# SafeTrace: A Safety-Driven Requirement Traceability Framework on Device Interaction Hazards for MD PnP

***Andrew Y.-Z. Ou***
Department of Computer Science,
University of Illinois Urbana-Champaign

Rahmaniheris, M., Jiang, Y., Sha, L., Fu, Z. and Ren, S.

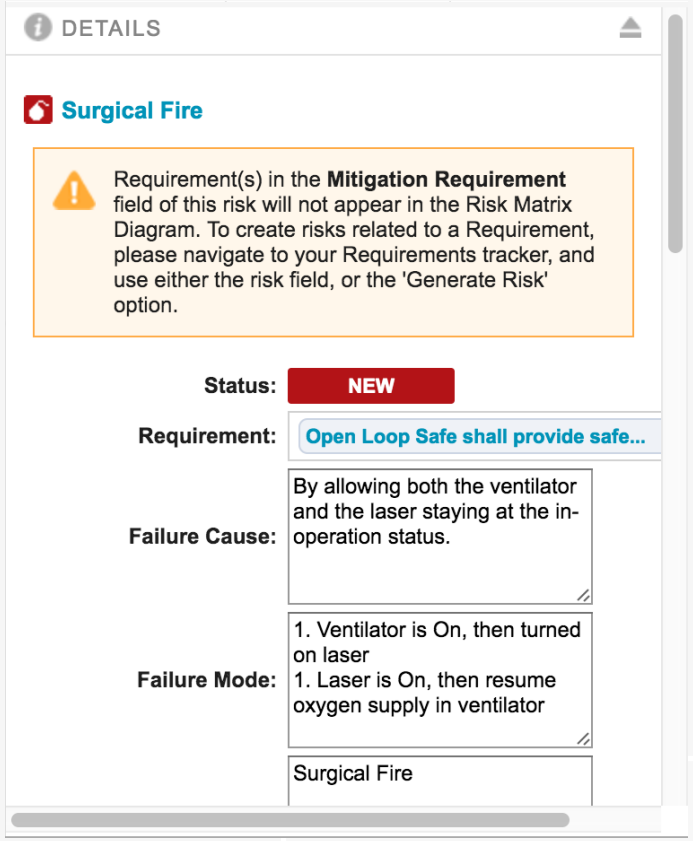
ACM SAC-2018 Pau, France, April 9-13th

# Motivation

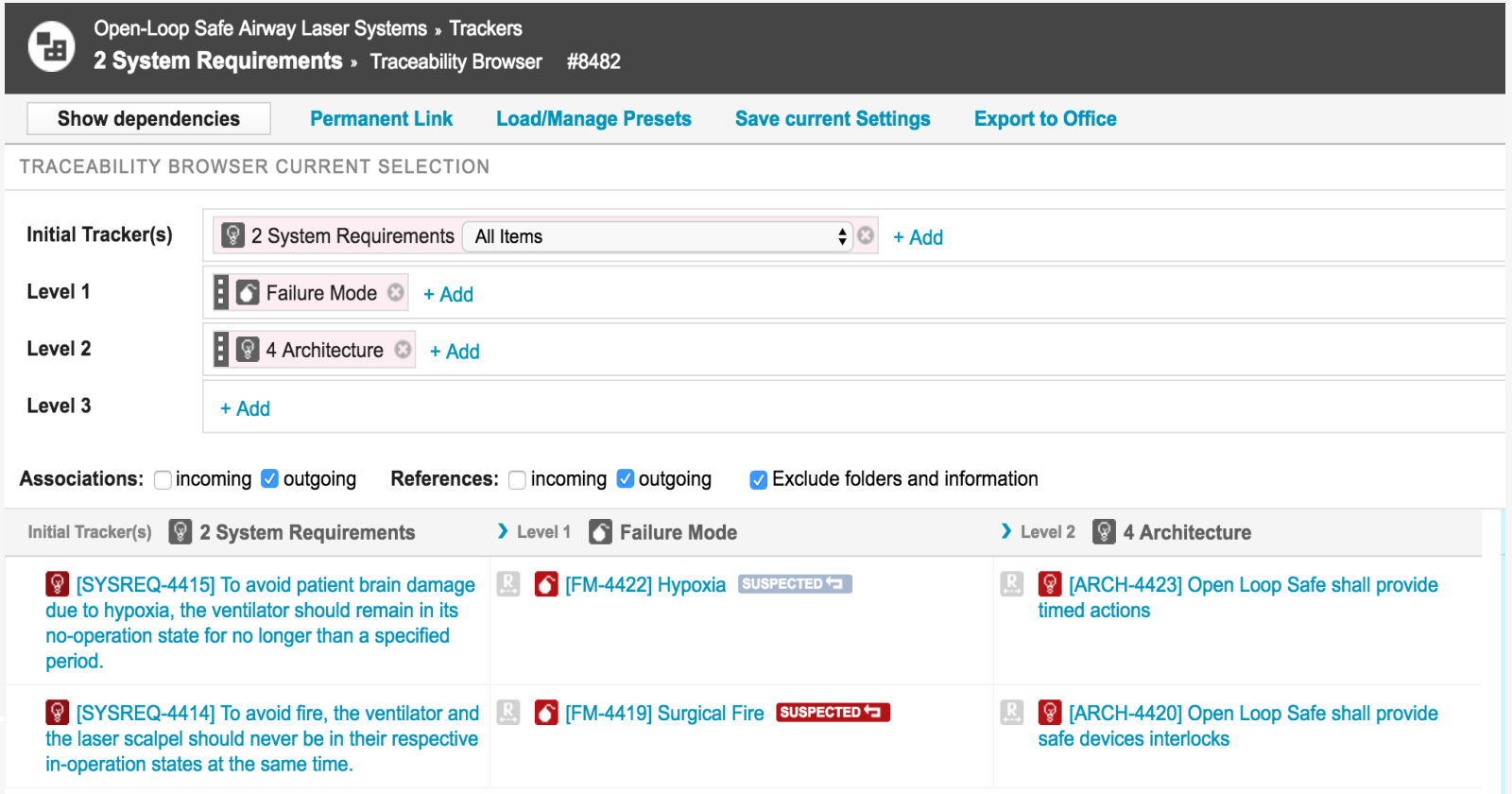
- Safety analysis and Traceability is **mandated** by medical devices standard or such as IEC 62304 and FDA
- However, Safety analysis is **partially/not traced** in traceability
  - Ex: IBM Rational DOORS, Yakindu Traceability, and Intland codeBeamer
- Even if some tools support safety analysis such as FMEA, however, the trace links are at relative higher level and *lack of a more fine grained control of trace links*.
- An **outdated** safety analysis may not reflect the latest safety status of a system

# Intland - codeBeamer

- Support only FMEA (Failure Mode Effectiveness Analysis, a table based safety analysis)
- To set up traceability, a downstream artifact should be manually added from the immediate upstream artifacts.



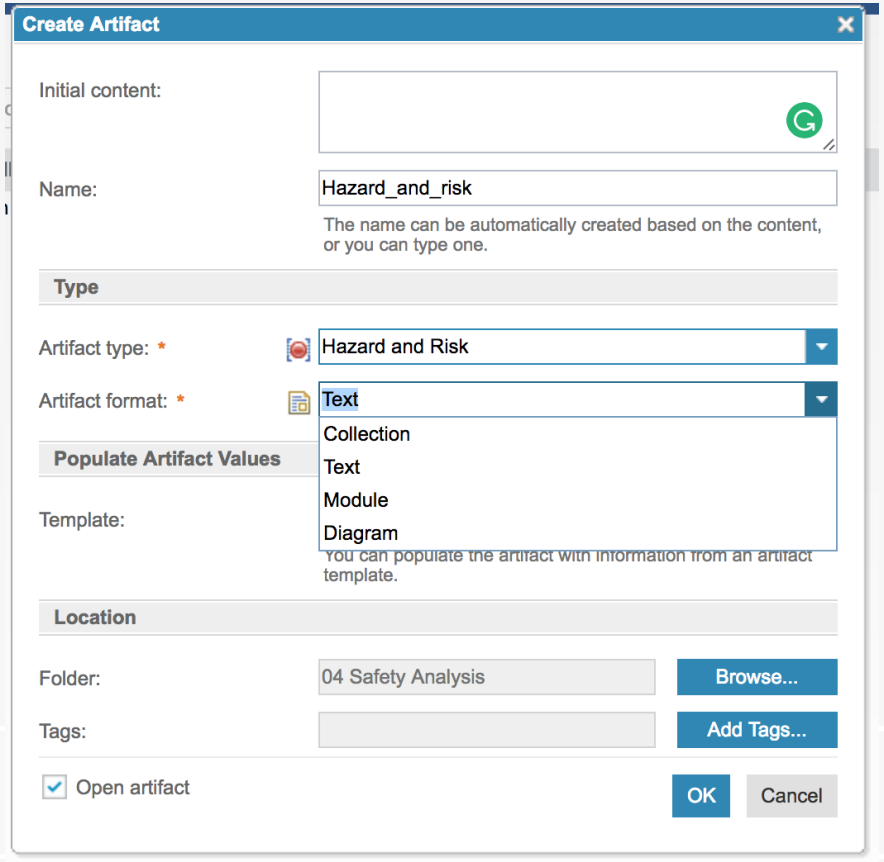

FMEA editor

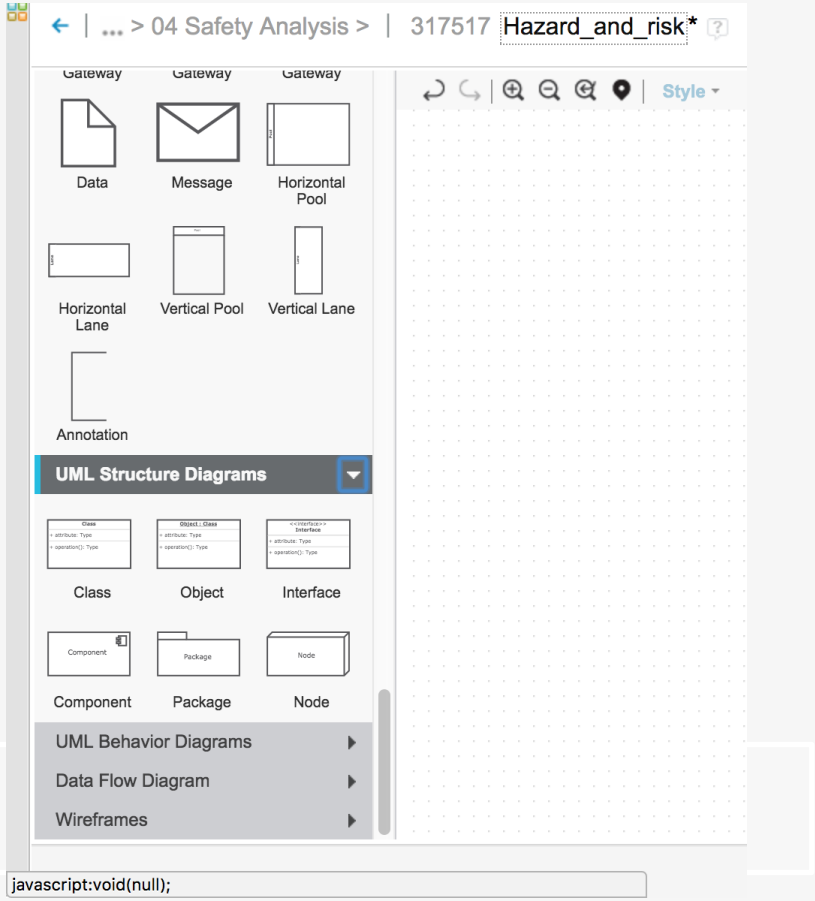



Traceability Browser, starting from a “tracker” and tracing to different levels

# IBM Doors

- No support for specific safety analysis methods (Hazard and risks in their terms), but only generic text, diagrams (ex: UML).



# Challenges

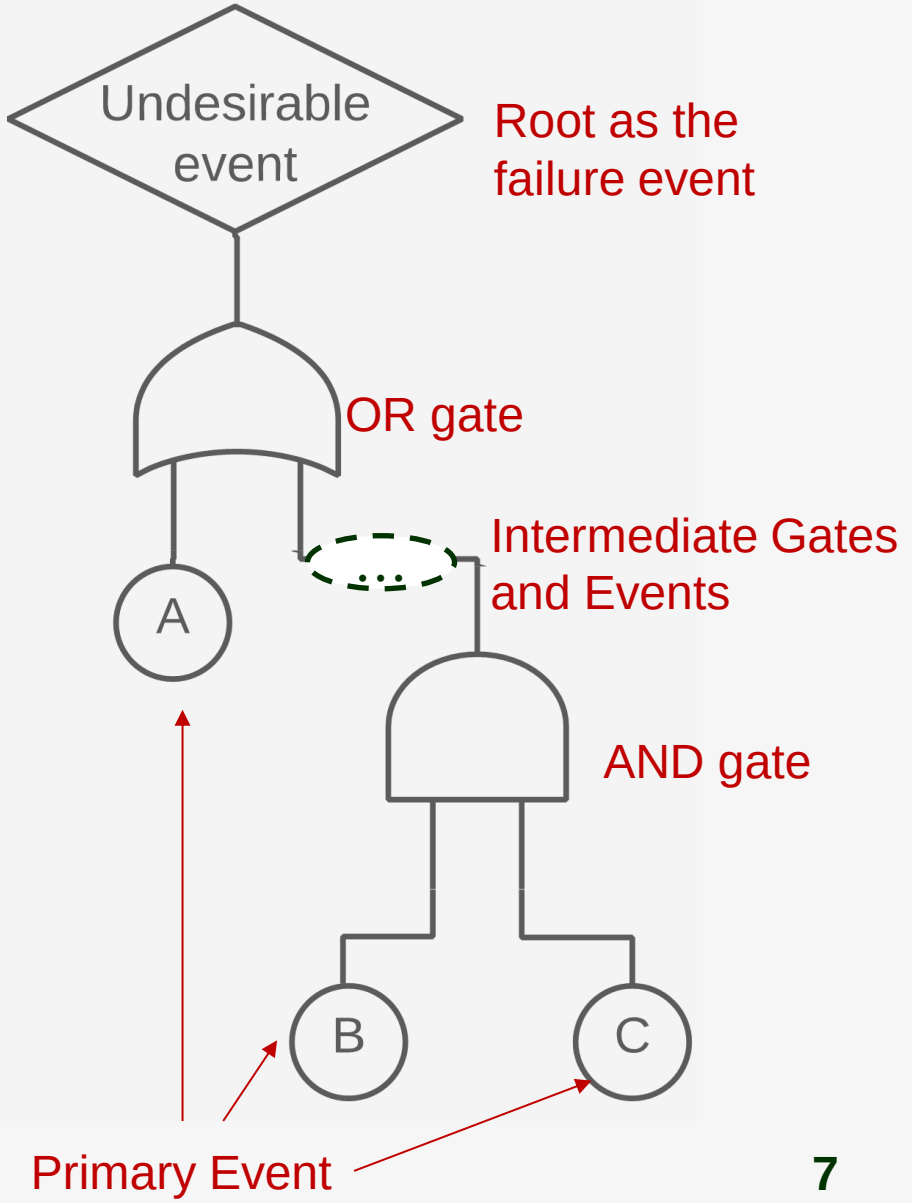
- How can we represent **device interactions** in safety analysis?
- What should be traced in safety analysis?
  - How can we leverage the analysis?
- How to integrate the trace links among safety requirements, system design, and safety analysis?
- How to perform change impact analysis?

# SafeTrace

- A safety-driven traceability framework integrating safety analysis
- Use fault-tree as safety analysis method
- Provide change impact analysis of requirements, and design changes on fault trees.

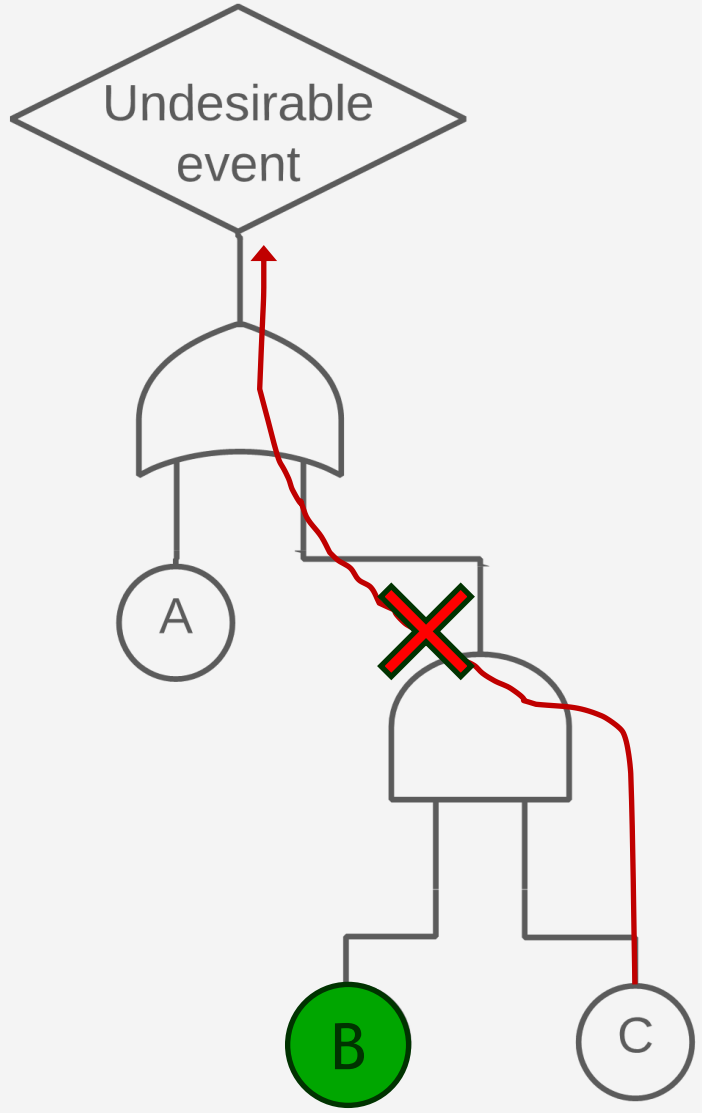
# Fault-Tree Analysis

- A widely used safety analysis method
- Embedded events and logics in a Tree Structure
- Provide quantitative evaluation such as reliability or Mean Time To Failure (MTTF)
- Provide qualitative evaluation for examining the system event combinations
- Many other possible semantics such as events happen *on certain Conditions*



# Fault-Tree Analysis

- Minimum cut set (mcs)
  - a set of primary events whose occurrence (at the same time) ensures that the TOP event occurs.
  - Preserve the logical relations
  - Ex: mcs = {{A}, {B,C}}
- Safe Guard Event always produces the False value
  - Ex: if B is a Safe Guard event, the path from C to root is broken



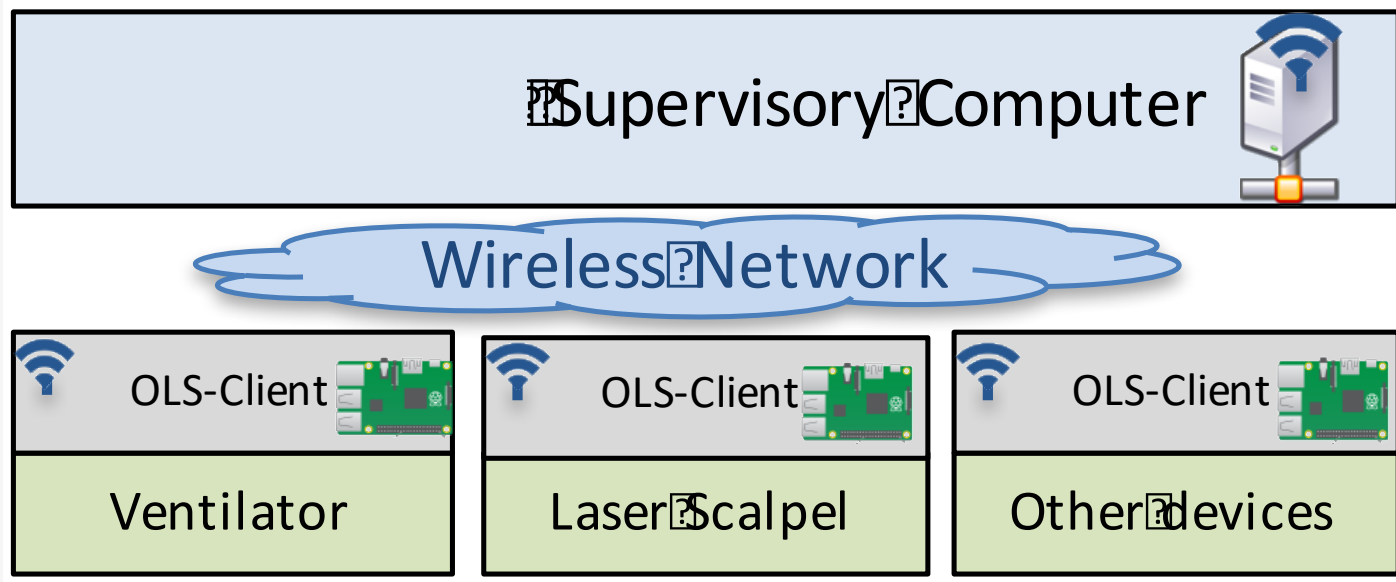

Always False

# Medical Scenario

- Tracheotomy Laser Surgery
- A physician uses a *laser scalpel* to unblock the patient's trachea when *ventilator* pauses supplying oxygen
- Potential Hazards:
  - **Surgical fire:** laser operating when oxygen level is high
  - **Hypoxia:** blocking of oxygen flow exceeds a certain duration

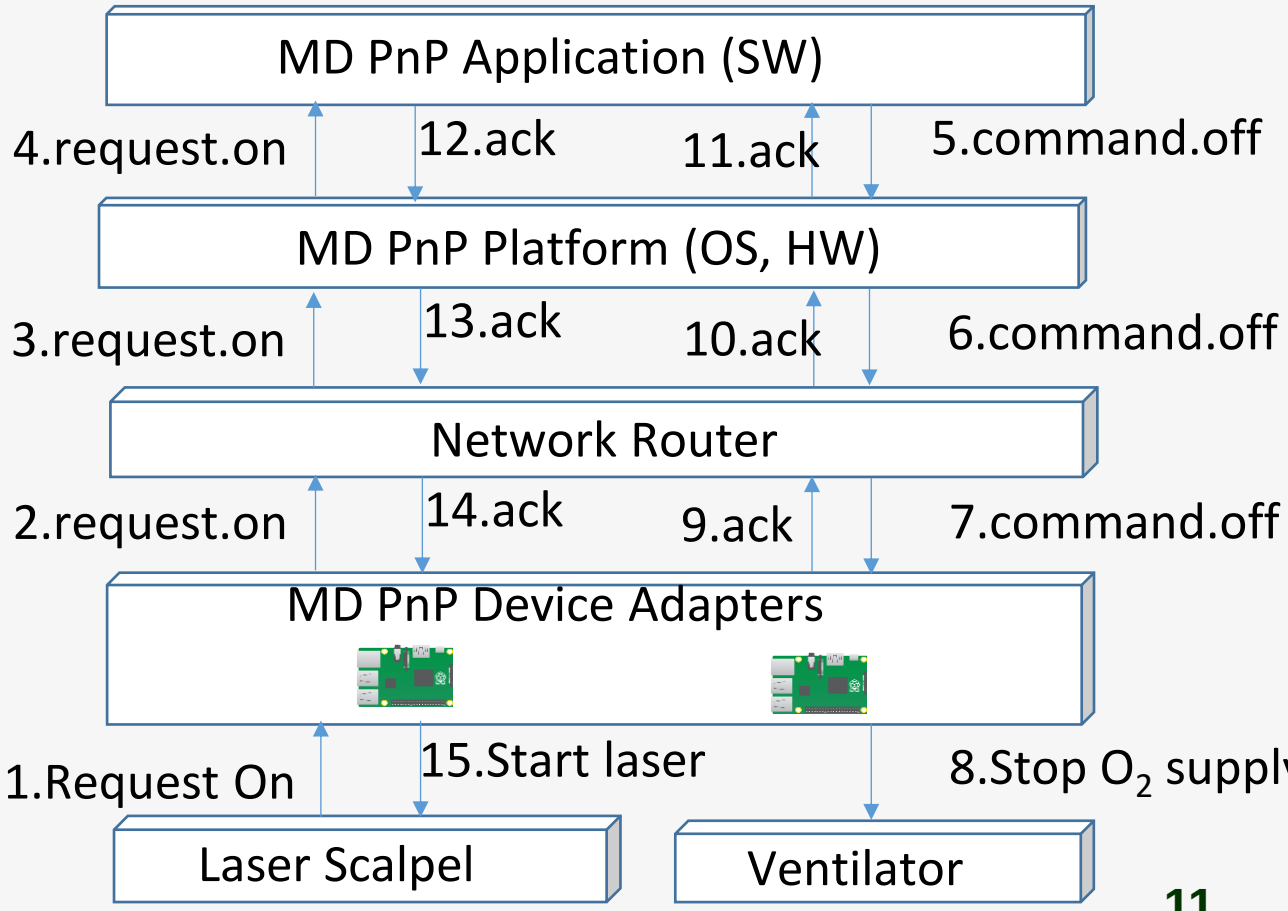
# MD PnP System Design for Tracheotomy

- Based on Medical Device Plug-and-Play (MD PnP) to provide medical devices interoperability
- **Supervisory computer** for devices coordination's
- A **certified safe adapter** on each device
  - Laser Scalpel
  - Ventilator
- Assume networked communication may fail anytime



# MD PnP for Tracheotomy – Command Flow

- **Laser** sends requests to **Supervisory computer** for devices coordination
- **Supervisory** prepares **Ventilator**
- **Ventilator** acknowledges
- **Supervisor** acknowledges **Laser**

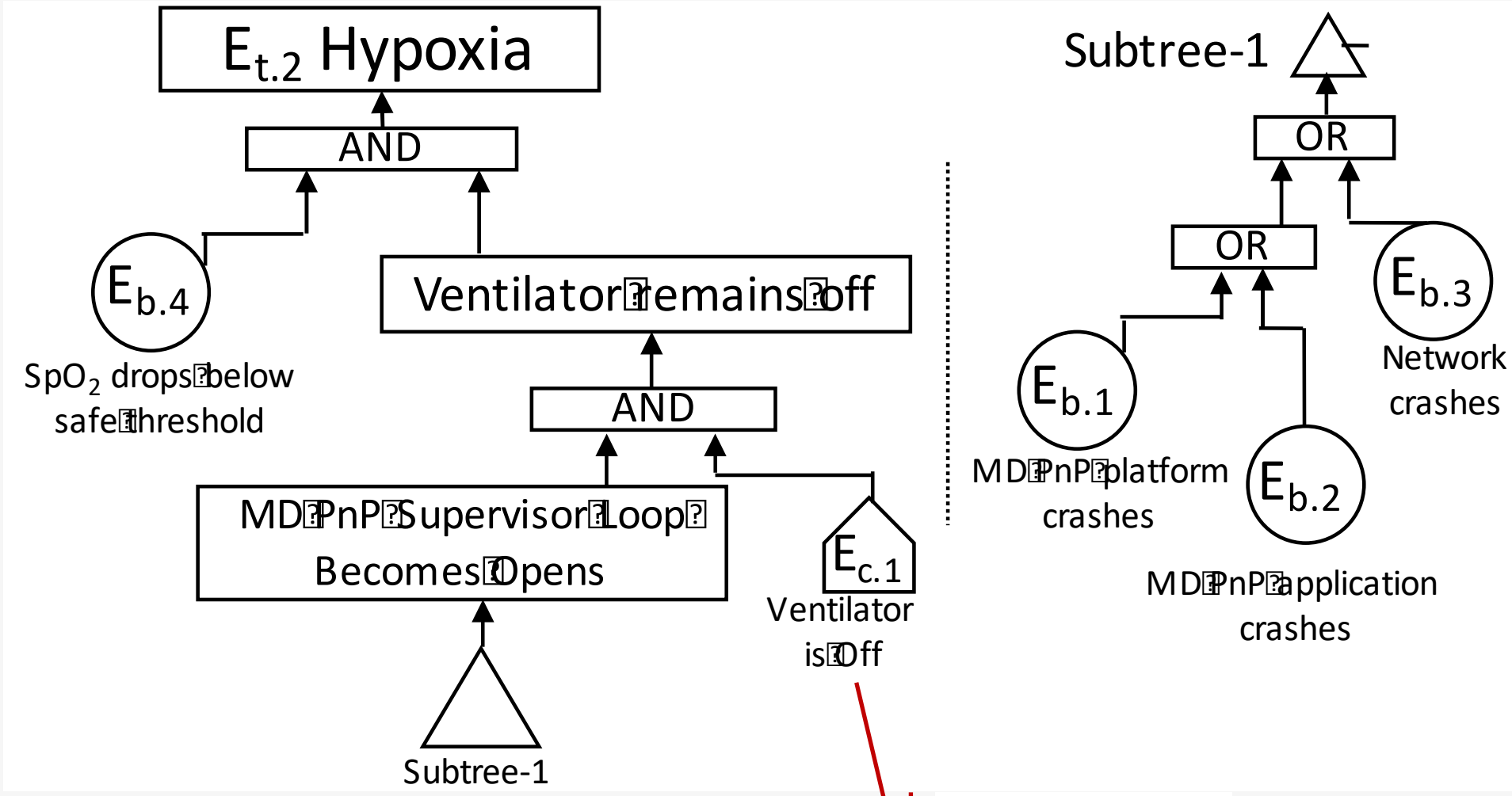


# MD PnP Tracheotomy System Safety Requirements

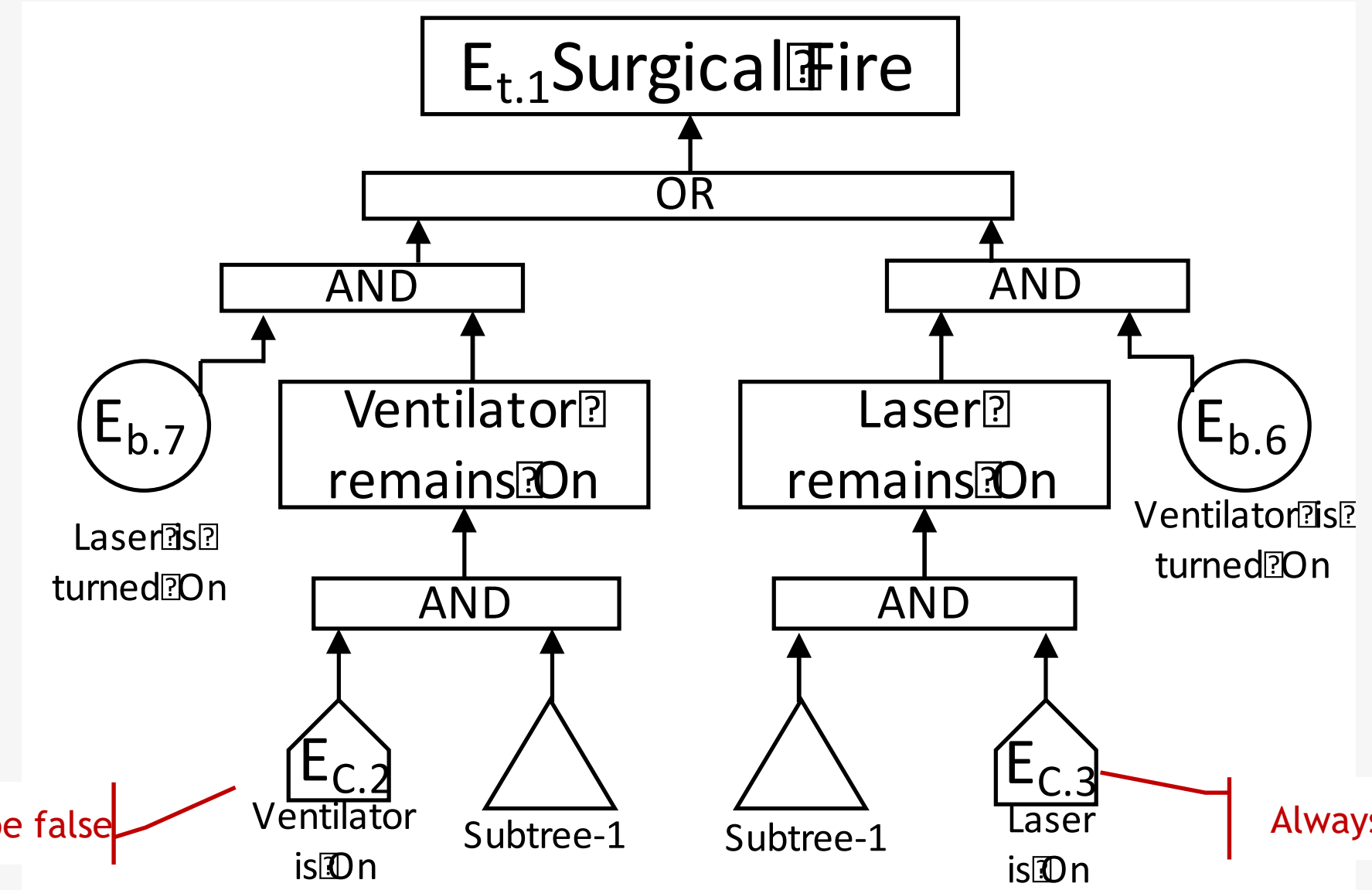
- Safety Requirement 1 (SafeReq-1): To avoid fire, the ventilator and the laser scalpel should never be in their respective in-operation states at the same time
  - => *requires device interactions*

- Safety Requirement 2 (SafeReq-2): To avoid patient brain damage due to hypoxia, the ventilator should remain in its no-operation state for no longer than a specified period.

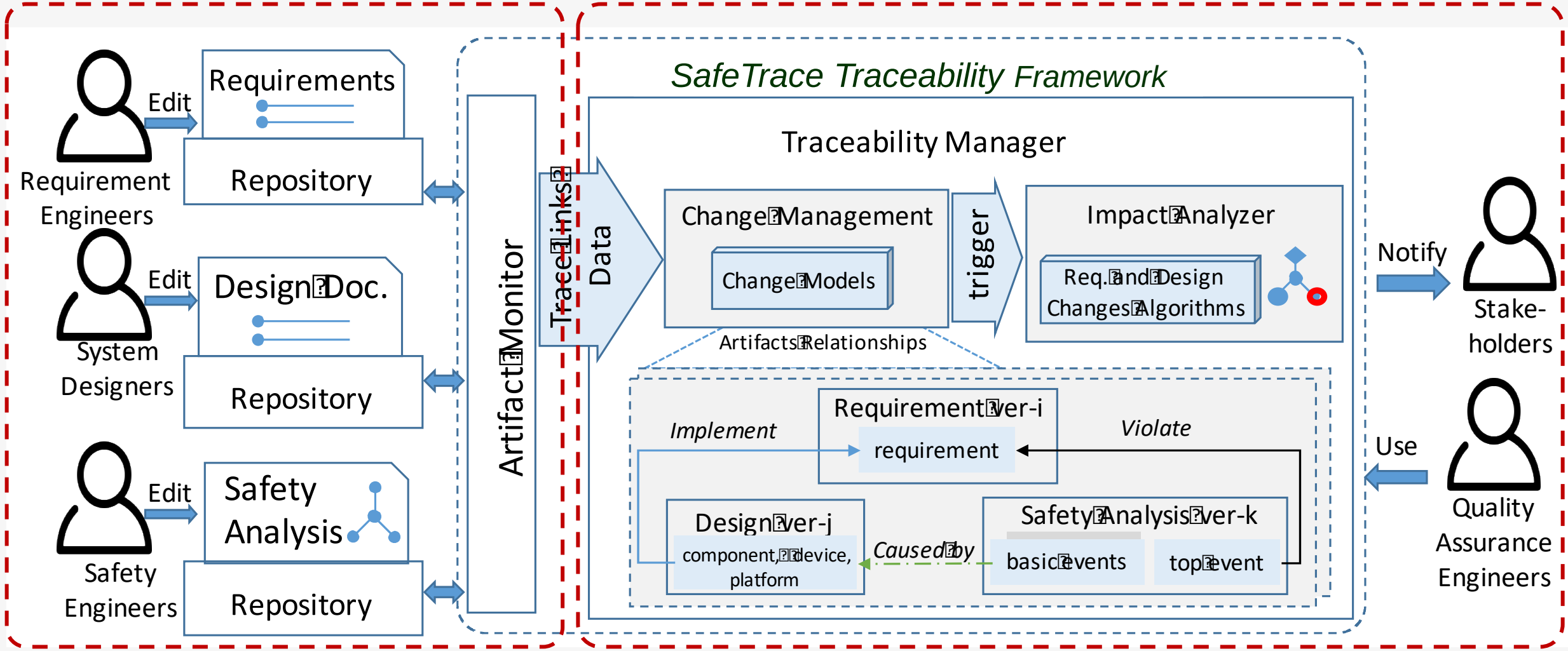
# Fault Tree of Hypoxia



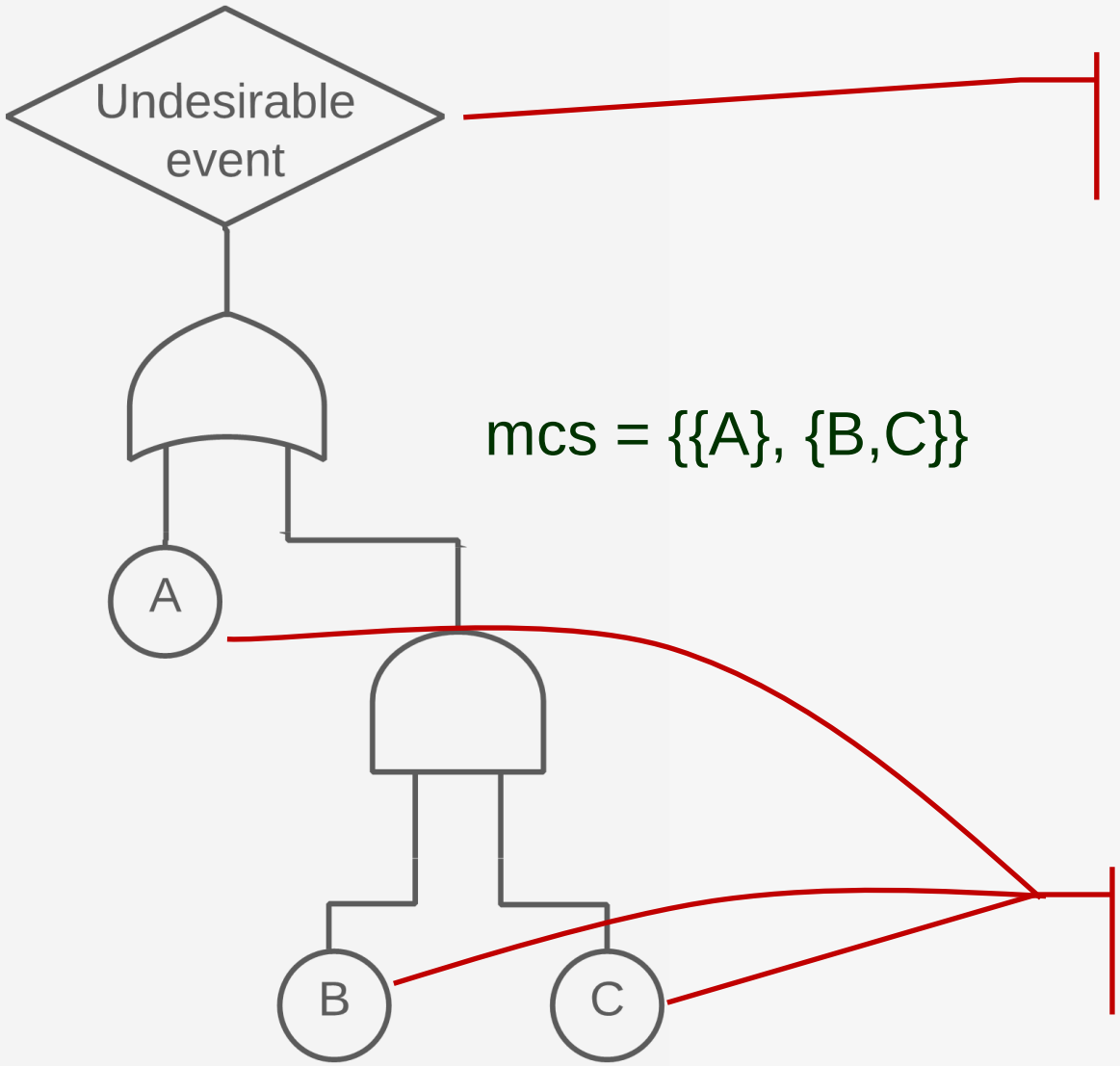
# Fault Tree of Surgical Fire



# SafeTrace Architecture

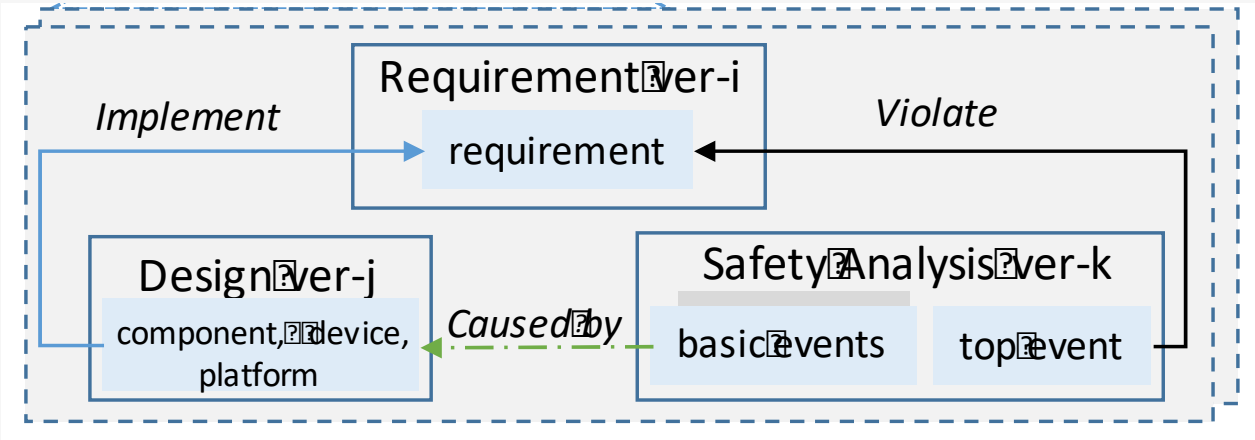


# Trace Links




mcs = {{A}, {B,C}}

The root event is traced to a safety requirement




A primary event is traced to a design component

# Requirement Change Impact Analysis

- Changes made to a *requirement* artifact includes the actions Creating, Deleting, or Updating
- Creating a req., see if the current design or FTA supports the new req.
- Deleting a req., see if the root of FTA becomes or design becomes isolated
- Updating a req., see if the current design or FTA supports the modified req.

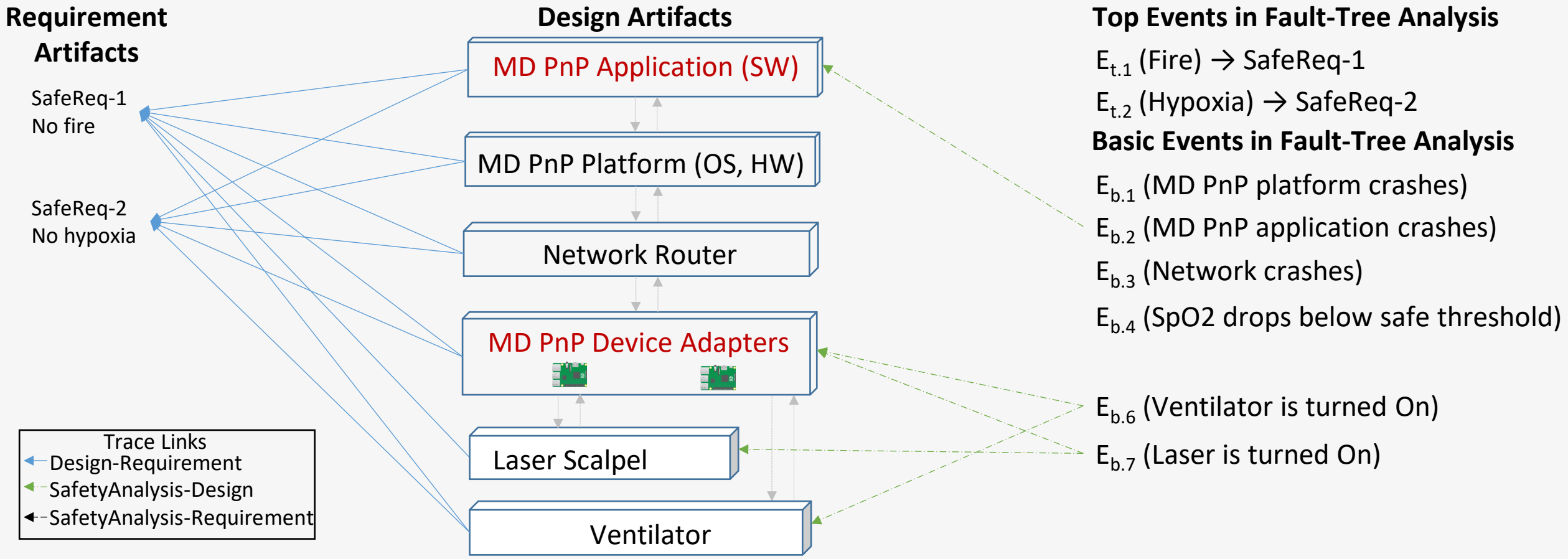
# Design Change Impact Analysis

- Changes made to a *design* artifact includes the actions Creating, Deleting, or Updating
- **Key idea**: Whether an Update in design will propagate to the failure at the root of a fault tree
- For each design artifact change *a*, find the associated events *e*, MCSs *mcs*, and requirements *req* and fault-tree *ft*
  - For each e associated with *a*, if *e* is the only event in *mcs,*
    - report *req* and *ft* could be impacted => Ex: mcs = {{**e**}, {B,C}}
  - Else if no safe guard event in mcs,
    - report *req* and *ft* could be impacted => Ex: mcs = {{A}, {B,**e**}}
  - *Else // e is in a cut set has a safe guard event*
    - report *req* and *ft* may NOT be impacted => Ex: mcs = {{A}, {B,**e**}}, B is a safe guard event

# Case Study - New Requirement

- New Requirement:
  - Safety Requirement 3 (SafeReq-3): The system shall bring the patient connected to the system to a safe state (i.e., supply the patient with oxygen) without causing either fire or hypoxia *if communications between the supervisor computer and medical devices fail.*
- Design changes:
  - Adding open-loop software into MD PnP application
  - Adding open-loop software into device Adapter
- Need to update the traceability graph and fault-tree analysis

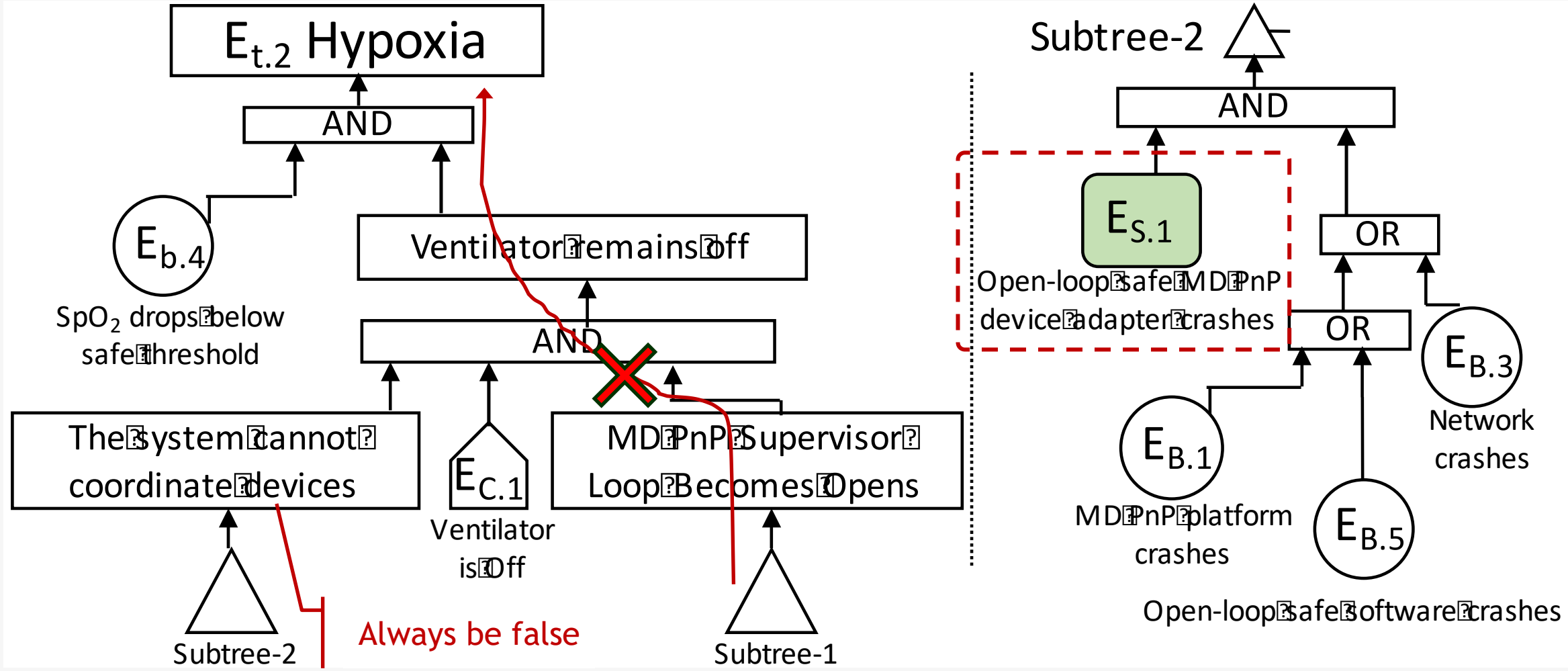
# Case Study - Traceability Graph without SafeReq-3



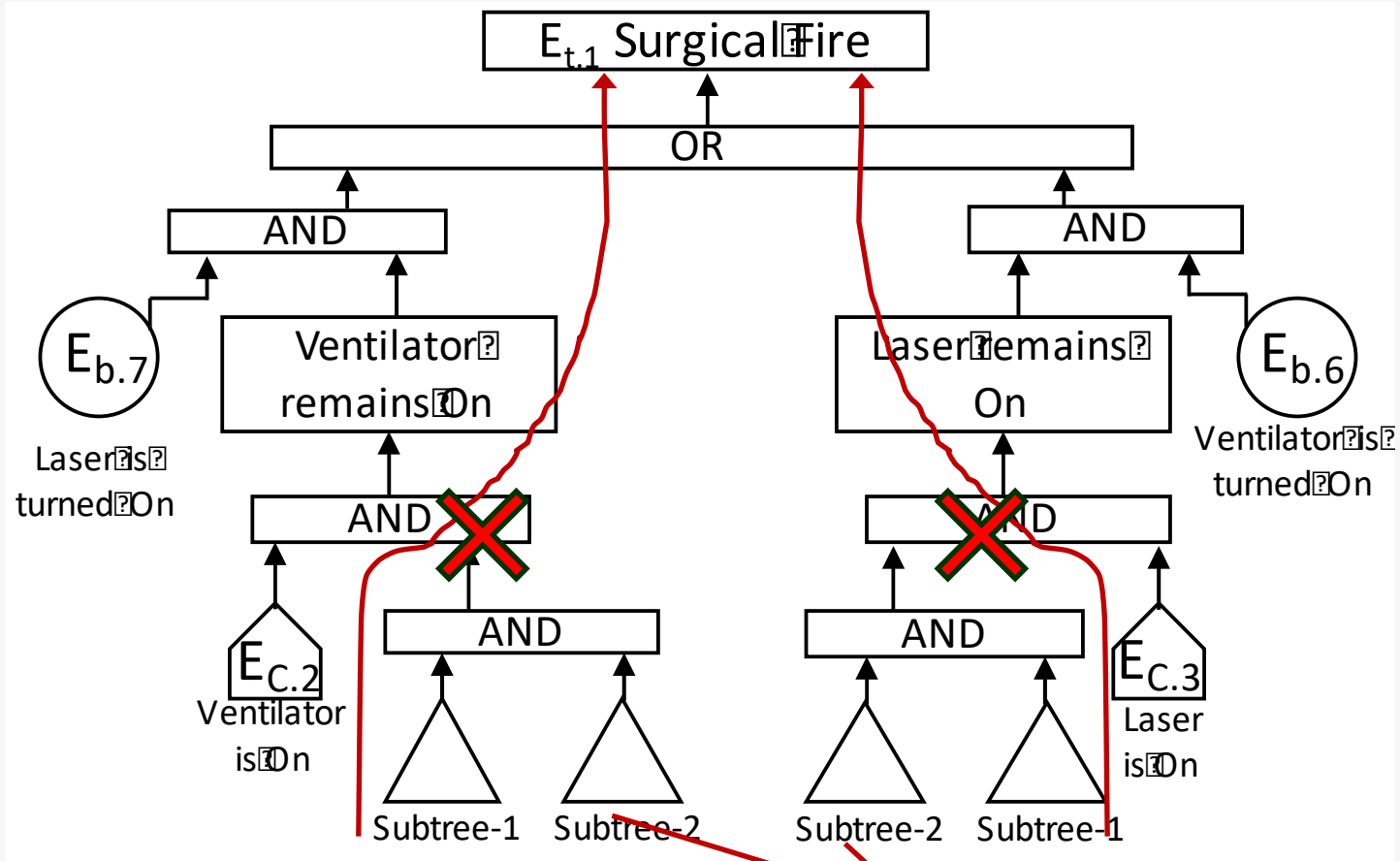

Note 1: Vertical arrows in design represent information flow only. They are not part of trace links.

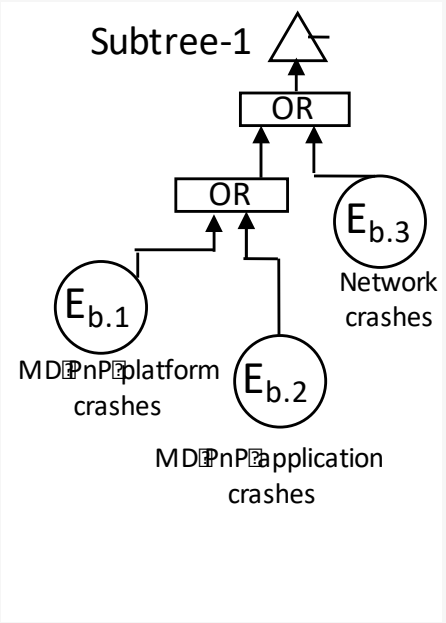
Note 2: No trace links setup for uncontrollable basic events $E_{b.1}$, $E_{b.3}$, and $E_{b.4}$

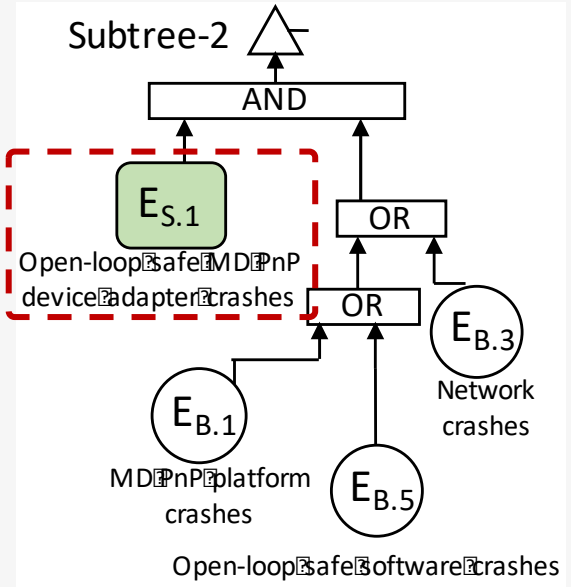
# Case Study- Phase 3 - Hypoxia



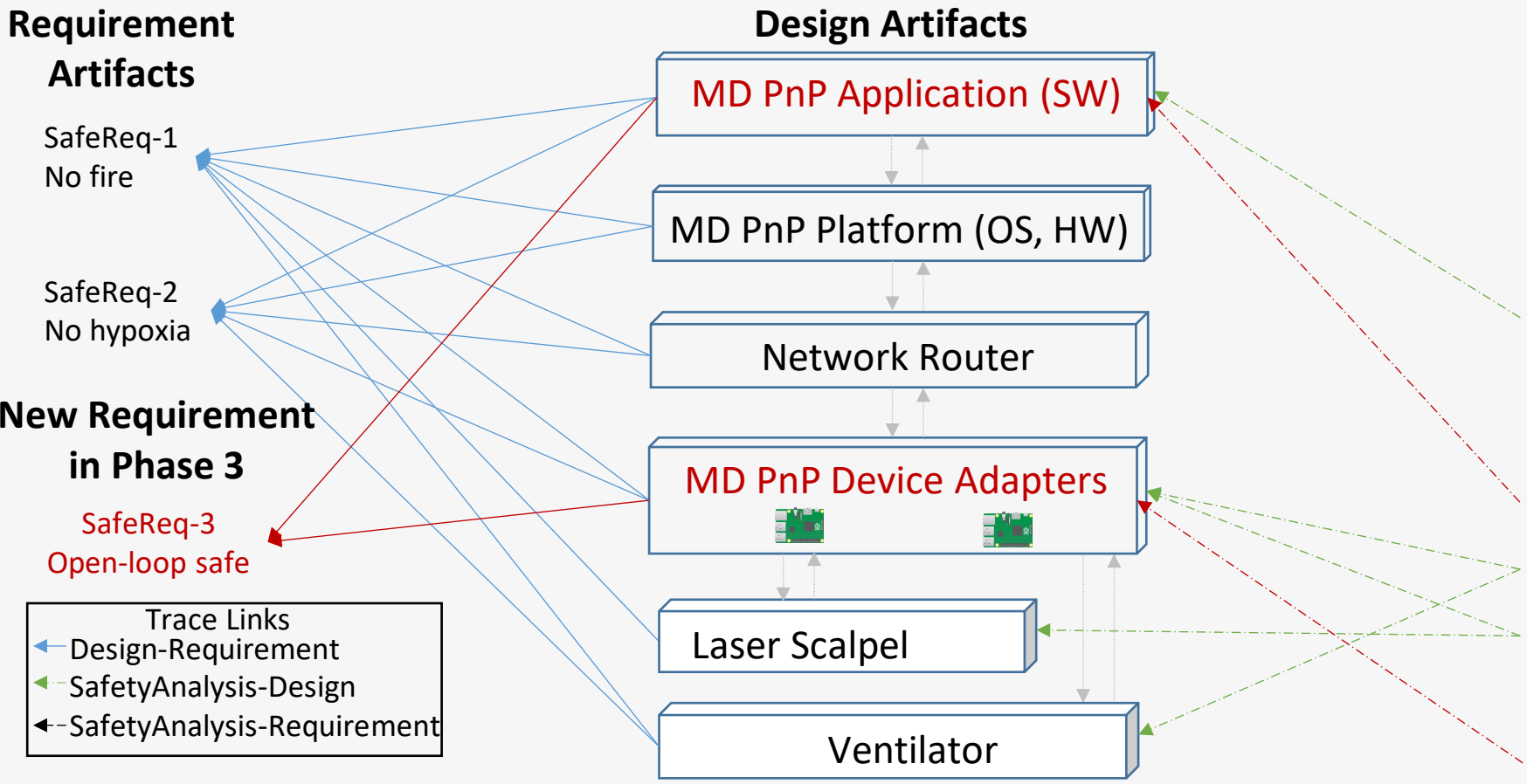
# Case Study - Phase 3 - Fire FT



# Case Study - Updated Traceability




Note: Vertical arrows in design represent information flow only. They are not part of trace links.

**Top Events in Fault-Tree Analysis**

$E_{t.1}$ (Fire) → SafeReq-1

$E_{t.2}$ (Hypoxia) → SafeReq-2

**Basic Events in Fault-Tree Analysis**

$E_{b.1}$ (MD PnP platform crashes)

$E_{b.2}$ (MD PnP application crashes)

$E_{b.3}$ (Network crashes)

$E_{b.4}$ (SpO2 drops below safe threshold)

$E_{b.5}$ (Open-loop safe software crashes)

$E_{b.6}$ (Ventilator is turned On)

$E_{b.7}$ (Laser is turned On)

**Safeguard Event in FTA in Phase 3**

$E_{s.1}$ (Open-loop safe MD PnP device adapter crashes)

# Discussion

- Manual setting up trace links could be tedious and error prone
  - Need computer-added automation in tool implementations
- The impact analysis based on MCS theory does not provide whether it is positive or negative impact
- Need to integrate SafeTrace with other artifacts such as source code, testing, statechart

# Conclusion

- SafeTrace manages traceability in life-critical systems including trace links
  - (1) between **design artifacts** and **basic events** in fault trees
  - (2) between **safety requirements** and the **top event** (i.e., failure proposition) of each tree
- Provides impact-analysis algorithms to identify the impacts on safety analysis that are caused by requirement- and design changes.
- Leverages the minimum cut sets of fault-tree analysis

# THANK YOU AND Q&A

# BACKUP SLIDES

# TO BE DELETED

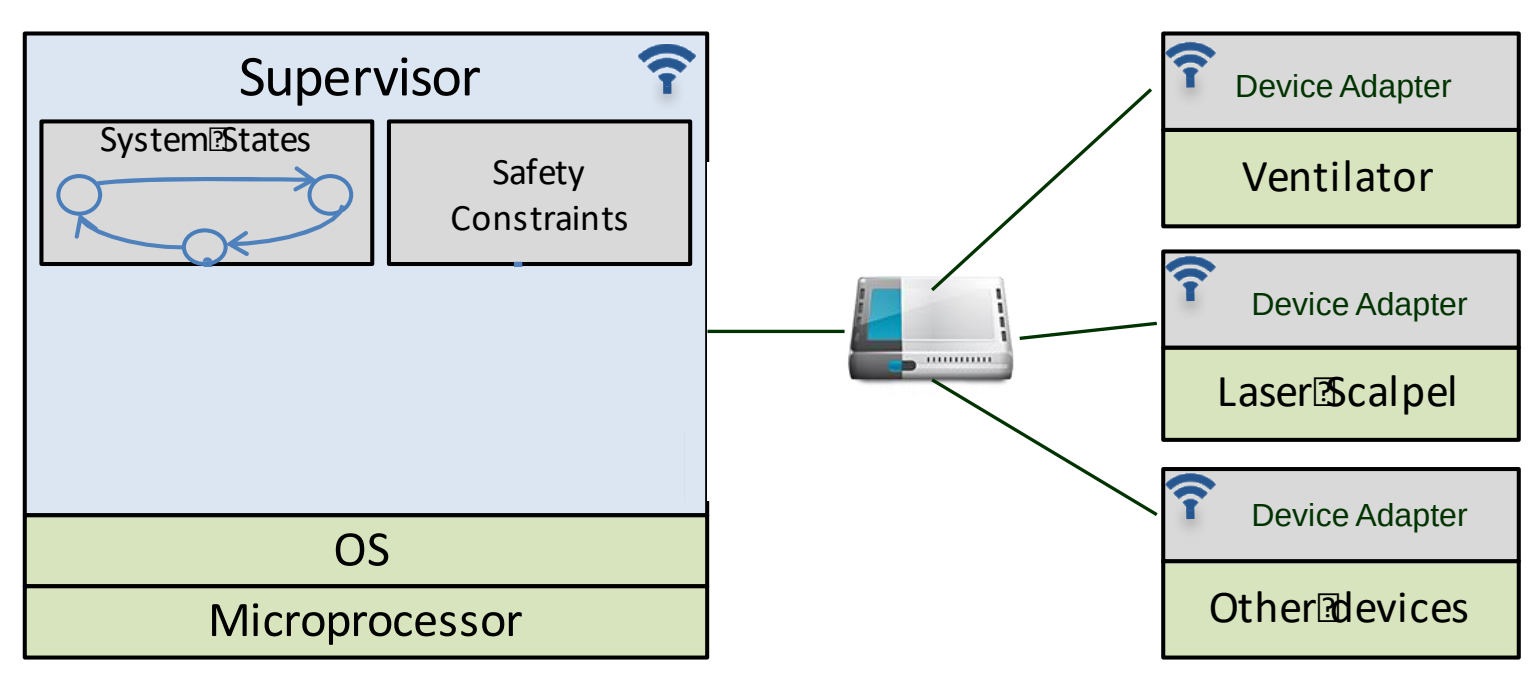
# Motivation

- Safety issues due to devices interactions
- Safety hazards in Airway Laser Tracheotomy
  - **Surgical fire**: laser scalpel is emitting while ventilator is supplying oxygen ($SpO_2$ is high)
  - **Brain hypoxia**: the oxygen supply in the ventilator is not resumed in time
- Medical Device Plug-and-Play Program
  - Provide medical device interoperability
  - Reduce the human errors

Picture Source: Airway Fires during Surgery, PA PSRS Patient Safe Advise 2007 Mar;4(1):1,4-6.

# Medical Device Plug-and-Play

- System architecture
  - Supervisor
  - Client-Adapter
  - Medical Devices
  - Wired Network
- Could we adopt MD PnP in a wireless network environment?
- What are the new challenges?



# Challenges

- **Fault model**:
  - Both the network and supervisor software may fail during the medical operations
  - The supervisor runs on a commercial computer which is not certified as a medical device
  - The communication might not be reliable
- **Assumptions**:
  - Client adapters and medical devices are certified safe component (Class-3)
  - The rate of failure could be negligible during the service of operations

# The Open-Loop Safe Problem

- An open-loop safe system is a system that satisfies its safety constraints while the communication is not guaranteed.
  - Ex: commands to resume oxygen supply could suffer long delay => Could violate safety constraints
  - Ex: commands to query device status might not arrive => Unknown system states, cannot perform surgeries
  - Ex: acknowledgements might not reach the other end => Unknown the status of a sent command

# Research Questions

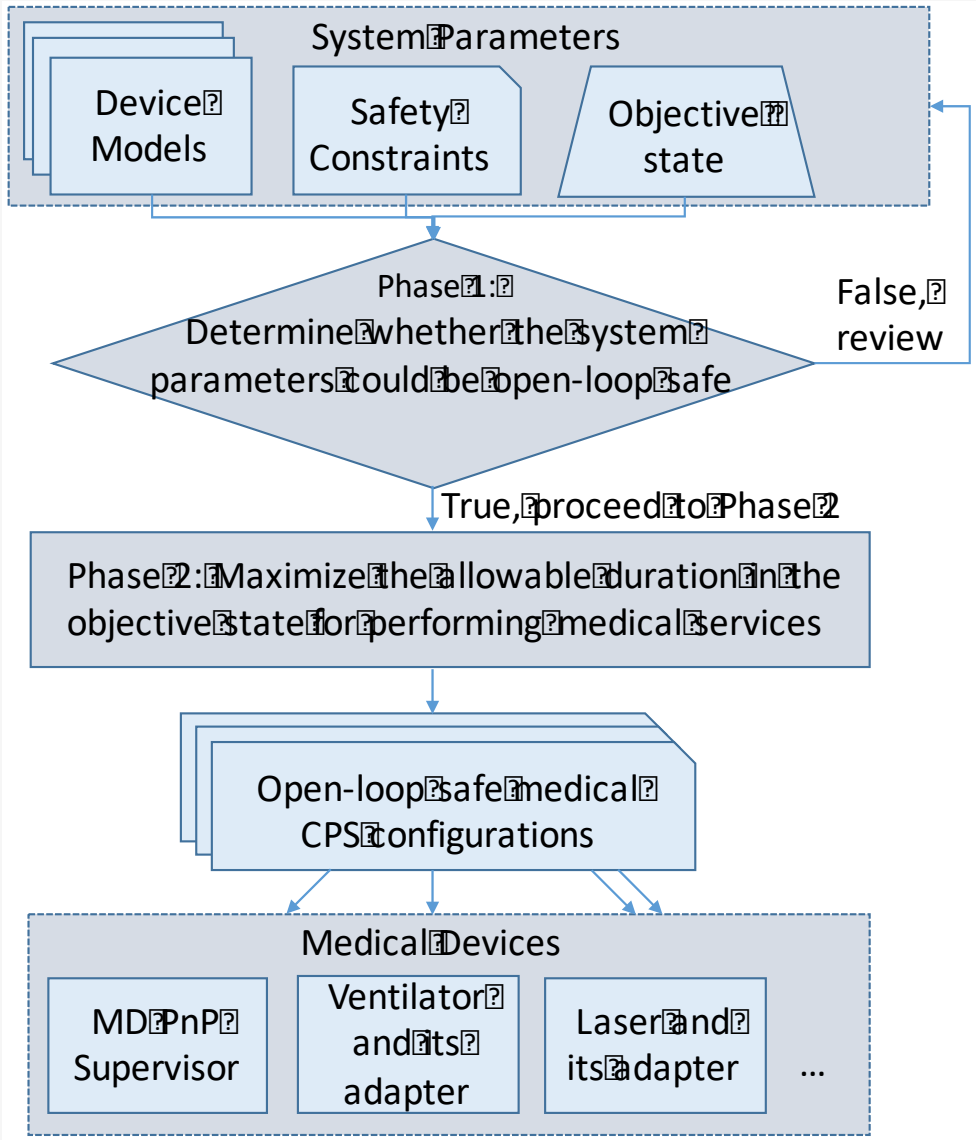
- Given a system of devices, devices status, safety constraints, system state to perform medical operations,
  - Can the system operate open-loop safely to perform the medical operations?
  - If the system is open-loop safe, what are the possible system transitions?
  - How can we find a path of system transitions that has the longest time in a system state allowing performing medical operations?

# Contributions

- This work
  - **Provides** a workflow toward developing an open-loop safe system
  - **Derives** a series of open-loop safe transitions as a foundation for systems with multiple medical devices
  - **Incorporates** safety constraints of interactions between devices.
  - **Assists** to select a system transition path that can allow medical personnel with the longest operation time for performing surgeries.

# Workflow for Open-Loop Safe System Developments

- **Phase I**: decide the existence of an open-loop safe path given a system model, safety constraints and the objective state.
- **Phase II**: find out the path that can allow a system to stay at for the longest period of time to perform the medical task.

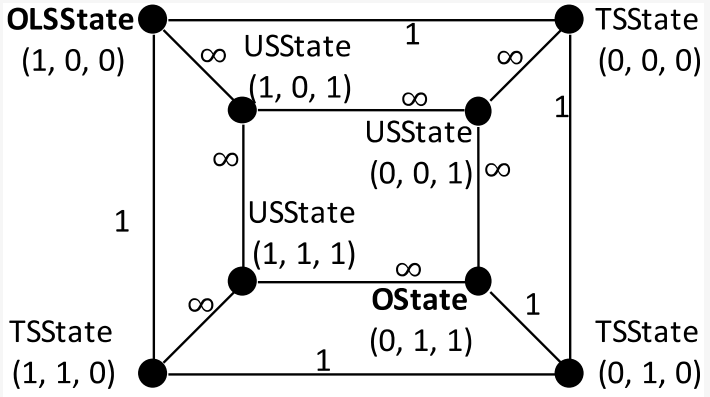


# System Models

- An open-loop safe system model is a three-tuple, $(S, SC, P)$
- $S$ : set of system states $s = (d_{1.status} \ldots d_{i.status})$, each state $s$ has a type
  - Open-Loop Safe State (OLSState)
  - Transient Safe State (TSState)
  - Operation State (OState)
  - UnSafe State (USState)
- $SC$: SafetyConstraints is a two-tuple, $(s_i\ ,\ p_i)$ where the system is allowed to stay at $s_i$ state for $p_i$ unit of time each time.
- $P$: an open-loop safe path $p$, $p$ has a source state, a destination state and a series intermediate states between the source and the destination state.
  - Ex: OLSState -> TSState -> TSState -> TSState -> OState (roll back to OLSState)

# Determining an Open-Loop Safe System

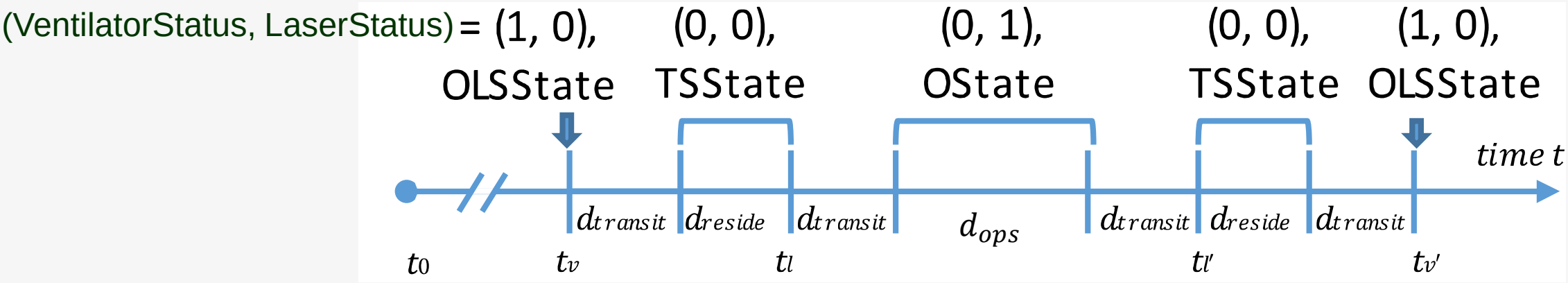
- 1) Construct an undirected weighted graph based on the given system states and safety constraints
  - Ex: state distance of (1,0,0) and (1,1,0) is one -> an edge with weight one
  - Ex: distance of (1,0,0) and (0,1,0) is two -> an edge with weight infinite
- 2) With the graph, next, we use the shortest path algorithm to find out the shortest path.
  - Compare the length of the path with the state distance between the source and the destination state.
  - OLSState (1,0,0) to OState (0,1,1)
    => state distance is three



# Finding the Open-Loop Safe Path for Surgeries

- **Transient Safe Period (TSP)** is a period of time that a device can stay at the certain status so that the whole system remains safe temporarily.
- A TSP for a device can be configured as a timer by an OLS-Client adapter
  - A device changes status when the timer starts
  - A device changes status again when the timer is fired
- TSP calculation example for laser in Airway Laser Surgery

$$d_{laser.on.timer} = d_{vent.off.timer} - 2(d_{transit} + d_{reside}) = d_{ops} + 2 \times d_{transit}$$



# Finding the Open-Loop Safe Path for Surgeries (Cont.)

- With a list of potential paths, for each candidate path $p$
- Find the first device ($d_{init}$) associated with a safety constraint along the path $p$
- Consider the safety constraint from that device $d_{init}$ sequentially along the path
  - Gradually shrink the TSP of each device along the path.
  - If a device also has a safety constraint specifying the maximum limited period of time staying in the certain status, then we take it into account when calculating the TSP for the device => minimum of the two constraints.
- For the devices listed before $d_{init}$ on the path, we expand the timer period backward from the initial device to the rest of devices.

# Case Study

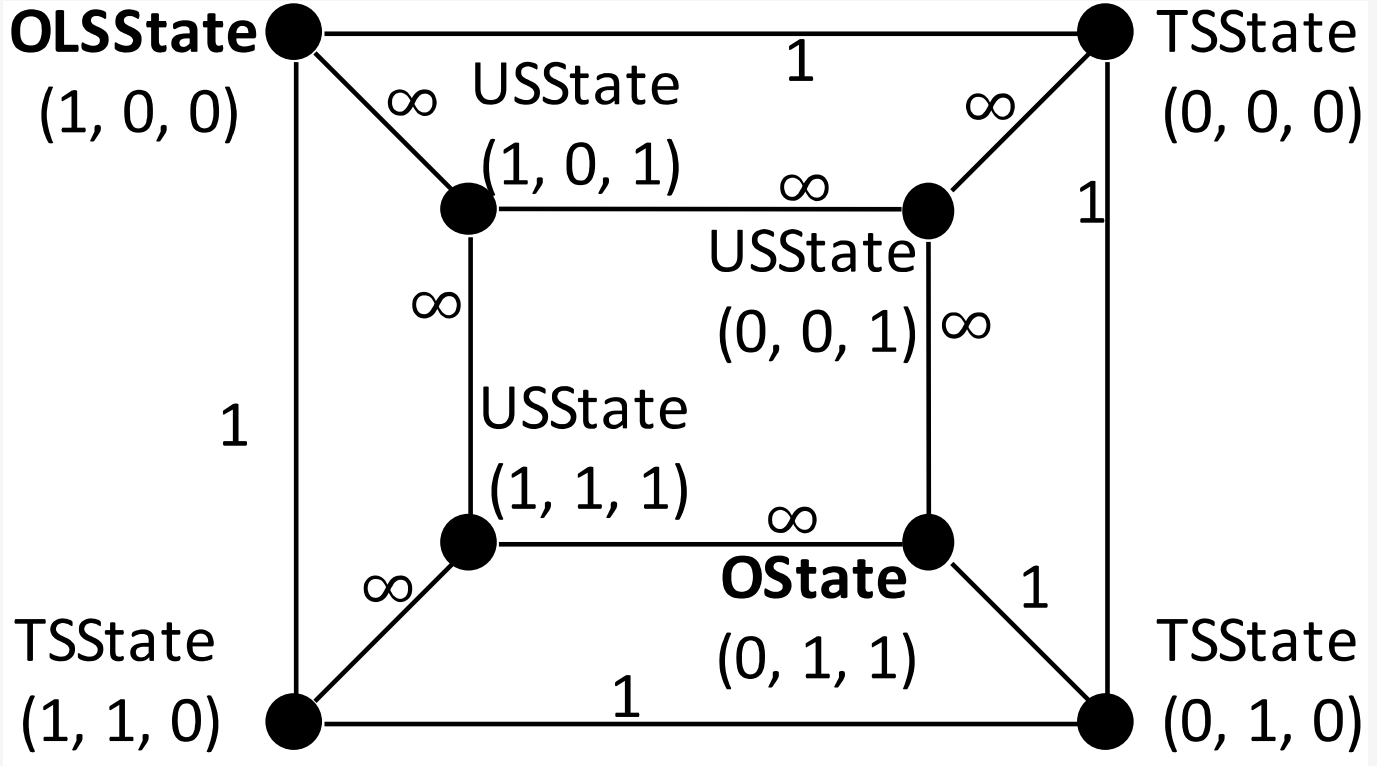
- Initially, an open-loop safe Tracheotomy surgery with two safety requirements
- Initial safety requirements
  - 1st No surgical fire
  - 2nd No brain hypoxia, Ventilator_Off_Max
- New safety constraints:
  - 3rd The laser scalpel can only operate safely and continuously within a period of time, Laser_On_Max
  - 4th Once the oxygen supply is paused, it is required to enable the plain air supply.

# Updated System States

System State Table

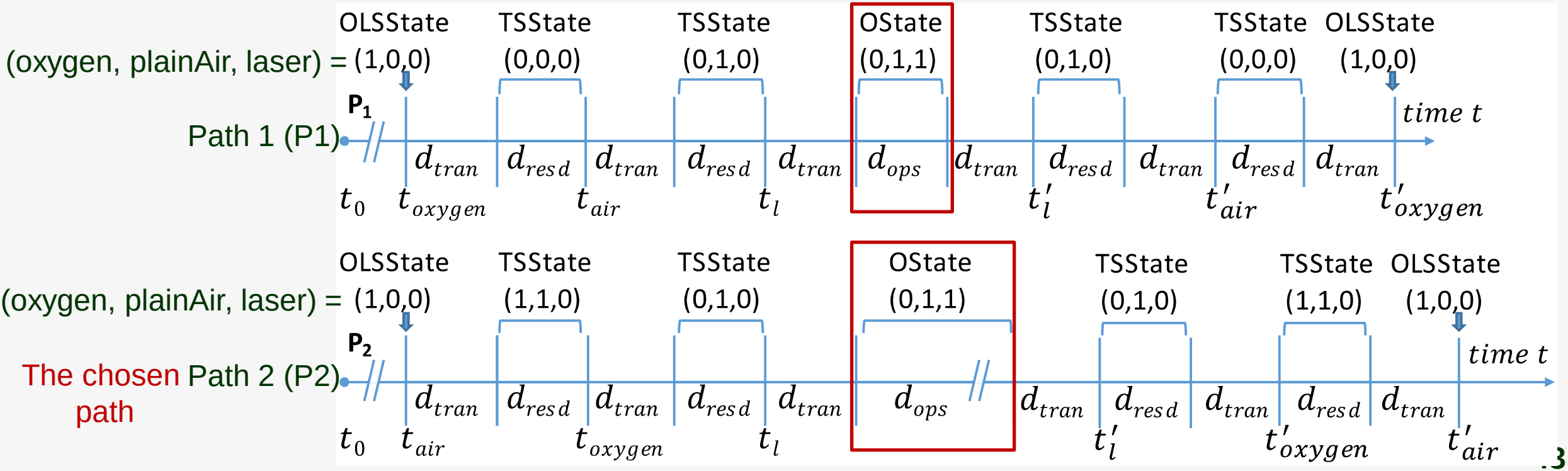
| State Types | System states (oxygen, plainAir, laser) |
|---|---|
| OLSState | (1,0,0) |
| USState | (1,1,1), (1,0,1) |
| OState | (0,1,1) |
| TSState | (0,0,1), (0,1,0), (0,0,0), (1,1,0) |

System State Graph



# Updated State Transitions

- Each path has the same weight of three, but the max operation time depends on the order of transitions on the path.
- P1: the oxygen is paused (0,0,0) and then the plain air is supplying (0,1,0)
- P2: first, the plain air is supplying (1,1,0), then the oxygen is paused (0,1,0)
- P2 has a longer period $d_{ops}$ for medical operations than P1



# Conclusion

- MD PnP in wireless network environment shall be able to counter against
  - 1) communication network failures
  - 2) supervisory computer crashes
- The paper suggests a framework toward achieving an open-loop safe MD PnP system by
  - Constructing System State Graph based on System Models and Safety Constraints
  - Generating system paths form the system graph
  - Finding the longest TSP for performing medical operations

# Future Work

- Other challenges remain:
  - Communication protocol to coordinate devices is needed
  - Dynamic system states because of medical device joining and aborting
  - Support multi-valued devices and complex transitions inside a device
  - Support other safety constraints, such as minimum staying period

Thank you very much!