UNIVERSITY OF TWENTE.

DFTCalc:
A tool for Advanced Reliablity, Availability, Maintenance and Safety analysis.

Enno Ruijters, University of Twente
Supervisor: Marielle Stoelinga

ProRail

Movares adviseurs & ingenieurs

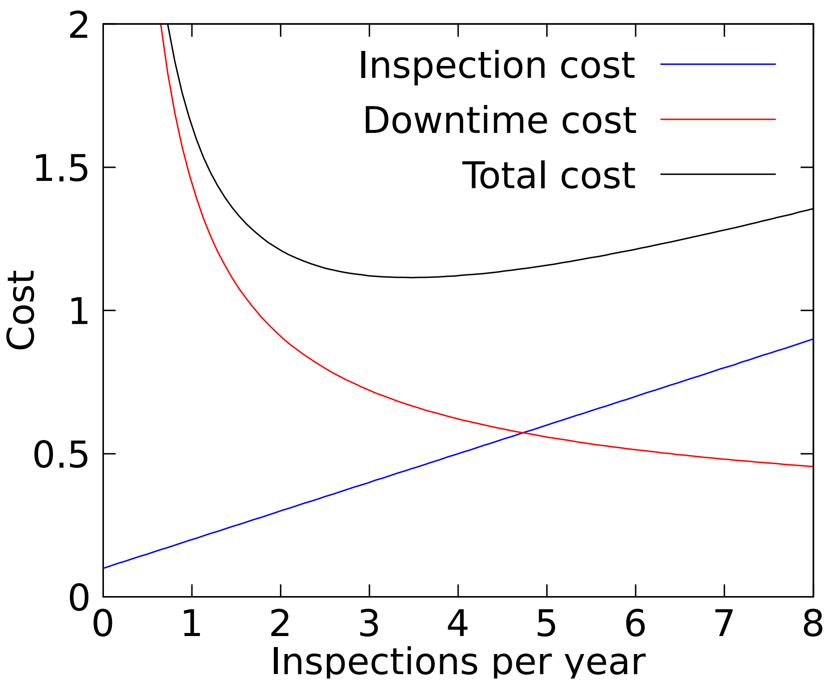
NedTrain

Reliability of critical systems

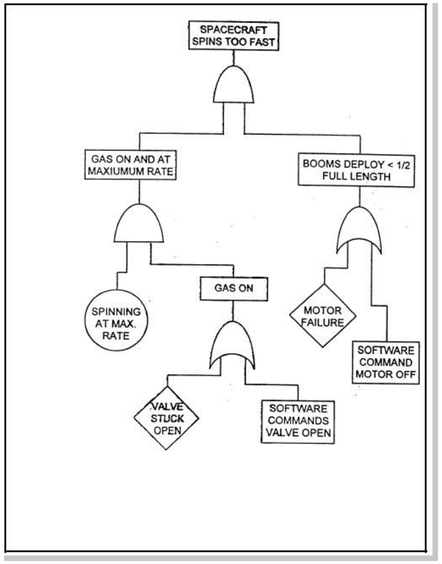
- System failures can be catastrophic
  - Airplanes, nuclear power stations, etc.
- How to ensure reliability:
  - At design stage: component selection, redundancy, diversity, isolation
  - During operation: Inspection, maintenance, repairs, replacement

- Effect of maintenace

  - Maintenance:
  - Improves reliability
  - Adds maintenance costs
  - Reduces costs of failure and downtime
  - Goal: Find cost-optimal maintenance policy

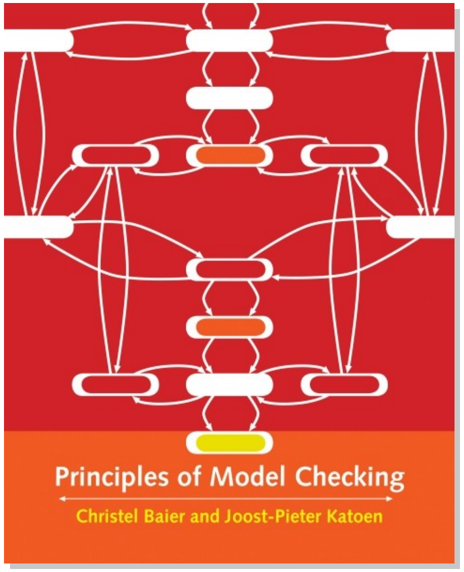


DFTCalc: 3 key ingredients



Fault Trees

Maintenance

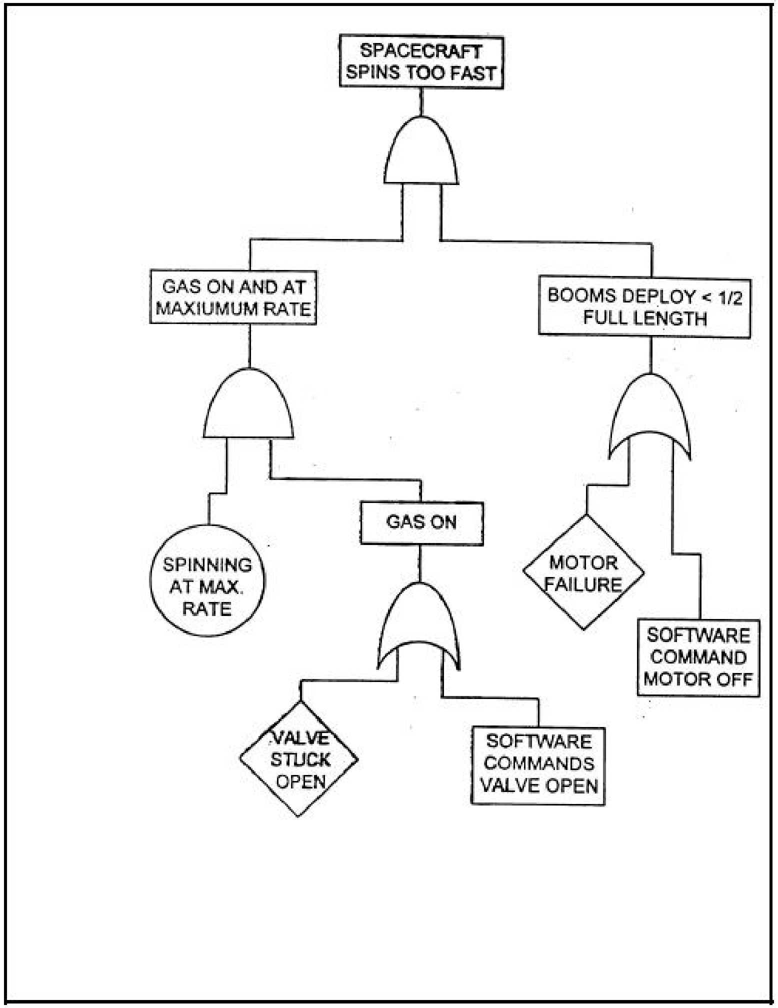


Model checking

DFTCalc analysis goals:

- What is the effect of maintenance on system performance:
  - reliability, availability, mean time to failures? …
- Can we do better (lower costs / better performance)?

Ingredient 1: fault trees



Preferred tool for RAMS

- *Model*
  - How do component failures propagate to system failures?
- *Analysis*
  - **P**[failure within mission time] (Reliability)
  - **E**[up-time] (Availability)
  - MTTF, MTBF
  - ….

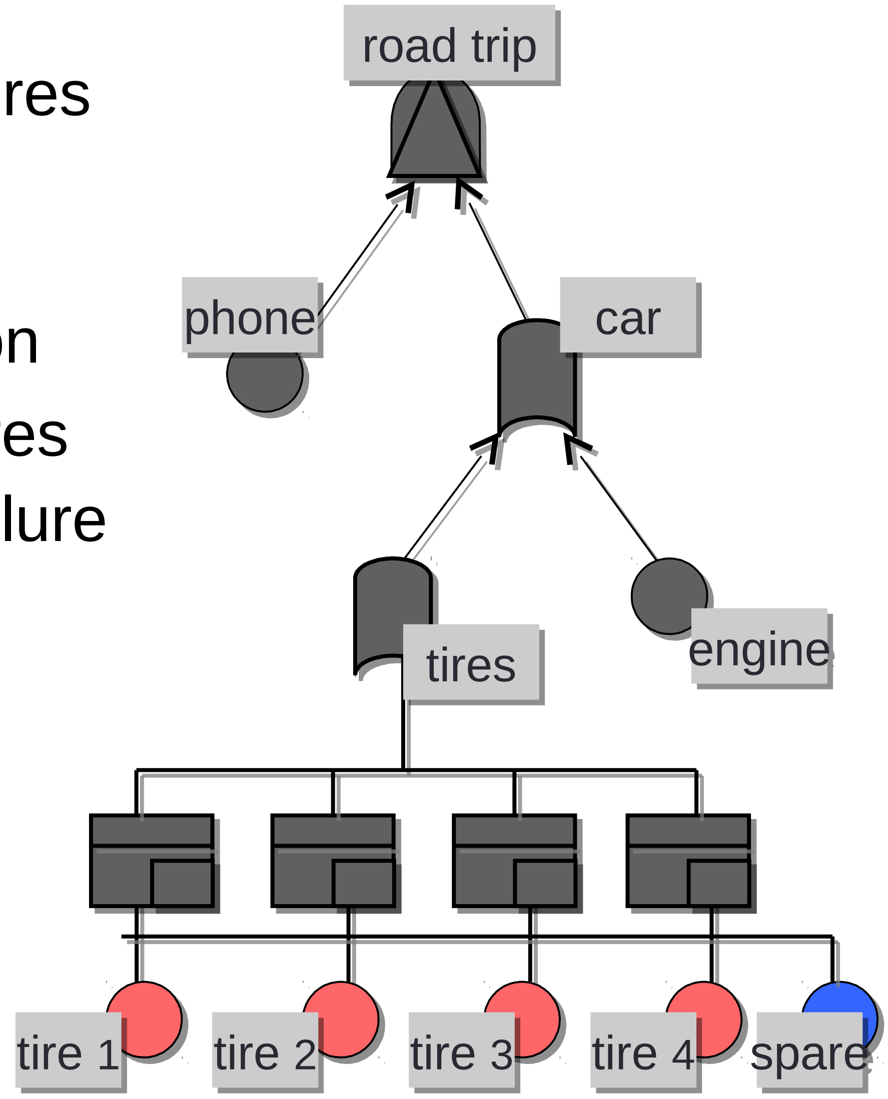
Talk:

- Add maintenance
- Large effect on MTTF
- Hardly considered

Ingredient 1: fault trees

Graphical formalism

- Decompose system failures into combinations of component failures
- Gates: failure propagation
- Leaves component failures
  - Traditionally contain failure rates/probabilities
  - We add degradation behavior
- Related: attack trees in security



fault trees: who uses them?

AIRBUS

BOEING

SPACEX

VolkerRail

delta π
performance improvement

Honeywell

ARCADIS

TOYOTA

NASA

esa

DB BAHN

Ingredient 2: maintenance

Types

corrective maintenance

preventive maintenance

Strategies

condition-based

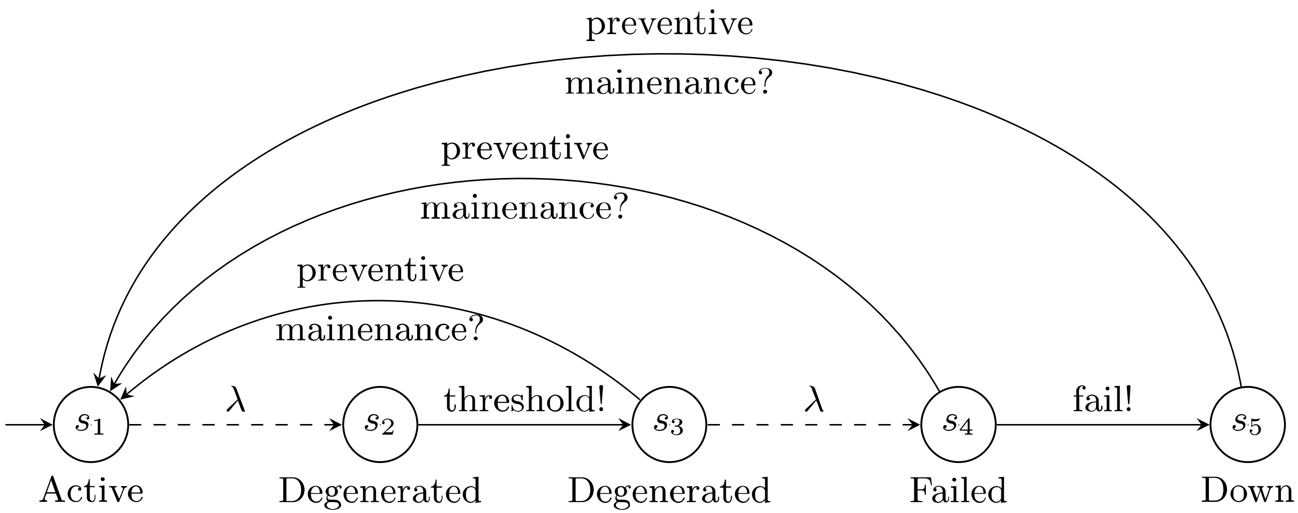
age-based

usage-based

Our approach

model these in FT leaves

Modelling: failure behaviour in BEs

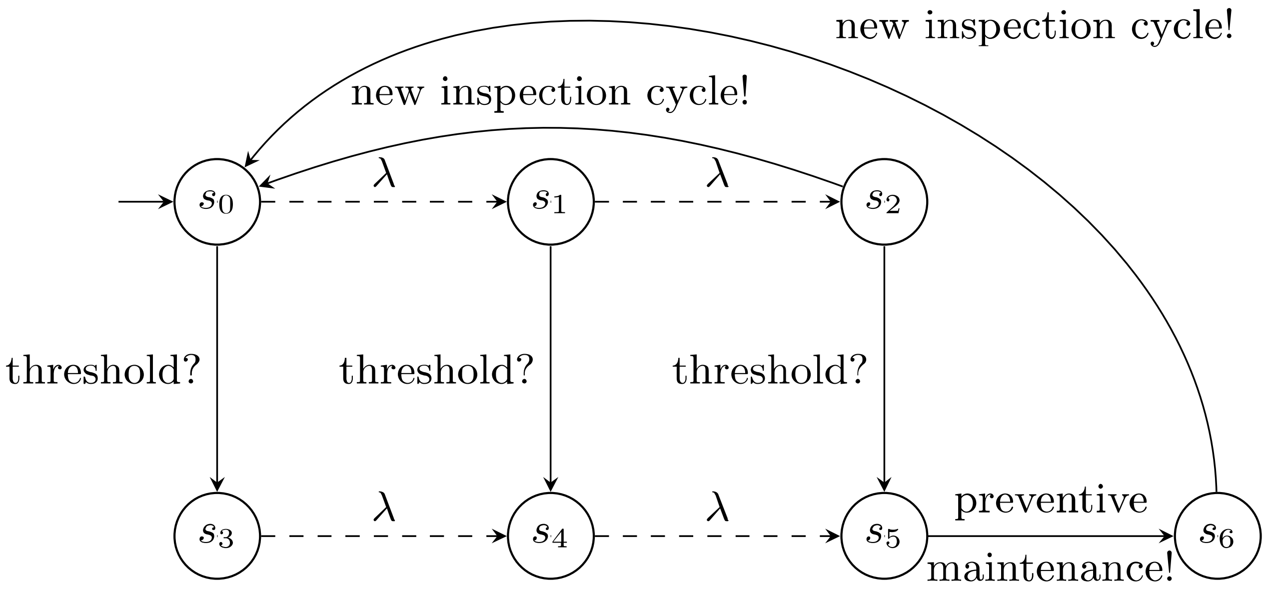


BE model

- Describes one failure mode / cause (eg from FMECA)
- Degradation behavior (phases)
- Detection threshold
- Maintenance effects

→ condition-based maintenance

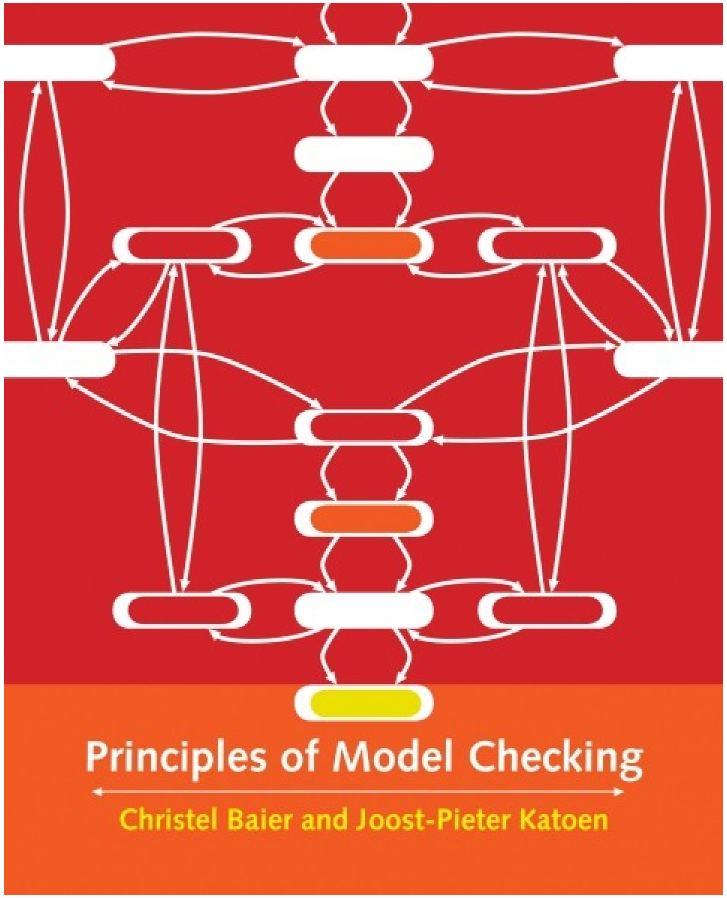
Modelling: Inspection module



Inspection module

- Above: dedicated for 1 components
- More complex for multiple components

Ingredient 3: (stochastic) model checking



Model checking

- state-of-art stochastic analysis
  - flexible, rigorous
  - used in HW verification
- 2007: Turing Award

2 flavors

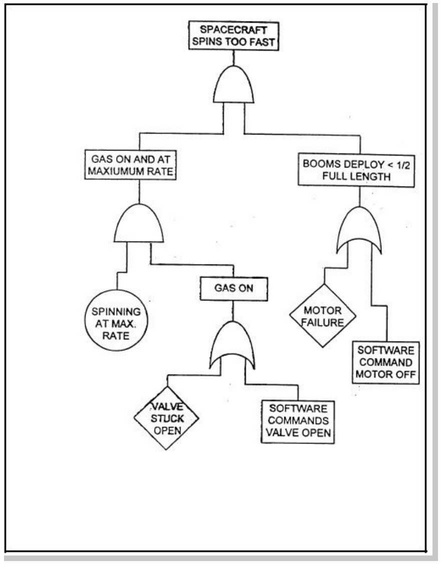
- verification: complete search
- statistical: simulation

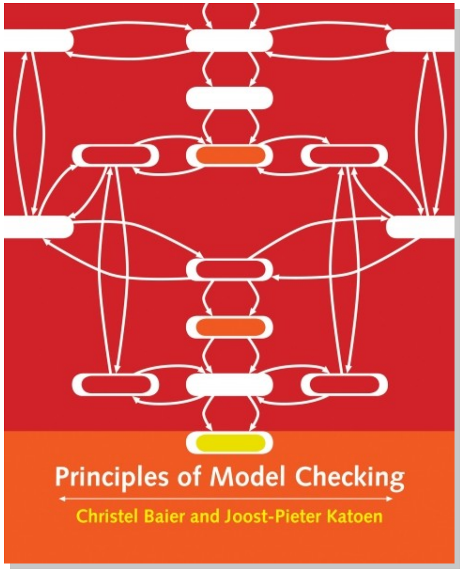
→ complimentary

Many tools

MRMC, Prism, UPPAAL, nuSMV, IMCA, ...

Recap: 3 key ingredients





Fault Trees | **Maintenance** | **Model checking**

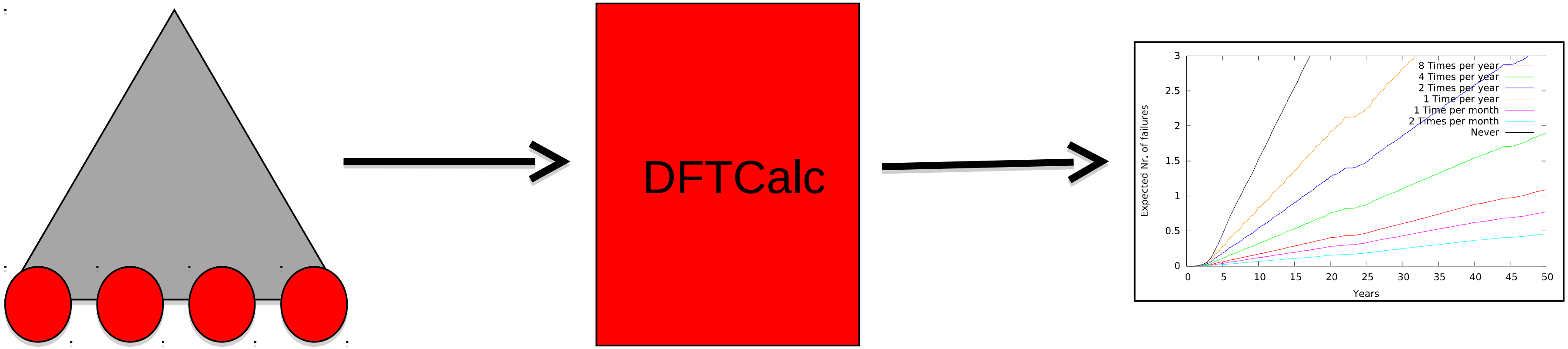
DFTCalc analysis goals:

- What is the effect of maintenance on system performance:
  - reliability, availability, mean time to failures? …
- Can we do better (lower costs / better performance)?

Outline

- Introduction
- Approach
- Case studies
- Conclusions

Our approach: how does it work?



FT + maintenance
•**Gates:** AND, SPARE
•**BEs:** failure behavior
•**IM/RU:** inspections, repairs

DFTCalc
• Extensible framework

Analysis
• system reliability over time
• mean time to failure
• availability

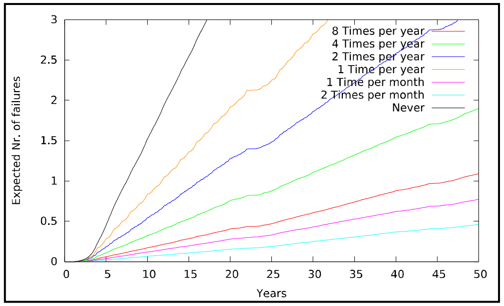
Questions:
•Does system meets reliability / availability requirements? Can we do better?
•What is the effect of different maintenance policies? (= different BEs / parameters)

Our approach: how does it work?

Fault Tree

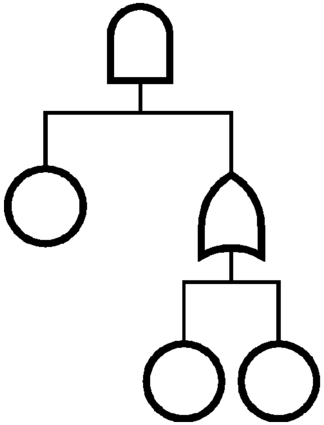
maintenance model

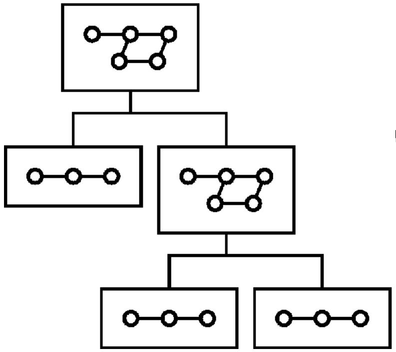
DFTCalc

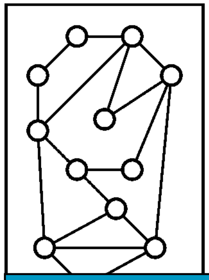


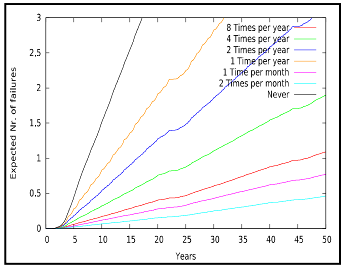
Translation

Analysis









Efficiency:

- Compositional aggregation
- Context-dependent state space generation

Our approach: alternative

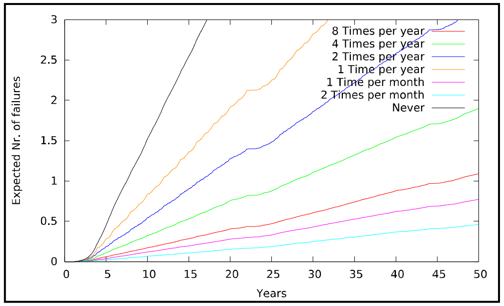
Fault Tree

maintenance model

→ Manual translation →

Uppaal-SMC

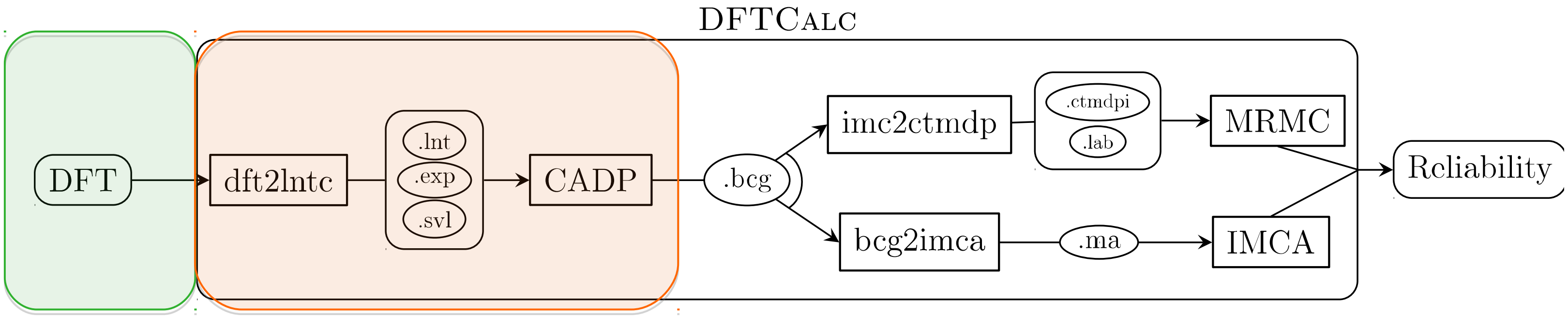
→



Benefits:
- Often much faster
- Supports arbitrary failure time distributions

Disadvantages:
- Results are less precise
- Can be much slower if high accuracy is desired

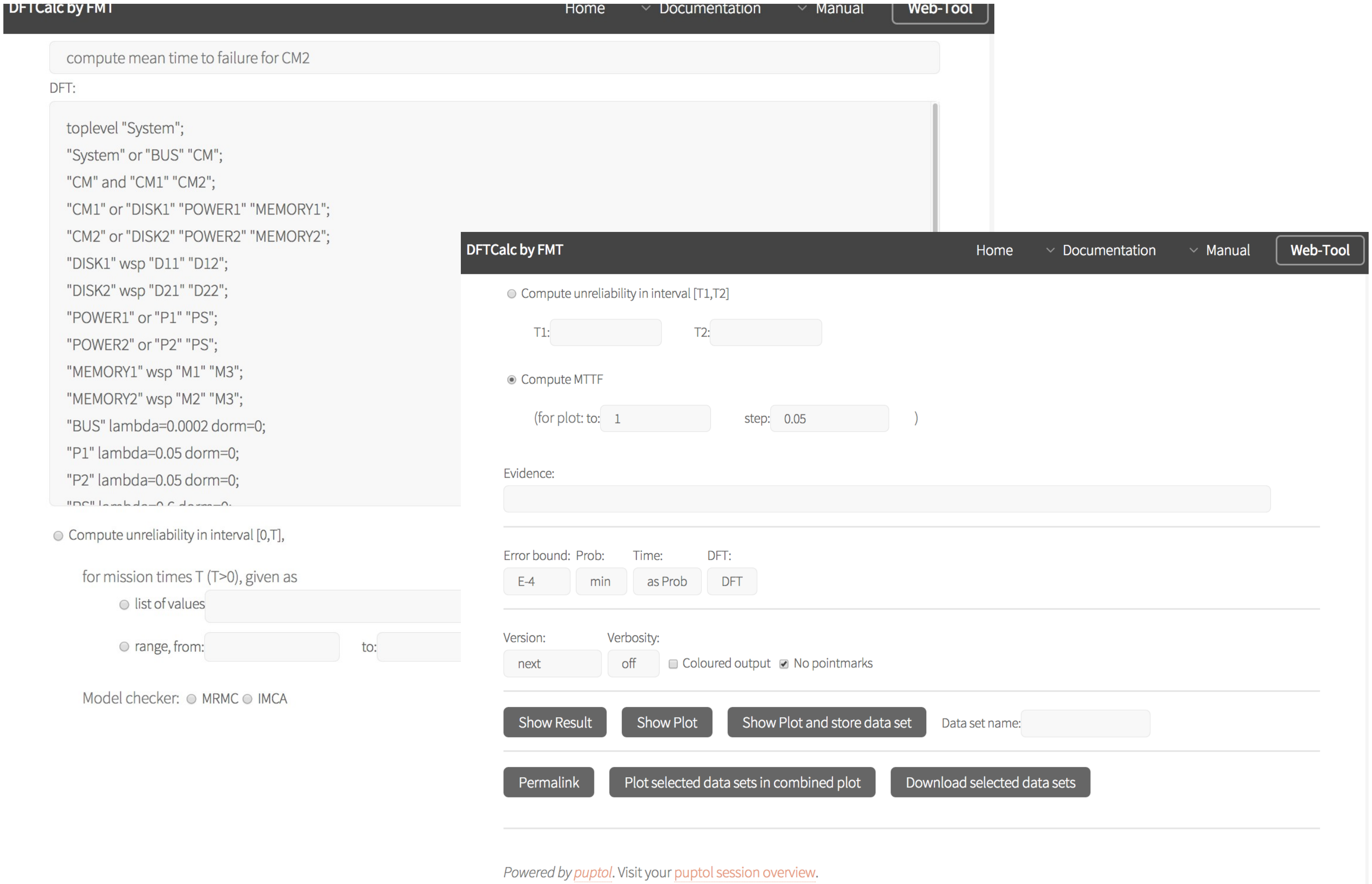
DFTCalc: Extensions



- New Inspection module
- New repair module
- New maintainable Basic Events

- Context dependent generation
- Inspection and repair communication

DFTCalc: web-interface



http://fmt.ewi.utwente.nl/puptol/dftcalc/

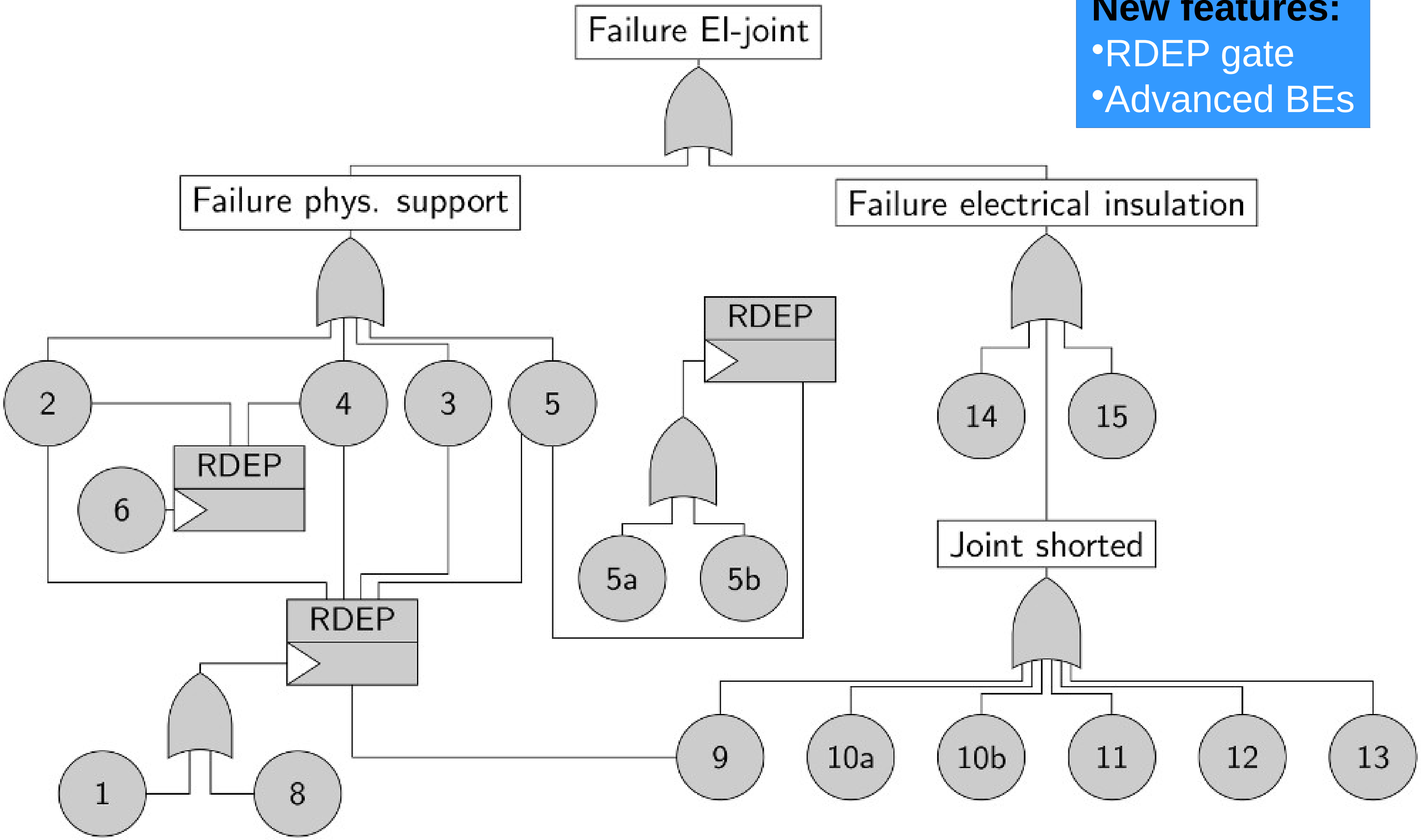
Outline

- Introduction
- Approach
- Case studies
- Conclusions

Case 1: Electrically Insulated Joint

- Electrically separates tracks
- 45.000 EIJs in the Netherlands
- Important cause of train disruptions

El-joint: modeling



New features:
- RDEP gate
- Advanced BEs

UNIVERSITY OF TWENTE.

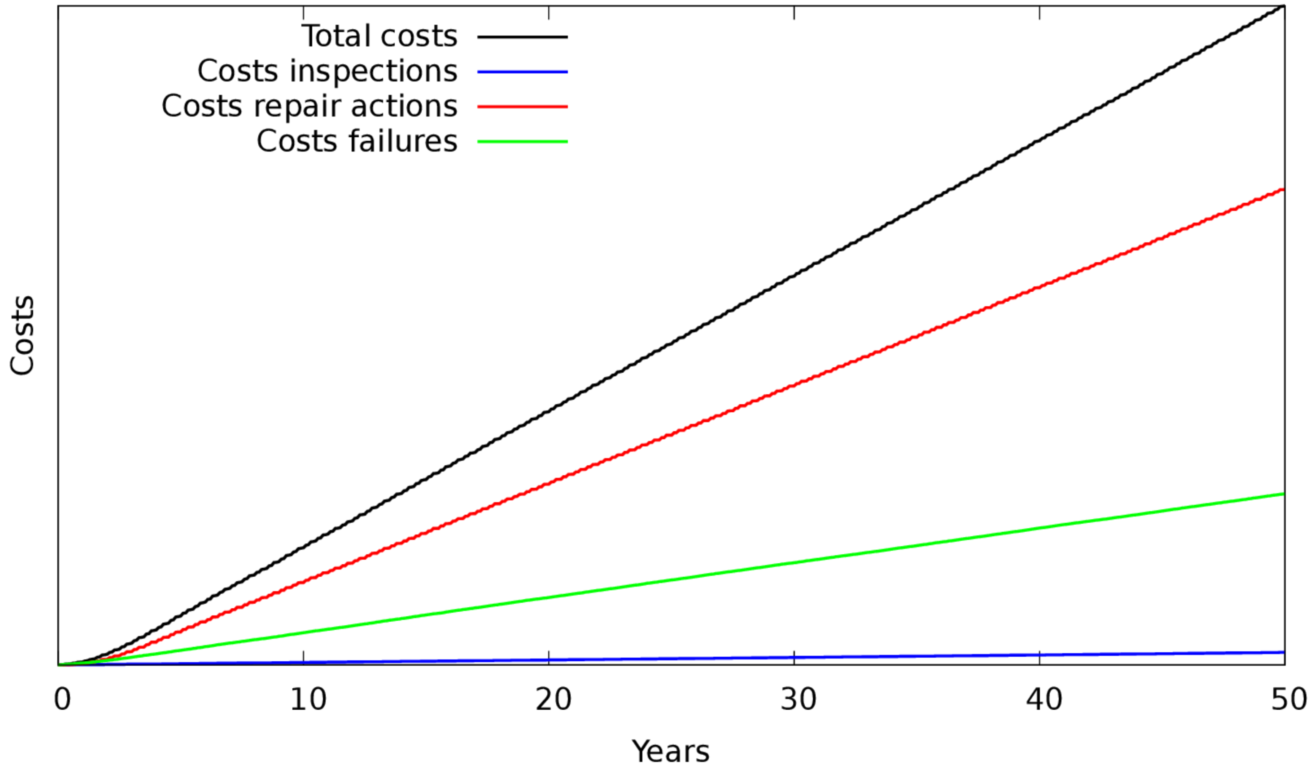
EI-joint: maintenance

Maintenance policy:

- Four trackside inspections per year.
- Repair action can either repair specific failure
  - (e.g. removing a foreign object)
- Or needs to replace the entire joint.

- Costs for inspections and maintenance actions are known.
- Costs for failures depends on how many passengers are affected.

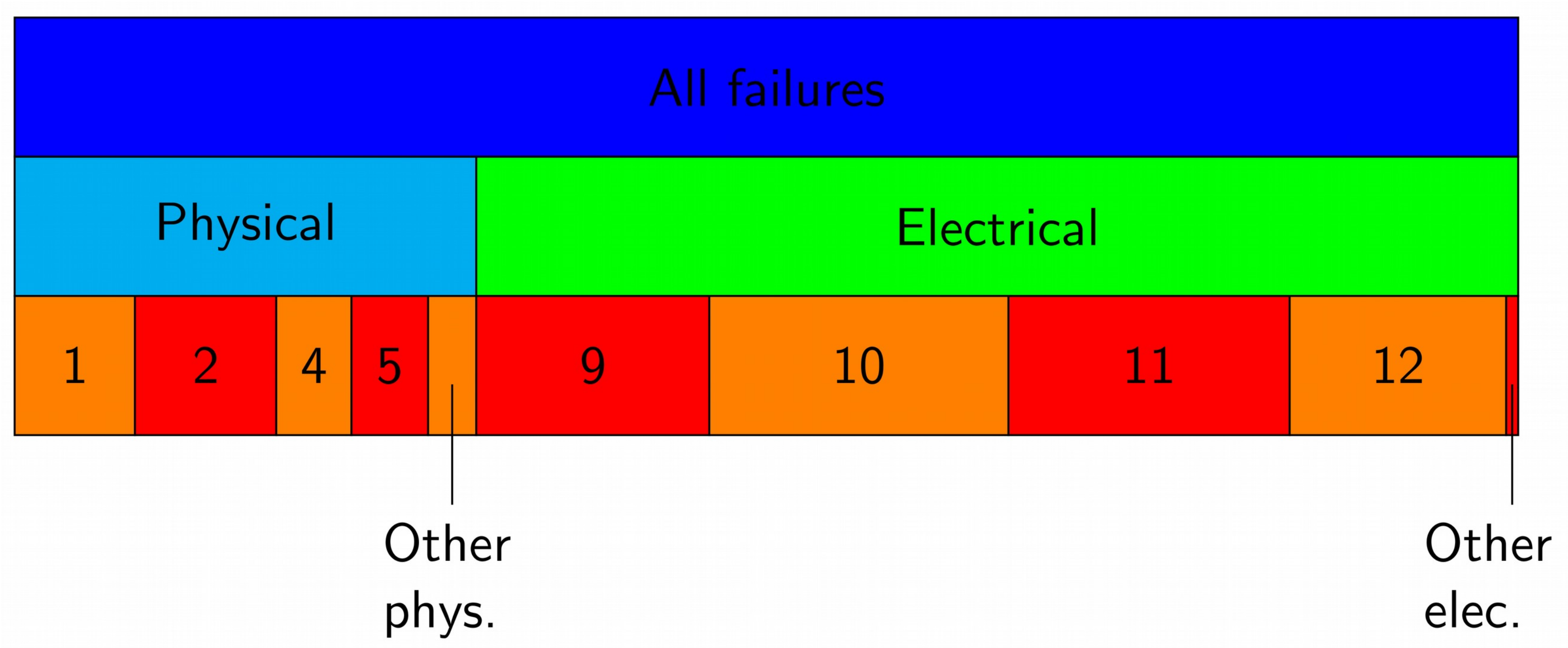
Results EI-joint: Current maintenance policy



Result:

- Failure behaviour is very linear after first few years.

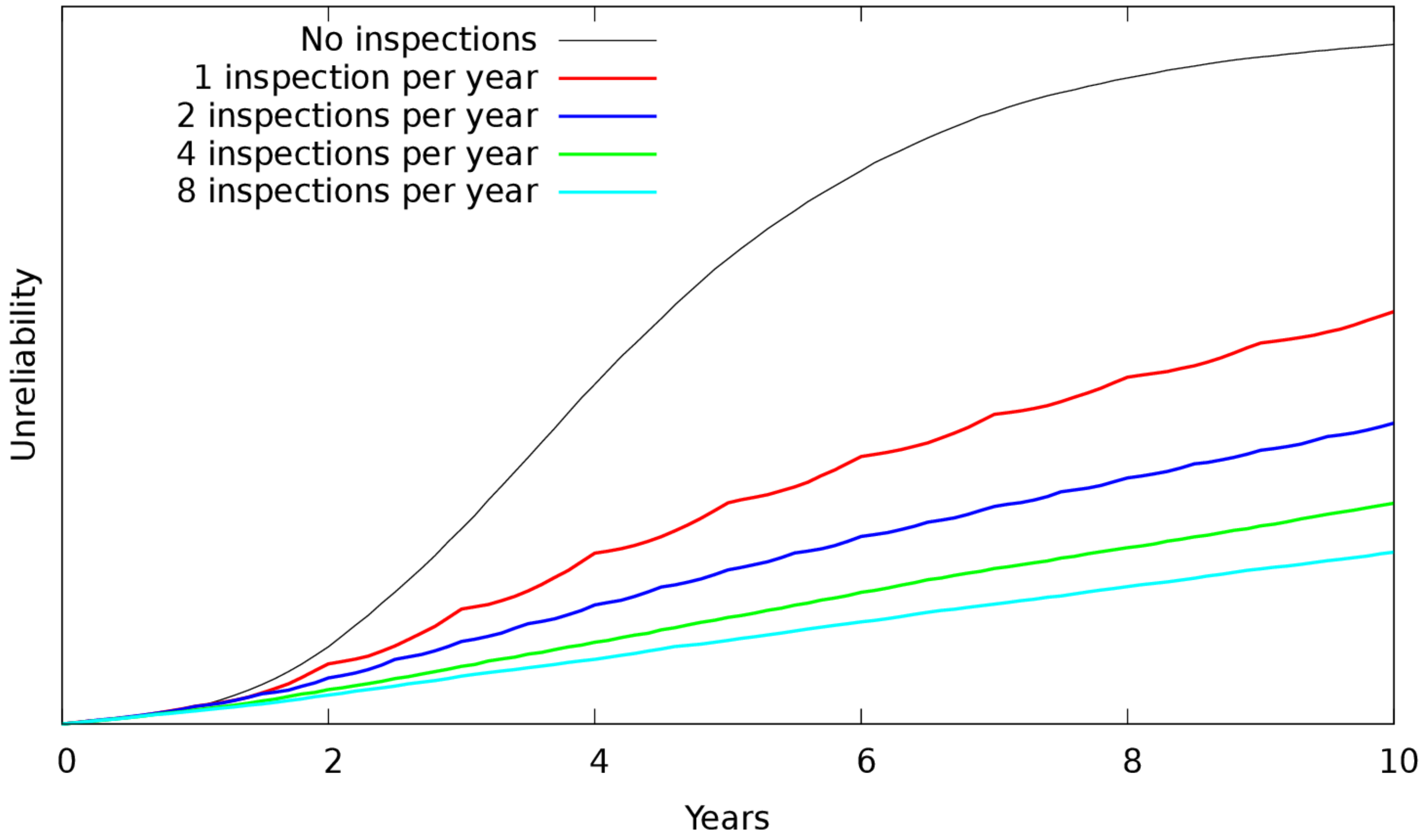
Results El-joint: Current maintenance policy



Breakdown of failure causes:

- Majority of failures are due to electrical insulation
- Almost all electrical failures are due to external shorts

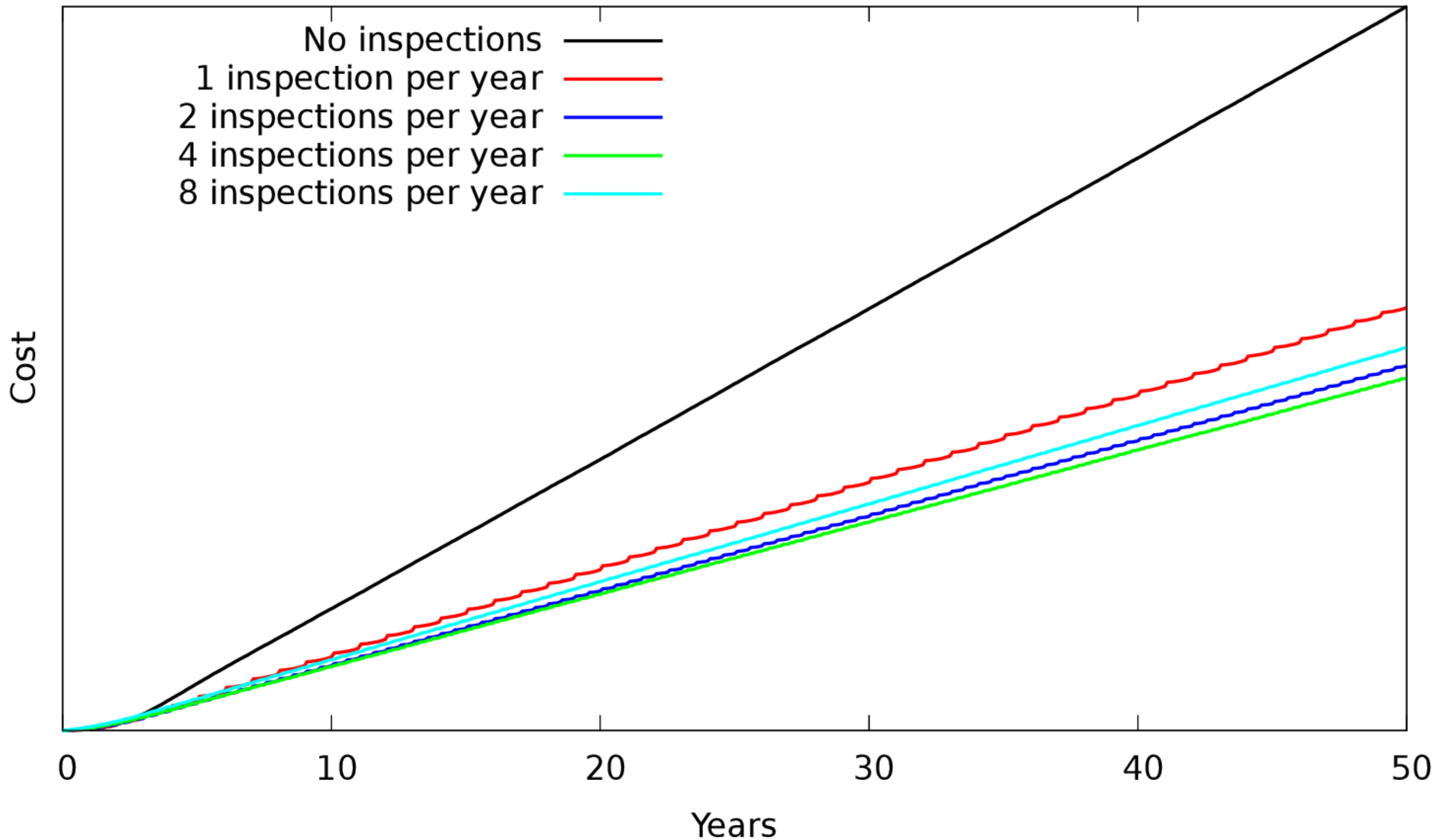
Results EI-joint: Different maintenance policies



Result:

- Inspections are clearly important.
- Does increased reliability lead to lower cost?

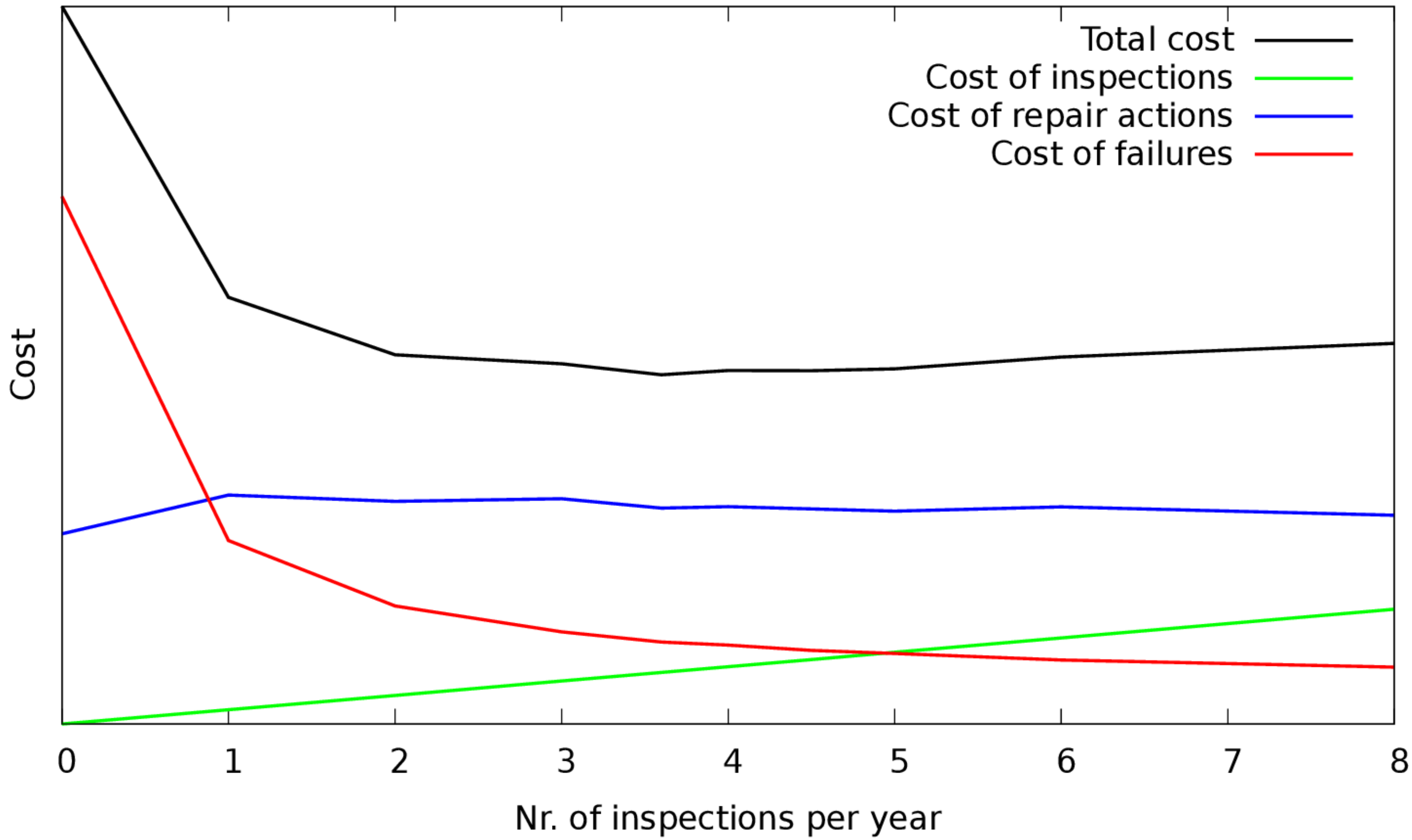
Results EI-joint: Different maintenance policies



Result:

- Inspections are important, but the exact frequency does not strongly affect cost.

Results EI-joint: Maintenance optimization



Result:

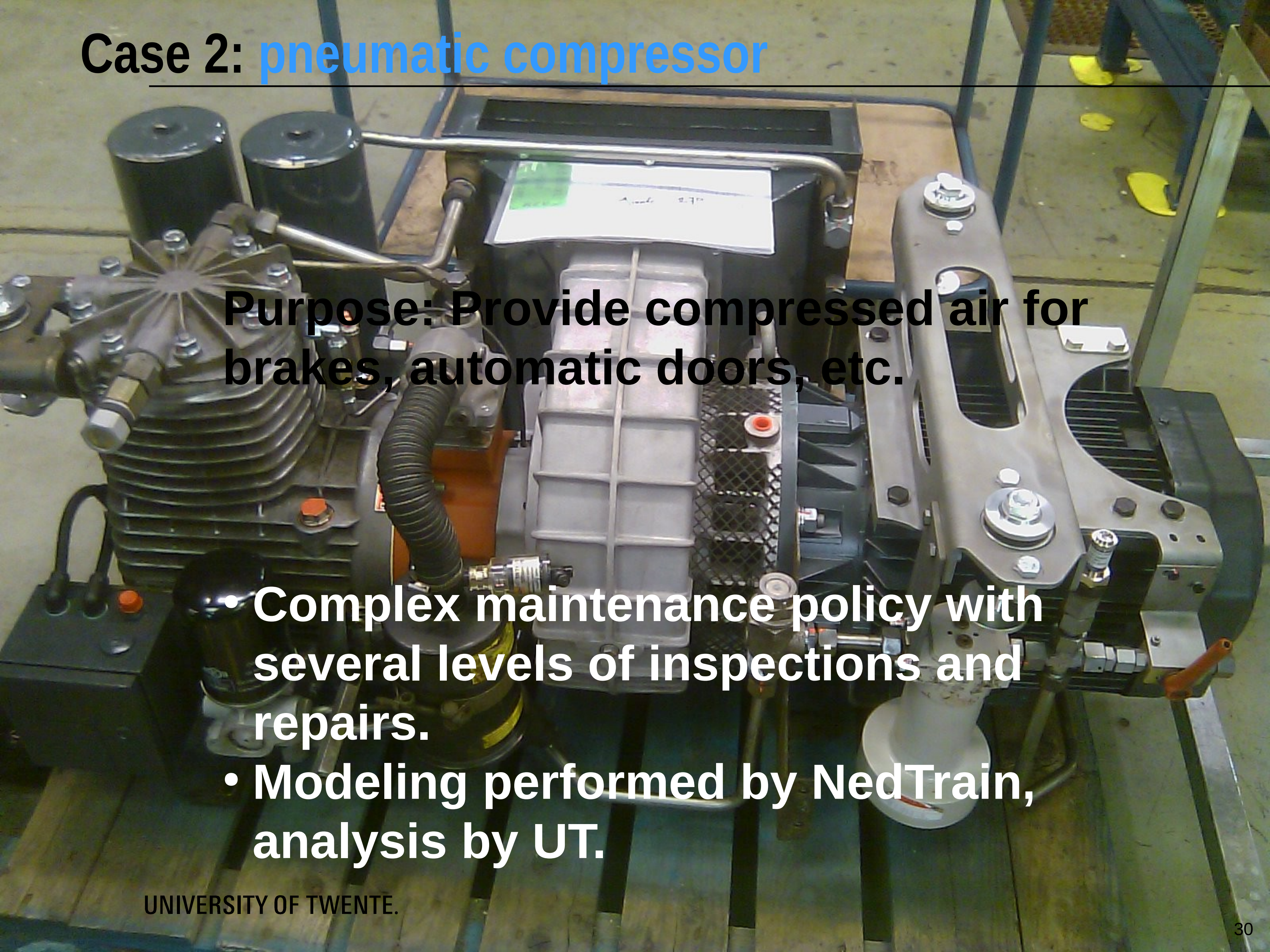
- Cost optimum around 3 – 4 inspections per year.
- Costs fairly constant between 3 and 6 per year.

El-joint: modeling process

- Fault tree based on existing FMECA by Prorail.
- Structure of FT is clear from context.
- Total failure rate per failure mode is documented.
- More details obtained using questionnaire to experts:
  - Variance of failure rate
  - External factors affecting failure (location, surface condition, etc.)
  - Translation of physical description of maintenance threshold ('>5mm vertical movement') to time-based description ('repair needed within 1 month')
- Tweaking and validation using recorded failure data.

- **Conclusions EI-joint**
  - Our model of the EI-joint agrees with reality under the current maintenance policy.
  - We find the cost-optimal maintenance policy consists of four inspections per year.
  - More inspections result in noticably fewer disruptions, but are not cost-effective.

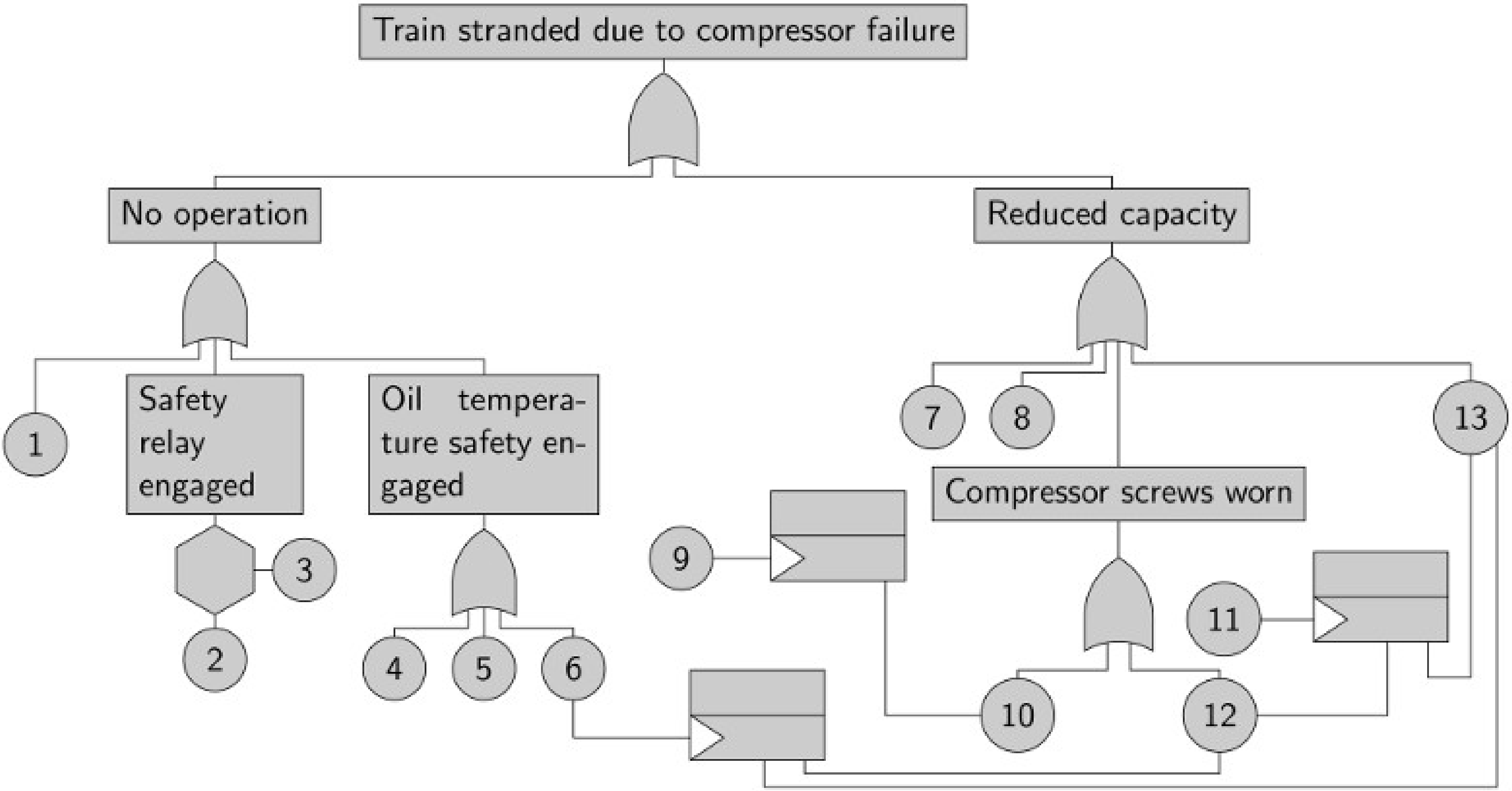
Case 2: pneumatic compressor



Purpose: Provide compressed air for brakes, automatic doors, etc.

- **Complex maintenance policy with several levels of inspections and repairs.**
- **Modeling performed by NedTrain, analysis by UT.**

Compressor: modeling



Similar features to the EI-joint fault tree

Compressor: maintenance policy

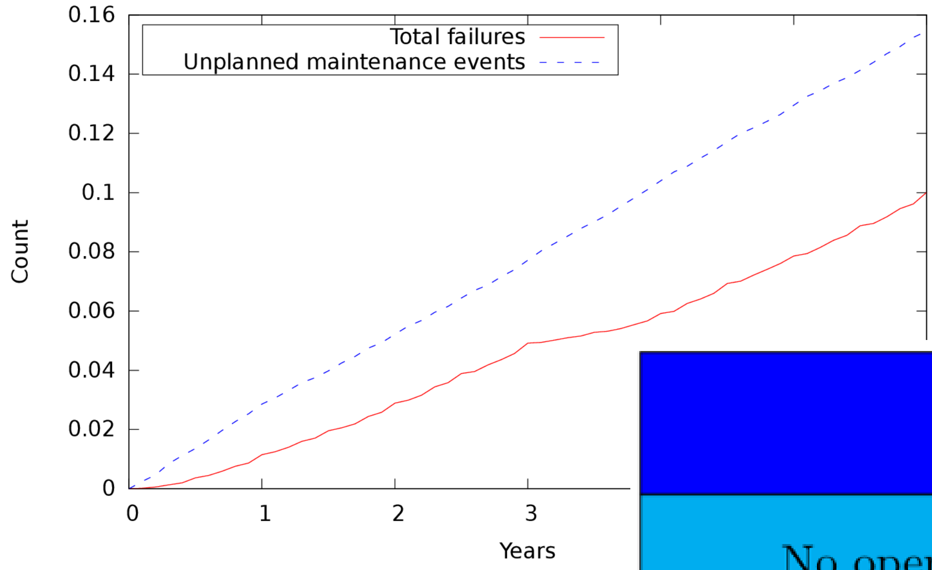
- Quick inspection every two days.
  - Check diagnostic computer logs for errors.
  - Visual inspection for obvious problems (e.g. oil leak).

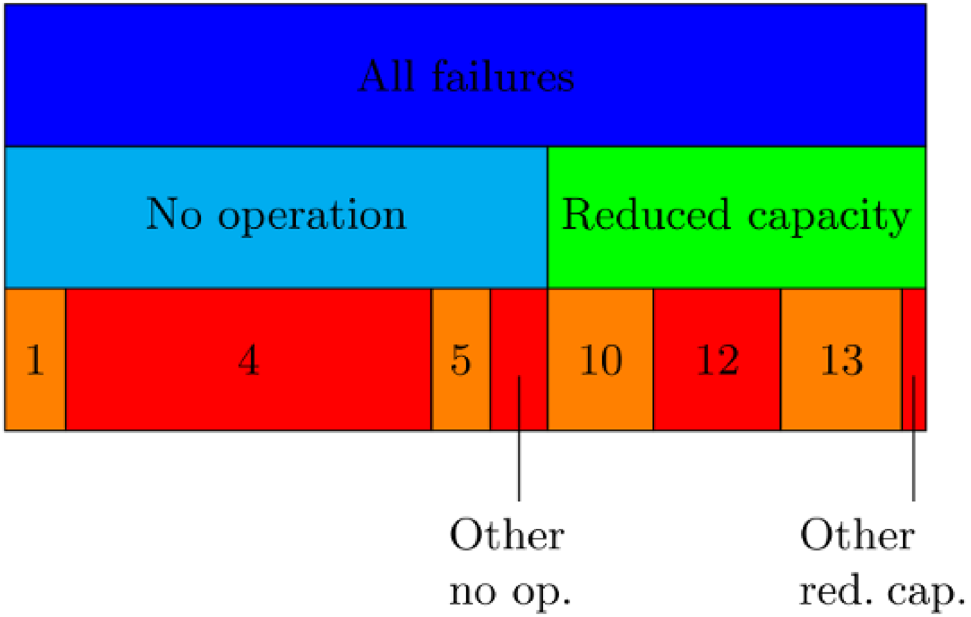
- Services every 3 months, more intensive every 9.
  - Replace consumables (e.g. filters)
  - Functional tests.

- Minor overhaul every 3 years, major overhaul at 6.
  - Compressor disassembled, components inspected.
  - After major overhaul, compressor is as good as new.

- At any level, if a fault cannot be repaired, the next level of maintenance is performed, at increased cost (called an unplanned maintenance event).

Results compressor: Current policy

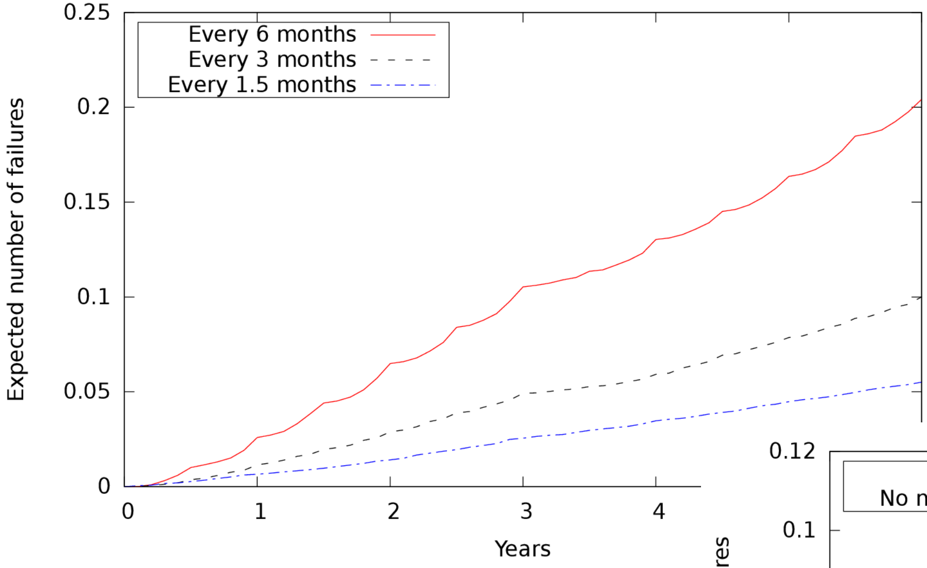


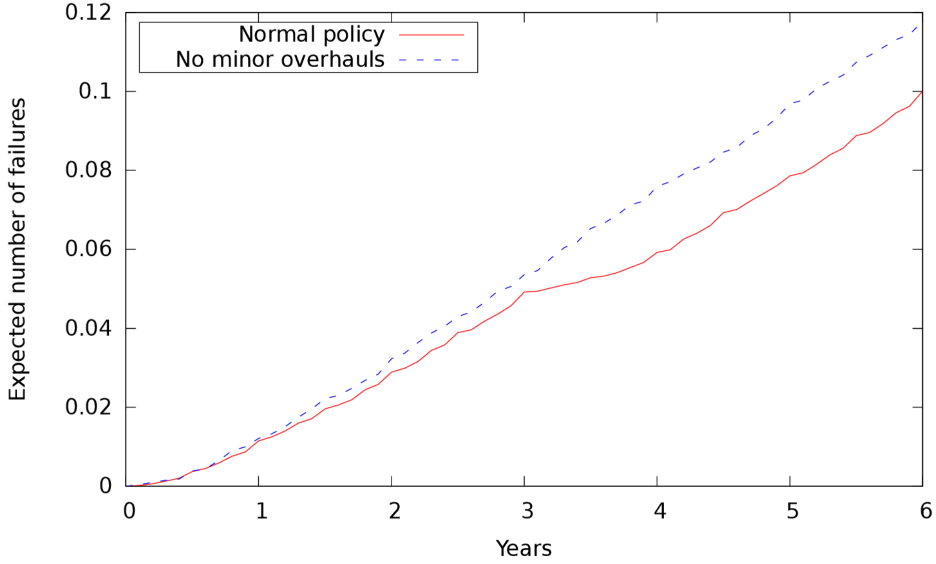


Result:

- Outcomes are fairly close to reality

Results compressor: Other policies





Results:

- Service period is important to maintain reliability.
- Minor overhaul does not have much effect.

Conclusions case studies

- Fault maintenance trees can model realistic maintenance strategies.
- We can analyze systems with maintenance and gain insight into cost-optimal performance.
- Our results are in agreement with reality.

Current work

- Replace phased degradation by continuous degradation (Completed but untested).
- Significant optimization of computation of fault trees with maintenance (completed, not yet public).

- Support for more advanced gates involving combinations of degraded BEs.
- Decent input language for fault trees with maintenance.
- Automatic optimization of complex maintenance policies.

Conclusions

- **Maintenance has large effect on RAMS**
  - should be analyzed in integral way
- **Fault maintenance trees**
  - Extend fault trees to include maintenance
- **DFTCalc**
  - extensible tool for reliability & availability analysis
  - compare different maintenance policies