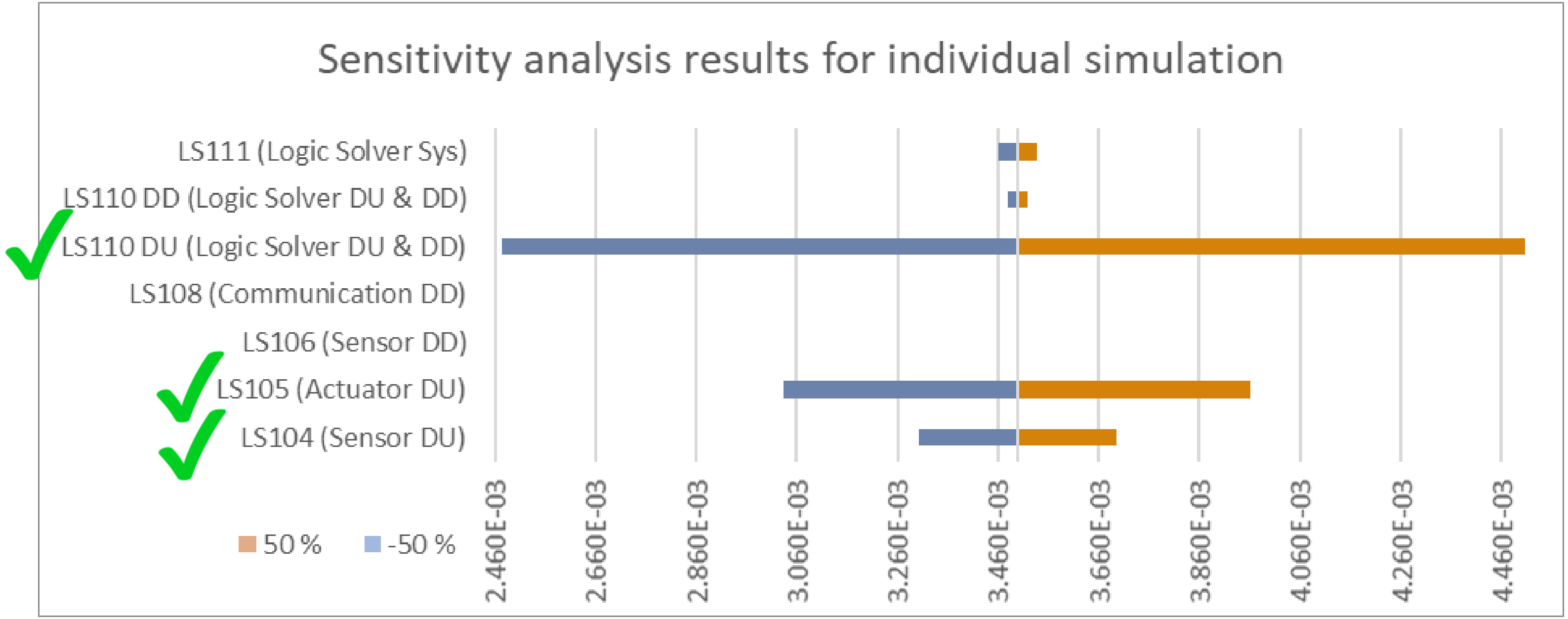
- Differences in the result are caused by several reasons:
  - Unseen parameters from Juntao's paper
  - Transition that are coupled between LS 1 and 2 in the Juntao's model (not modelled due to missing information)

Comparison with traditional quantitative modeling approach

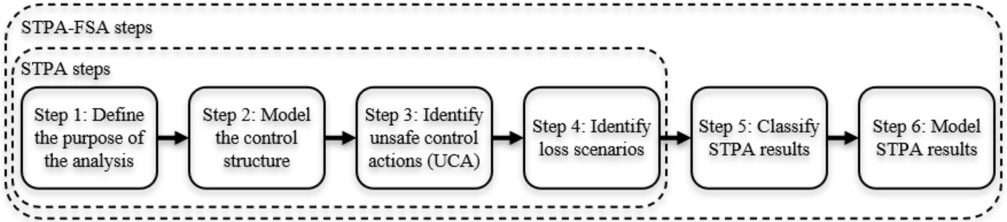
- STPA-FSA approach is essentially quantifying PFH and demand rate in the same model



Prioritization based on quantified value



Reduction of model uncertainty




```
//------------------------------------------------------------------------------------------------
domain ControlledProcessState {Normal, Context, Safe, Hazardous, Loss}
domain CCFState {Normal, CCF}
//------------------------------------------------------------------------------------------------
block ControlledProcess
    block Com## //Control element model for a group of component ##, following the modeling pattern for CCF
        ...
    end
    ...
    parameter Real pContext = 8760.0; // Frequency of context transition
    parameter Real pRestorefrHaz = 8.0; // Required restoration time from hazardous
    parameter Real pRestorefrSaf = 8.0; // Required restoration time from safe
    ControlledProcessState CP_State (init = Normal); // Controlled process state
    event evContextchg (delay = exponential(1./pContext));
    event evRestorefrHaz (delay = exponential(1./pRestorefrHaz));
    event evRestorefrSaf (delay = exponential(1./pRestorefrSaf));
    event evSCA## (delay = 0.001); //Delay is for observation purpose
    event evUCA## (delay = 0.001); //Delay is for observation purpose
    transition
        evContextchg:   CP_State == Normal -> CP_State := Context; // Change in the operational condition
        evRestorefrHaz: CP_State == Hazardous and (SF### and ...)
            -> CP_State := Normal;  // System restoration transition from hazardous
        evRestorefrSaf: CP_State == Safe and (SF### and ...)
            -> CP_State := Normal; // System restoration transition from safe
        evSCA##:        CP_State == Context and (SF### and ...)
            -> CP_State := Safe; // Safe control action transition
        evUCA##:       ?Com##.evDemand & CP_State == Context and (CF### or ...)
            -> CP_State := Hazardous; // Unsafe control action transition
        Boolean SF###, ... (reset = false);
        Boolean CF###, ... (reset = false);
        assertion
            SF### := (Com##.Ele##.H_State == Working and Com##.Ele##.C_State == Normal);
            ...
    observer Boolean UCA## = if CP_State == Hazardous then true else false;  // Observed UCA
    observer Boolean SCA## = if CP_State == Safe then true else false; // Observed SCA
    observer Boolean LS### = if (CP_State == Context and CF###) then true else false; // Observed Loss Scenario
    ...
    observer Integer EleUFailure = Com##.Ele##.Failure; // Observed individual failure
    ...
end
//------------------------------------------------------------------------------------------------
```

```
//------------------------------------------------------------------------------------------------
domain HardwareState {Working, Undetected, Detected, Repair}
domain ElementState {Normal, Demand}
domain Phase {Operation, Inspection}
//------------------------------------------------------------------------------------------------
class CompWRHF
    HardwareState H_State (init = Working); // Component hardware state
    ElementState C_State (init = Normal); // Synchronization with controlled process state due to demand
    Phase Team_Phase (init = Operation); // Maintenance team working phase
    parameter Real pLambdaU = 1e-6; // Undetected failure rate
    parameter Real pLambdaD = 1e-4; // Detected failure rate
    parameter Real pMu = 1.25e-1; // Repair rate
    parameter Real pRepairDelay = 24.0;
    parameter Real pInspectionPeriod = 4380.0;
    parameter Real pInspectionDuration = 5.0;
    Integer UndetectedFailure, DetectedFailure (init = 0);
    event evUndetectedFailure (delay = exponential(pLambdaU));
    event evDetectedFailure (delay = exponential(pLambdaD));
    event evCCFU (delay = 0, hidden = true);
    event evCCFD (delay = 0, hidden = true);
    event evDemand (delay = 0, hidden = true);
    event evPeriodicInsp (delay = pInspectionPeriod);
    event evCompleteInsp (delay = exponential(1/pInspectionDuration));
    event evRepairStart (delay = exponential(1/pRepairDelay));
    event evRepairEnd (delay = exponential(pMu));
    transition
        evUndetectedFailure: H_State == Working
            -> {H_State := Undetected; UndetectedFailure := UndetectedFailure + 1;}
        evDetectedFailure:   H_State == Working
            -> {H_State := Detected; DetectedFailure := DetectedFailure + 1;}
        evCCFU:              H_State == Working
            -> {H_State := Undetected; UndetectedFailure := UndetectedFailure + 1;}
        evCCFD:              H_State == Working
            -> {H_State := Detected; DetectedFailure := DetectedFailure + 1;}
        evDemand:            H_State == Undetected or H_State == Detected
            -> {if H_State == Undetected then H_State := Detected; C_State := Demand;}
        evPeriodicInsp:      Team_Phase == Operation -> Team_Phase := Inspection;
        evCompleteInsp:      Team_Phase == Inspection
            -> {Team_Phase := Operation ; if H_State == Undetected then H_State := Detected;}
        evRepairStart:       Team_Phase == Operation and (H_State == Detected) ->  H_State := Repair;
        evRepairEnd:         Team_Phase == Operation and H_State == Repair
            -> {H_State := Working; C_State := Normal;}
    Boolean input, output (reset = false);
    assertion
        output := if H_State == Working then true else false;
end
//------------------------------------------------------------------------------------------------
```

Input for risk assessment method using STPA (Kim, 2020)

Table 4. Evaluation criteria for loss scenarios.

Criteria	Category and description
Likelihood[a] (LH)	5. Event that is expected to occur frequently. 4. Event that happens now and then and will normally be experienced by the personnel. 3. Rare event, but will possibly be experienced by the personnel. 2. Very rare event that will not necessarily be experienced in any similar plant. 1. Extremely rare event.
Strength of knowledge on loss scenario (SOK)	5. Complex scenario with no or few experience. 4. Complex scenario with a small number of experiences. 3. Complex scenario with a large number of experiences. 2. Straightforward scenario with a small number of experiences. 1. Straightforward scenario with a large number of experiences.

[a]For details of classifications of likelihood, readers can refer to Rausand.[20]

$$RPN_{LossScenario} = RPN_{UCA} \times LH \times SOK_{LossScenario} = SV \times ATR \times SOK_{UCA} \times LH \times SOK_{LossScenario}$$

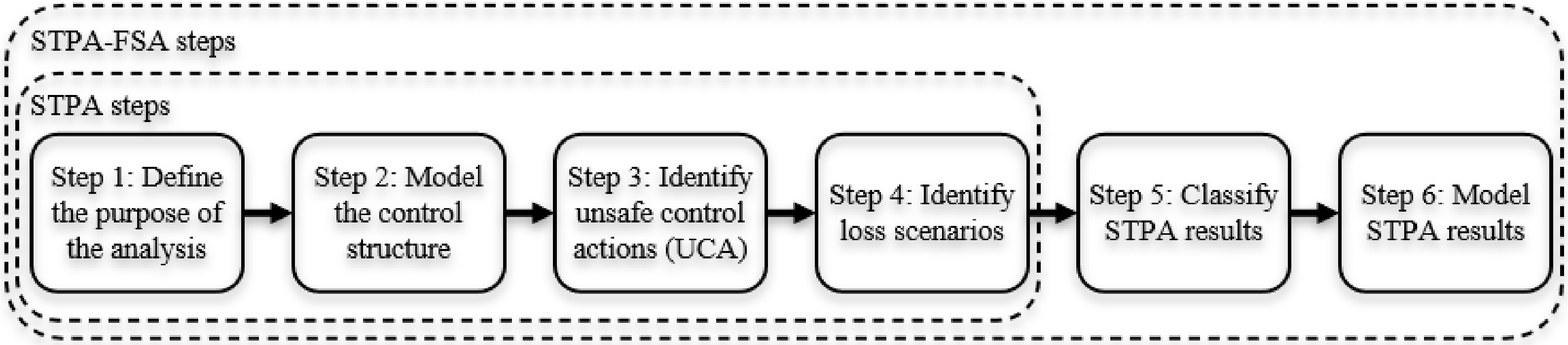
Approach limitation

- Data uncertainty
- Completeness uncertainty
- Aggregation of multiple scenarios into one model (for UCAs)

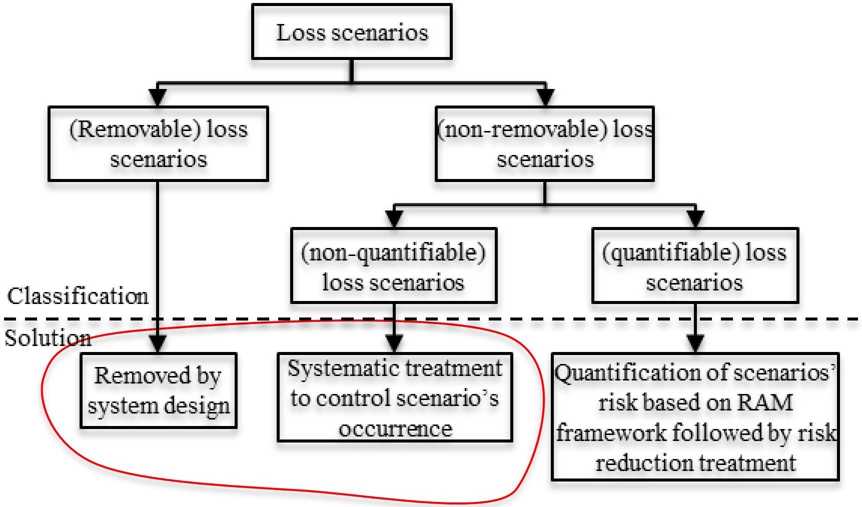
Data uncertainty

Parameter	Value	Probability distribution
SS Sensor failure rate	DD = 2.490e-8 /hour	Exponential
SS Sensor erratic reading rate	DU = 2.122e-7 /hour	Exponential
SS Actuator failure rate	DU = 3e-7 /hour	Exponential
Communication equipment failure rate	DD = 1e-6 /hour (assumption, need discussion)	Exponential
PCS/SS logic solver failure rate	DU = 3.810e-8 /hour; DD = 4.25e-7 /hour	Exponential Exponential
SS software systematic fault introduction rate	Sys= 5e-6 /hour (assumption, need discussion)	Exponential
Repair time	8 hour	Exponential
Repair delay	8 hour	Exponential
Inspection period	once every 6 months	Dirac
Inspection duration	24 hour	Exponential
Frequency of context change	once per year	Exponential
System restoration time	8 hour	Exponential
Simulation time	87,600 hour	n/a
Number of simulations	100,000	n/a

Completeness uncertainty



Aggregation of multiple scenarios into one model (for UCA)



Omission of some scenario's risk

NTNU

Reference

References

[1] N. Leveson, Engineering a safer world: systems thinking applied to safety, Engineering systems, The MIT Press, Cambridge, MA, 2011.

[2] T. Bjerga, T. Aven, E. Zio, Uncertainty treatment in risk analysis of complex systems: The cases of STAMP and FRAM, Reliability Engineering & System Safety 156 (2016) 203–209.

[3] N. Leveson, J. Thomas, STPA handbook, 2018.

[4] M. Rodríguez, I. Díaz, A systematic and integral hazards analysis technique applied to the process industry, Journal of Loss Prevention in the Process Industries 43 (2016) 721–729.

[5] S. Sultana, P. Okoh, S. Haugen, J. E. Vinnem, Hazard analysis: Application of stpa to ship-to-ship transfer of lng, Journal of Loss Prevention in the Process Industries 60 (2019) 241–252.

[6] S. M. Sulaman, A. Beer, M. Felderer, M. Höst, Comparison of the FMEA and STPA safety analysis methods—a case study 27 (1) (2019) 349–387.

[7] B. Rokseth, I. B. Utne, J. E. Vinnem, Deriving verification objectives and scenarios for maritime systems using the systems-theoretic process analysis, Reliability Engineering & System Safety 169 (2018) 18–31.

[8] ISO/PAS 21448, Road vehicles–safety of the intended functionality, Standard, International Organization for Standardization (2019).

[9] H. Kim, M. A. Lundteigen, A. Hafver, F. B. Pedersen, Utilization of risk priority number to systems-theoretic process analysis: A practical solution to manage a large number of unsafe control actions and loss scenarios, Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability.

[10] A. Hafver, S. Eldevik, I. Jakopanec, O. V. Drugan, F. Pedersen, R. Flage, T. Aven, Risk-based versus control-based safety philosophy in the context of complex systems, in: Safety and reliability - theory and applications : proceedings of the 27th European Safety and Reliability Conference (ESREL 2017), Portoroz, Slovenia, 18-22 June 2017, Boca Raton et al.: CRC Press, 2017, pp. 38–38.

[11] W. G. Temple, Y. Wu, B. Chen, Z. Kalbarczyk, Systems-theoretic likelihood and severity analysis for safety and security co-engineering, in: International Conference on Reliability, Safety and Security of Railway Systems, Springer, 2017, pp. 51–67.

[12] J. Zhang, H. Kim, Y. Liu, M. A. Lundteigen, Combining system-theoretic process analysis and availability assessment: A subsea case study, Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability 233 (4) (2019) 520–536.

[13] H. Jianbo, Z. Lei, X. Shukui, Safety analysis of wheel brake system based on STAMP/STPA and Monte Carlo simulation, Journal of Systems Engineering and Electronics 29 (6) (2018) 1327–1339.

[14] NUREG 1855, Guidance on the treatment of uncertainties associated with pras in riskinformed decision making, Standard, United States Nuclear Regulatory Commission (2009).

[15] H. Meng, L. Kloul, A. Rauzy, Modeling patterns for reliability assessment of safety instrumented systems, Reliability Engineering & System Safety 180 (2018) 111–123.

[16] ISO/TR 12489, Petroleum, petrochemical and natural gas industries – reliability modelling and calculation of safety systems, Standard, International Organization for Standardization (2013).

[17] IEC 61508, Functional safety of electrical/electronic/-programmable electronic safety-related systems – part 1-7, Standard, International Electrotechnical Commission (2010).

[18] M. Batteux, T. Prosvirnova, A. Rauzy, Modeling patterns for the assessment of maintenance policies with altarica 3.0, in: International Symposium on Model-Based Safety and Assessment, Springer, 2019, pp. 32–46.

[19] A. Rauzy, Notes on computational uncertainties in probabilistic risk/safety assessment, Entropy 20 (3) (2018) 162.

[20] A. Hoyland, M. Rausand, System reliability theory: models and statistical methods, Vol. 420, John Wiley & Sons, 2009.

[21] P. Hokstad, K. Corneliussen, Loss of safety assessment and the IEC 61508 standard, Reliability Engineering & System Safety 83 (1) (2004) 111–120.

[22] A. Toola, The safety of process automation, Automatica 29 (2) (1993) 541–548.

[23] L. Xie, M. Lundteigen, Y. Liu, Safety barriers against common cause failure and cascading failure: literature reviews and modeling strategies, in: 2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), IEEE, 2018, pp. 122–127.

[24] M. Rausand, Risk assessment: theory, methods, and applications, Statistics in practice, Wiley, Hoboken, N.J., 2011.

[25] IEC 61025, Fault tree analysis (FTA), Standard, International Electrotechnical Commission (2006).

[26] IEC 61078, Reliability block diagrams, Standard, International Electrotechnical Commission (2016).

[27] J. I. Aizpurua, E. Muxika, Model-based design of dependable systems: limitations and evolution of analysis and verification approaches, International Journal on Advances in Security 6 (1).

[28] IEC 61165, Application of Markov techniques, Standard, International Electrotechnical Commission (2006).

[29] IEC 62551, Analysis techniques for dependability – Petri net techniques, Standard, International Electrotechnical Commission (2012).

[30] T. Prosvirnova, Altarica 3.0: a model-based approach for safety analyses, Ph.D. thesis, École Polytechnique (2014).

[31] M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, G. Franceschinis, Modelling with generalized stochastic Petri nets, Vol. 292, Wiley New York, 1995.

[32] S. F. Railsback, V. Grimm, Agent-based and individual-based modeling: a practical introduction, Princeton university press, 2019.

[33] O. Nývlt, S. Haugen, L. Ferkl, Complex accident scenarios modelled and analysed by stochastic petri nets, Reliability Engineering & System Safety 142 (2015) 539–555.

[34] J.-P. Signoret, Y. Dutuit, P.-J. Cacheux, C. Folleau, S. Collas, P. Thomas, Make your petri nets understandable: Reliability block diagrams driven petri nets, Reliability Engineering & System Safety 113 (2013) 61–75.

[35] Q. Wu, D. Gouyon, P. Hubert, E. Levrat, Towards model-based systems engineering (mbse) patterns to efficiently reuse know-how, Insight 20 (4) (2017) 31–33.

[36] B. Hamid, J. Perez, Supporting pattern-based dependability engineering via model-driven development: Approach, tool-support and empirical validation, Journal of Systems and Software 122 (2016) 239–273.

[37] D. Cook, W. D. Schindel, Utilizing mbse patterns to accelerate system verification, Insight 20 (1) (2017) 32–41.

[38] N. A. Zikrullah, M. J. P. van der Meulen, M. A. Lundteigen, A comparison of hazardous scenarios in architectures with different integration types, in: Proceedings of the 30th European Safety and Reliability Conference and the 15th Probabilistic Safety Assessment and Management Conference (ESREL 2020 PSAM15), Research Publishing Services, 2020, pp. 4001–4008.

[39] N. A. Zikrullah, H. Kim, M. J. van der Meulen, G. Skofteland, M. A. Lundteigen, A comparison of hazard analysis methods capability for safety requirements generation, Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability (2021) 1748006X211003463.

[40] N. A. Zikrullah, M. J. P. van der Meulen, M. A. Lundteigen, Building blocks for safety arguments with stpa, in: xxx, xxx, n.d., pp. xxx–xxx.

[41] J. P. Thomas IV, Extending and automating a systems-theoretic hazard analysis for requirements generation and analysis, Ph.D. thesis, Massachusetts Institute of Technology (2013).

[42] OREDA, Handbook 2015, 6th edition – Volume I and II, Princeton university press, 2015.

[43] DNVGL-RP-A203 technology qualification, Recommended practice, DNV GL (2009).

[44] S. Yamada, Software reliability modeling: fundamentals and applications, Vol. 5, Springer, 2014.

[45] B. Kirwan, A guide to practical human reliability assessment, CRC press, 1994.

[46] N. A. Zikrullah, M. J. P. van der Meulen, G. Skofteland, M. A. Lundteigen, Systems-theoretic process analysis results for system with different integration types, dataset V1, doi: 10.17632/prwtzmt3kg.1 (2021).

[47] DNV GL, Safety 4.0, retrieved 2021–03–10. URL : https://www.dnvgl.com/research/oil-gas/safety40/index.html (2018).

[48] Recommended practice for analysis, design, installation, and testing of safety systems for subsea applications, Standard, American Petroleum Institute (2015).

[49] S. Håbrekke, S. Hauge, T. Onshus, Reliability data for safety instrumented systems–PDS data handbook, 2021 edition, Vol. 13502, SINTEF.


Thank you