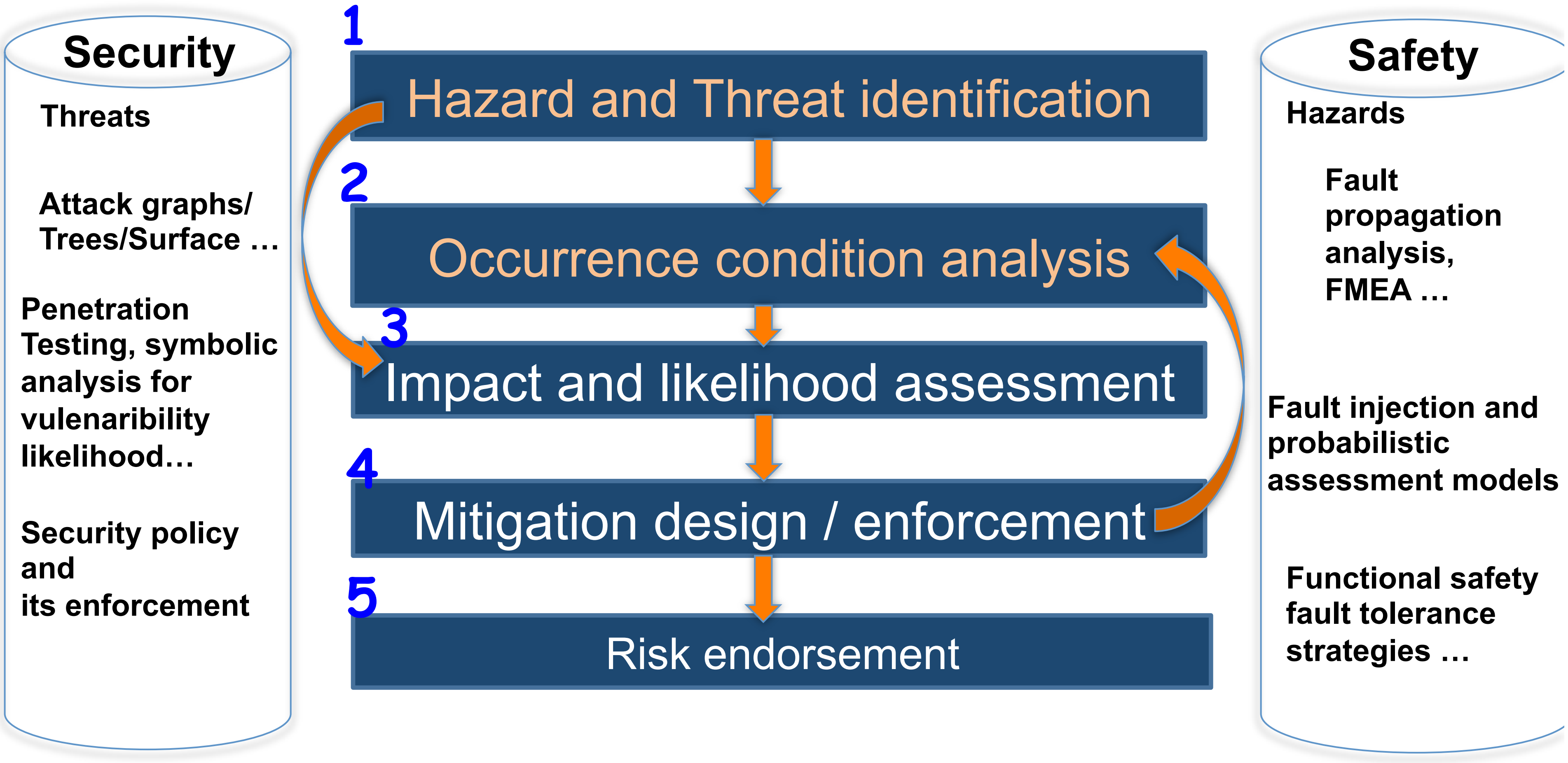
Formal methods to assist arbitration between safety and security requirements and assumptions

Works carried out by Sarah Nait Bahloul

Supervisors :
Thomas Robert, Yoran Halgand

Speaker: Thomas Robert

Engineering process: understanding motivations



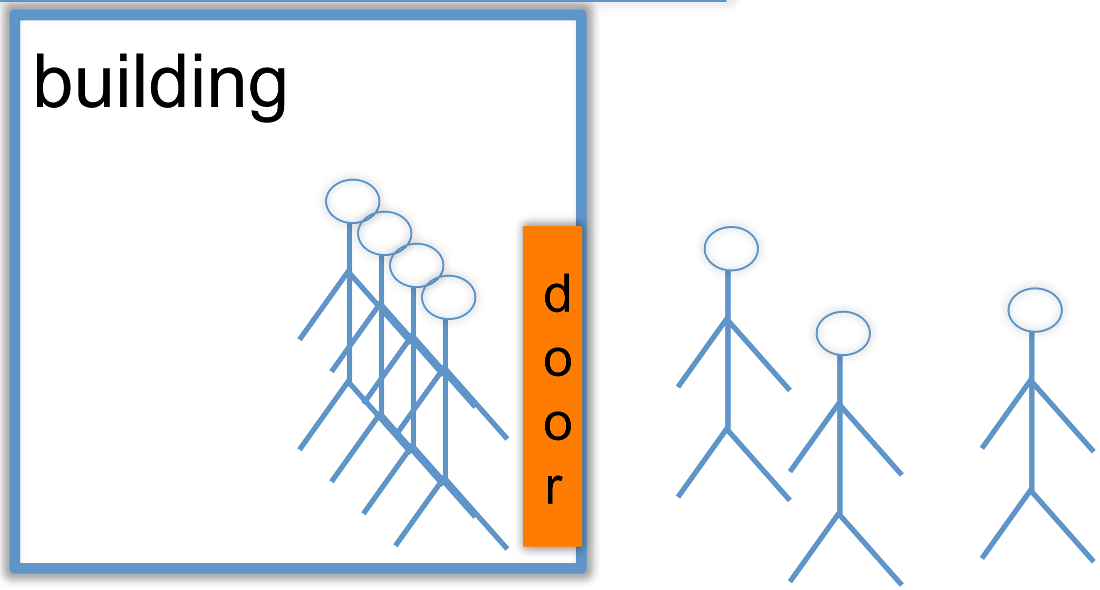
Example : building & door related concerns …

(first iteration) Reference model in litterature

Asset: persons, printed data

Security Goal: data confidentiality

Safety Goal: Person integrity



Example : building & door related concerns …

(2nd iteration)

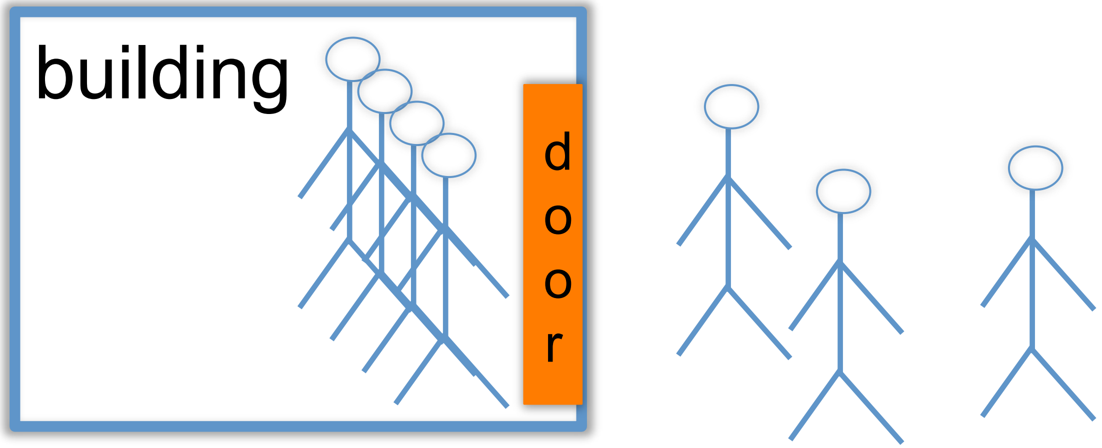
Security Goal:
data confidentiality

Asset: persons,
printed data

Threats: Intruder accessing data
attacker listening discussion

Safety Goal:
Person integrity

Hazards
Fire, starvation…



Difficult problems in secruity and safety engineering:

- **Capture risk causes and mitigation strategies**
  1. Identification of assets
  2. Hazard/threat identification, checklists, security threat profiles
  3. Account for background knowledge and best practices

- **Ensure risk analysis can be relied upon**
  1. Explicit knowledge representation
  2. Traceability, identification of risk causes
  3. Likelihood and severity models and ranking strategies

A push for Model Driven Engineering

- **State of the Art: Standards provide guidelines on**
  - requirements and analysis to carry out
  - Catalog of threats and risk causes…
- **Methods provide guideline on**
  - which information need to be gathered,
  - its level of details,
  - the process followed to collect it …
- **Models used to assist method enforcement through computer assisted manupilation / analysis**
- **Problem : many different type of information with different interpretations of methods and models**

Addressed problem in the study

Identify issue in merging risk causes models and mitigation strategies

Propose models and methods to assist engineers

Merging threat and hazard model and their mitigation

- **improve analysis coverage-accuracy + decisions**
- **example:**

	Threat	**Threat + Security Policy**
Hazards	Merged causes of risk => coverage	Import safety risks to security
Hazard + Functional Safety	Import security risk into safety	Full merge

- **Problems: increase the list of potential causes**

⇒ **More difficult to rank them, to interpret them**

⇒ **Potential gap in level of abstractions**

Understanding interdependencies between safey and security

Taxonomy based on the impact on mitigation success

Mitigation A designed to ensure goal GA
Mitigation B designed to ensure goal GB

- **Conflict :**
  mitigation A will fail due to action from B
  Goals GA and GB cannot be satisfied together
- **Mutual renforcement :**
  A improves GB satification likelihood
- **Functional or Conditional dependency**
  A ensure GA if B ensure GB

Our Claim

If we understand the type of information shared with respect to its role in risk cause modelling

=> We can predict the type of interaction that can be expected

Distinct risks with non independent mitigation stratgies

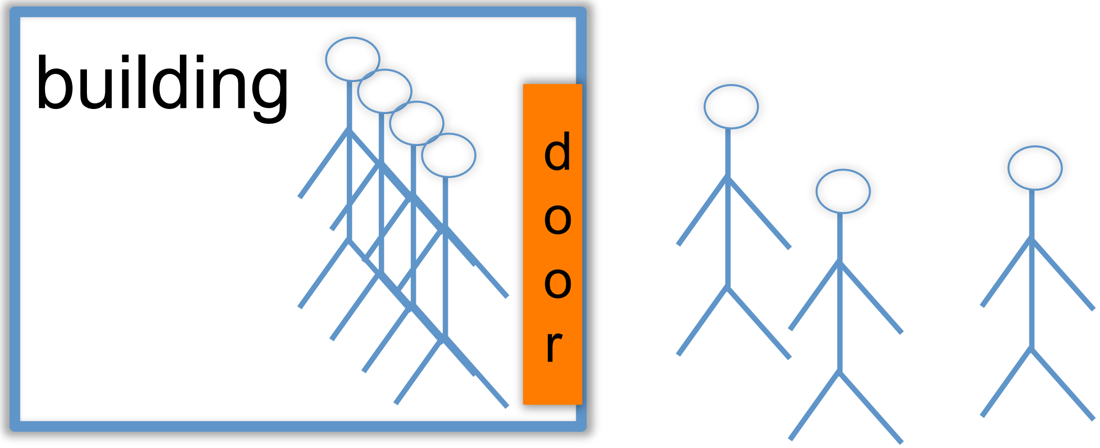
(2nd iteration)

Threats: Intruder accessing data

Asset: persons, printed data

Mitigation through controlling door state (open/close)

Hazards
Fire



Example : define security/safety goals

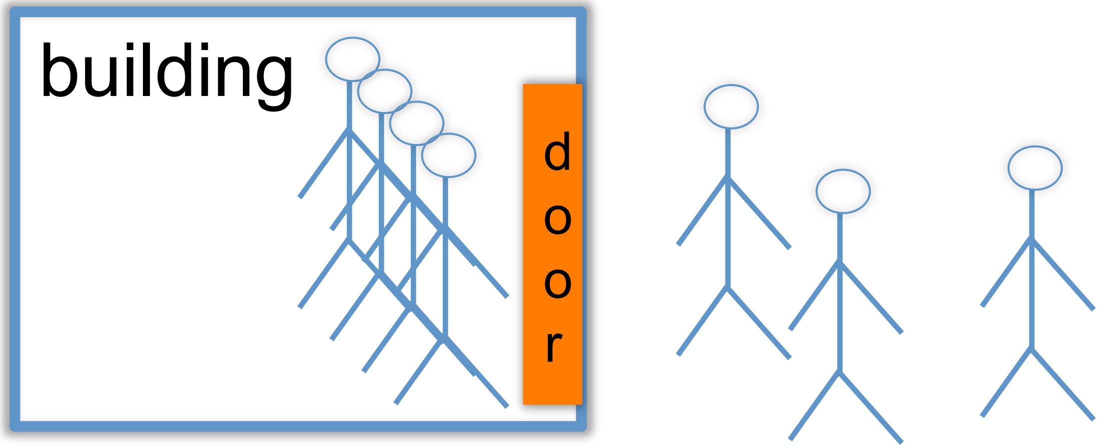
(2nd iteration)

Threats: Intruder accessing data

Asset: persons, printed data

Risk related to person flows
security goal : unauthorized person are outside
safety goal : avoid person blocked inside the building

Hazards
Fire



Consequences of events as state conditions

- **Use variables to describe system architecture**
- **Use variables to describe sub-system states**
- **Use variables to describe non functional states :**
  - Variables specific to hazard definition and status
  - Variables specific to threat definition and status
- **Use variables to describe mitigation mechanism state**
- **Use logical constraints to bind everything**

Remark description potentially eased using modelling langage such as UML/SysML or DSL like Figaro …

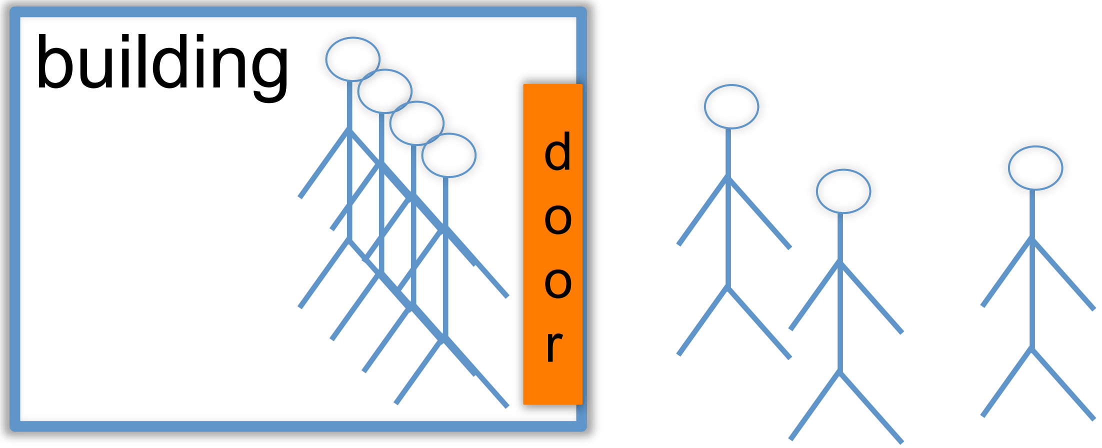
Example : events as state conditions on variables
Sun et al

Asset: persons, printed data

Authorized : boolean
locked : boolean

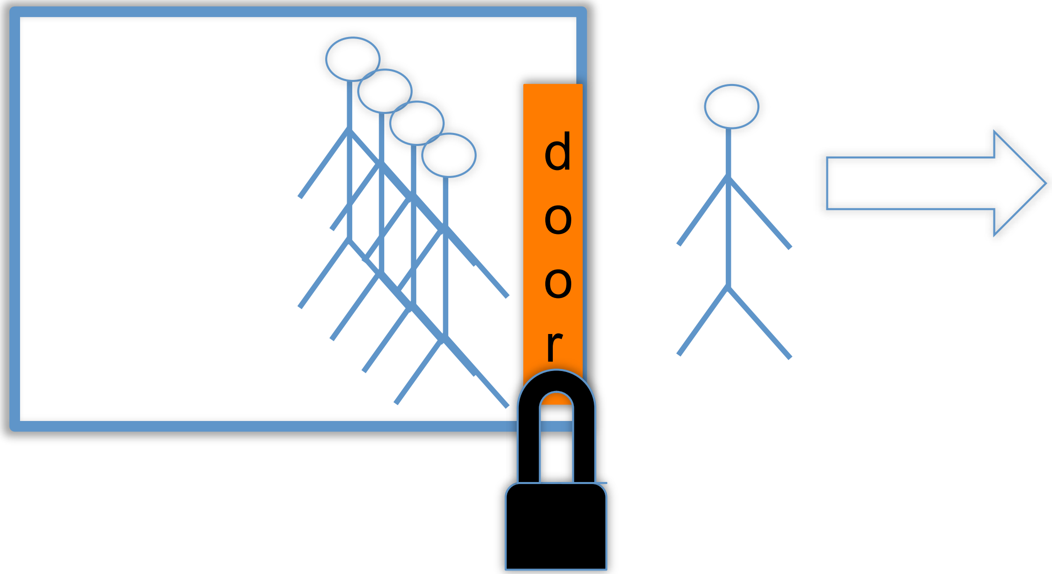
Door : {open, closed}, openable : boolean

Type : {fire,none}



Implicit interpretation of state variable to define undesired events

Door state impact on persons flows from inside to outside



Non ambiguious
no moves

Hazard for Persons : fire+no moves

Purpose of detailed formalization

Separate behavior or structure description from goals

not(authorized)=>door.lock

Ensure clear modelling of undesired system states

$$\forall p \in Inside_i, authorized(p)$$

Distinguish Assumptions from Requirements

Expected : Assumptions ensure Requirements
Analysis worthless if unrelated or trivially bound

Analysing risk cause–consequence

Analysis goal: determine undesired event cause or likelihood

Means :

- **event sequences (e.g. execution)**
- **causal relationship (inferred from background knowledge)**

Issues in model accuracy

The causal relationship between events is unclear,
or events seem highly unrelated,
or event occurrence conditions unclear ….

Issues in model consistency in risk causes or mitigation strategies

Causal relationship and event sequencing rules seem contradictory,

Choosing the type of modelling abstraction
Causality vs Execution

Causality centered model:
Do not necessarily capture system state dynamic

focus on causal dependencies to determine possible causes == correlation constraints

=> Useful for risk definition through invariants, forbidden instantaneous state configurations

Execution centered model

Try to provide a model for system state dynamic, allows describing sequences of states

⇒ Usefull for risk conditions expressed as state transitions

Remark : With enough detail both are equivalent ….

2nd Formalization : events as state conditions on variables

Variables to represent fixed entity states :

Detected_Haz:{fire,none}

door_state:{open,closed}

Locked : boolean

Variables used to represent set of entities

Inside:{p1, p2…}, Outside{p1',….pk'}

+ predicate to define their features

Authorized : Person → boolean

Understanding interdependencies between safey and security

Taxonomy based on the impact on mitigation success

Mitigation A designed to ensure goal GA
Mitigation B designed to ensure goal GB

- **Conflict :**
  mitigation A will fail due to action from B

- **Mutual renforcement :**
  A improves GB satification likelihood

- **Functional or Conditional dependency**
  A ensure GA if B ensure GB

false problems and false hopes

Misleading model + conflicts => weak design deadlock

Goal1 and Goal2 contradictory but Goal1 abusively strict or defined ignoring method guidlines

Over constrained models => coverage pb = artificial reinforcement

Hyp1 + Hyp2 => some state ignored abusively

Ensure the quality on each model before merging !!!!
Or
Provide guideline to control the quality of a unified description

Why this situation is misleading

iSO 27000 : security goals should target primary assets

⇒ Security goals specified on system state variables or dedicated security state variables ≠ mitigation

Example : authorized : dedicated variable
+ list of persons inside the building

Observation : often example are asymetric

- **Safety states clearly stated / separation goals vs mitigation**
- **Security goals specified as known security function configurations**

Diagnosing issue in risk cause models and merging

Idea: constrain risk cause model content to assist engineer in merging models

Determine the role of variable :
System / Risk Mitigation/Risk definition

A controlled variable = value can be fixed arbitrarily
example : lock,   counter example : detected_haz

What if we classify variables in « free »/ controlled

« mitigation » related variables => controlled

Safety hazard states => « free »

Allows defining guidelines on how to combines models with free/ controlled variables

The conflicting-security safety goals : diagnosis / handling

The door example => contradicting goal ?

Risk defintion : Detected_Haz, authorized(p): free
System state : door_state: controlled, Inside, Outside: free
Mitigation : locked: controlled,

Observation 1 :
Safety do not have controlled mitigation !!!
=> safety ensured avoiding door_state mitigation

Observation 2 :
Security Goal : not authorized(p) => locked mitigation usage (≠ undesired state)

The conflicting-security safety goals : diagnosis / handling

Diagnosis table logic :

1) **Determine Goal specification sanity based on variable class/role**
2) **Check if Goals are contradictory if yes go to 3)**
3) **Interact with engineer for exception identification (put in assumption : this is impossible)**
4) **Otherwise propose modifying goals through loosening one goal constraint (often requires more information to discriminate states)**

Illustration on the example with possible advice

1) **Analyse Goal sanity :**

Safety OK
Security : Mitigation control
(use in Goal specification mitigation controlled variables)

2) **Check conflicts: found**

(Detected_Haz=fire, Inside not empty, p1 in Outside Authorized(p1) =false, door_state=closed)
Goal Sec true => locked = true => door_state closed
Goal Saf true => door_state=open (false)

(Detected_Haz=fire, Inside not empty, p1 in Outside Authorized(p1) =false, door_state=open)
Goal Sec true => locked = true => door_state closed (false)
Goal Saf true => door_state=open (true)

4) **Suggest modifying security goal specification**

Avoiding false alarms ….

Check conflicts: found

(Detected_Haz=none, Inside not empty, p1 in Outside Authorized(p1)=false, door_state=open)

Goal Sec true => locked = true => door_state closed (false)

Goal Saf true

False conflict as door_state can be controlled => this state can be ignored safely

Classification of Diagnosis

False conflicts identification

Recommandation: skip

Conflicting Goals with mitigation based goals

Recommandation: refine Goal definition

Pure Conflicting Goals:

Partition the set of state in risk definition to allows trade-offs or alter system features

….

Formal definition of each case
=> can be automated and engineers = final decision

Classification of Reinforcement

Observation:
likelihood difficult to capture in causal dependencies

Interpretation 1:

Mitigation A works due to assumptions Hyp1 on free variables (uncovered exceptions)

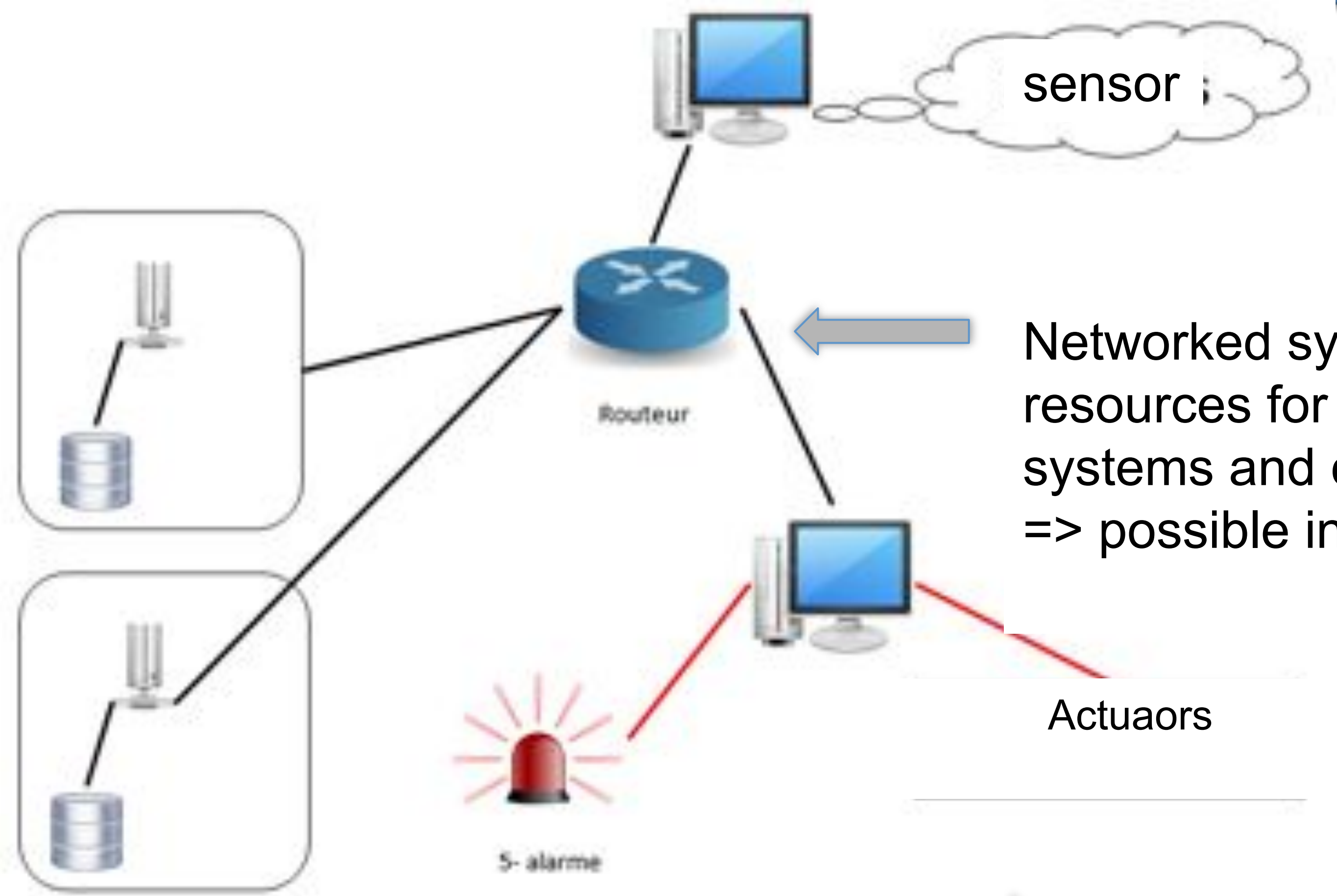
With Mitigation B some of these exception are covered…

Interpretation 2: (require additional variable typing)

Goals A satisfied ignoring some controlled variable values through mitigation A, mitigation B ensure Goals A even for these states.

…..

Complex system analysis



Networked system sharing resources for Information systems and control systems
=> possible interactions

Assume diagnosis say merging is OK
how to take advantage of it

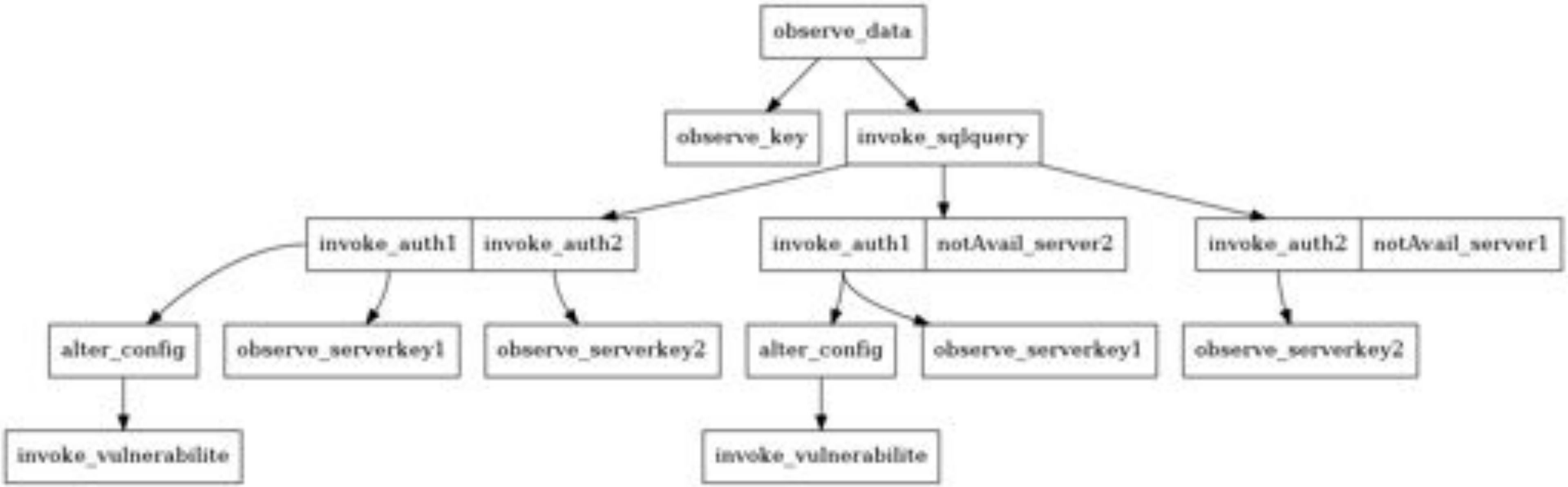
Find mixed causality

Reinforcing coverage if
Causality of undesired event
Reinforcement of mitigation
if constraints pile up

The implementation of mixed attack and fault tree

logical database to store background knowledge

- **vulnerabilities and possible impact**
- **Fault propagation logic (e.g. fault algebras)**
- **Mitigation impact reduction rules ….**



Conclusion

Results

- A method to handle merging risk cause models
- Refined case study with merging issues and benefits
- Tool support for automated reasonning

Future works

- Refine diagnosis rules
- Apply to larger case studies, or at different levels of abstraction (manage level of abstractions)
- Integrate to existing well spread modelling language