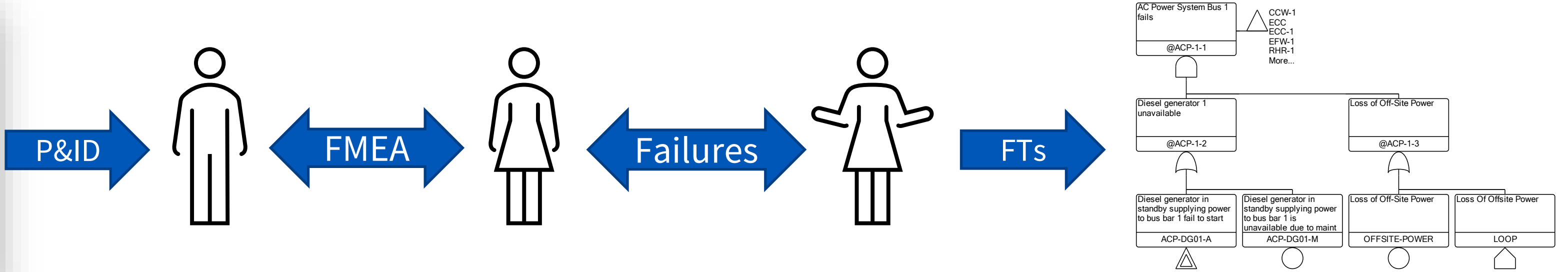
Control Logic Encoding using RS ModelBuilder

Pavel Krcal, Helena Troili, Ola Bäckström
28 June 2022

RISK SPECTRUM

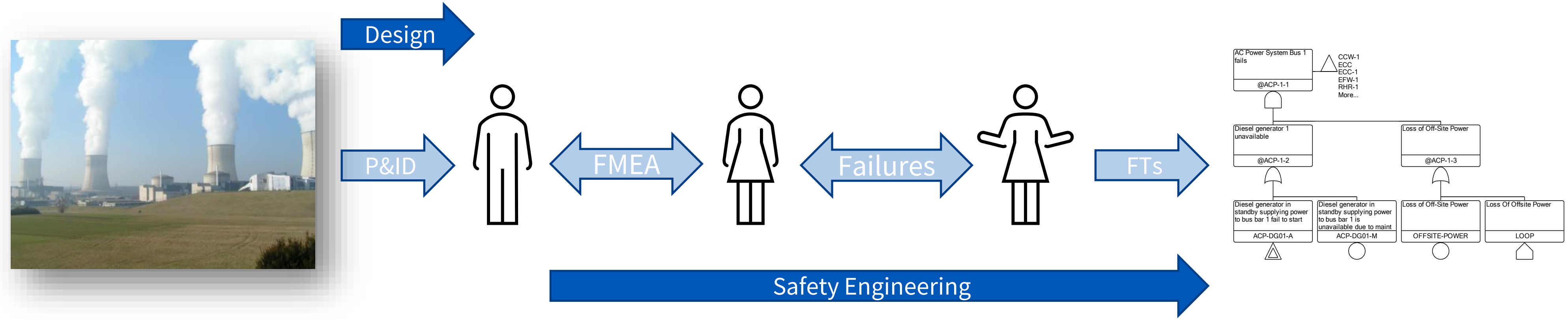
Model-Based Safety Assessment

Building fault trees for a PSA model



Model-Based Safety Assessment

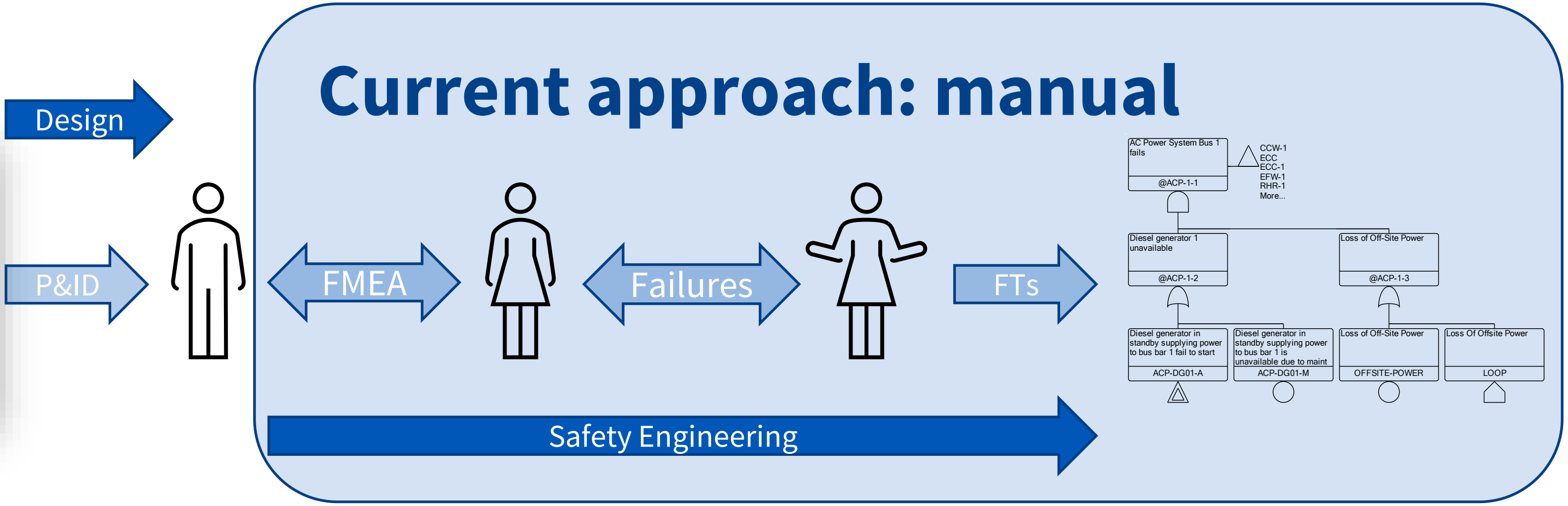
Building fault trees for a PSA model



RISK SPECTRUM

Model-Based Safety Assessment

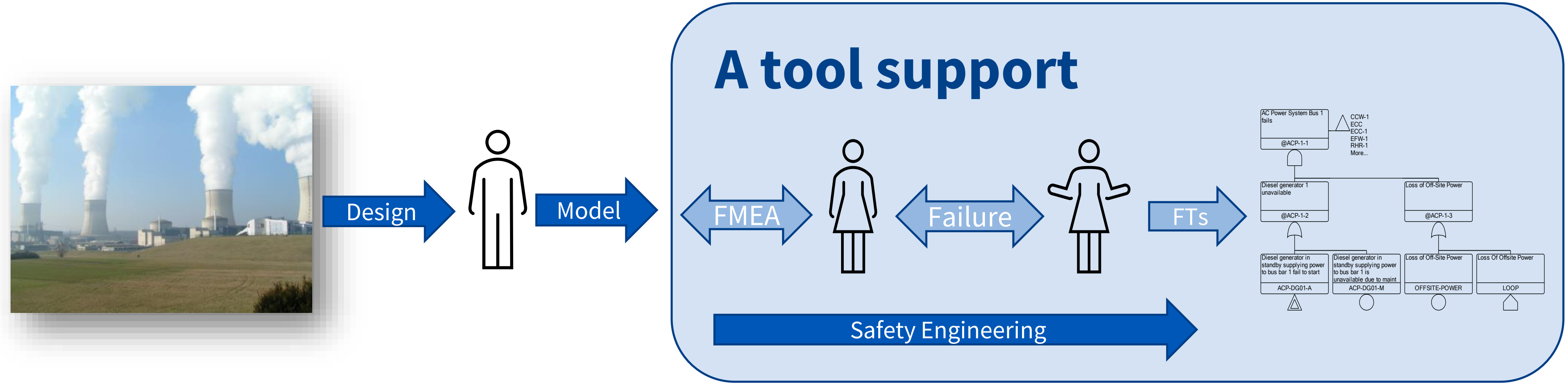
Building fault trees for a PSA model



RISK SPECTRUM

Model-Based Safety Assessment

Encapsulate and reuse expert knowledge



RISK SPECTRUM

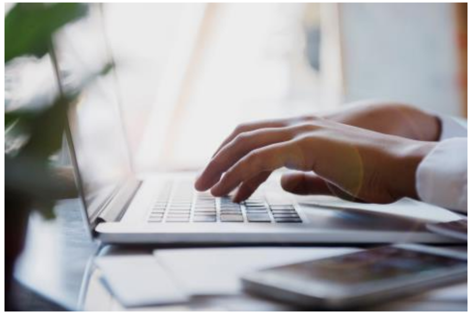
Model-Based Safety Assessment

Models close to the design

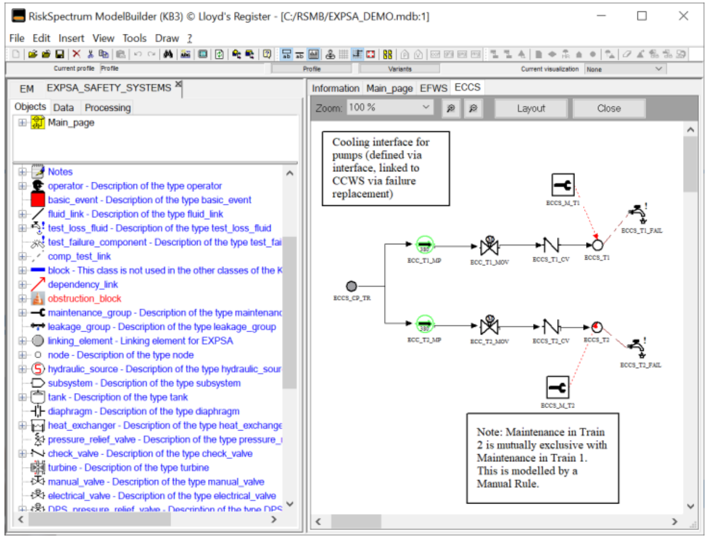
- Knowledge separation
- Laymen can build models
- Tailored safety analysis applications
- A closer link between the plant and the safety model (digital twin)

RiskSpectrum ModelBuilder (KB3)

System model



ModelBuilder – Platform



Solvers

Fault trees (RiskSpectrum PSA)

Monte Carlo simulation (YAMS)

Knowledge base – Figaro (a component library)

RISK SPECTRUM

Knowledge Base Definition (Fragments)

```
CLASS prod_system EXTENDS power_source ;
        INTERFACE
                rep KIND repair_crew CARDINAL 0 TO 1;
        CONSTANT
                fail_rate DOMAIN REAL DEFAULT 1e-3 ROLE DESIGN;
                rep_rate DOMAIN REAL DEFAULT 5E-2 ROLE DESIGN;
                fail_start_prob DOMAIN REAL DEFAULT 1E-2 ROLE DESIGN;
        ATTRIBUTE
                state DOMAIN 'producing' 'standby' 'under_repair' 'failed' 'required' DEFAULT 'producing';
                rank DOMAIN INTEGER DEFAULT 0;
```

```
prod4
STEP default_step
GROUP Simu_group
IF request AND state = 'standby'
THEN state <-- 'required', request <-- FALSE;
```

```
fail_in_op
GROUP Simu_group
IF state = 'producing'
MAY_OCCUR
FAULT fail DIST EXP(fail_rate)
INDUCING state <-- 'failed',
FOR_ALL r A rep DO rank <-- max_rank(r) + 1 ;
```

RISK SPECTRUM

Control Logic Modeling

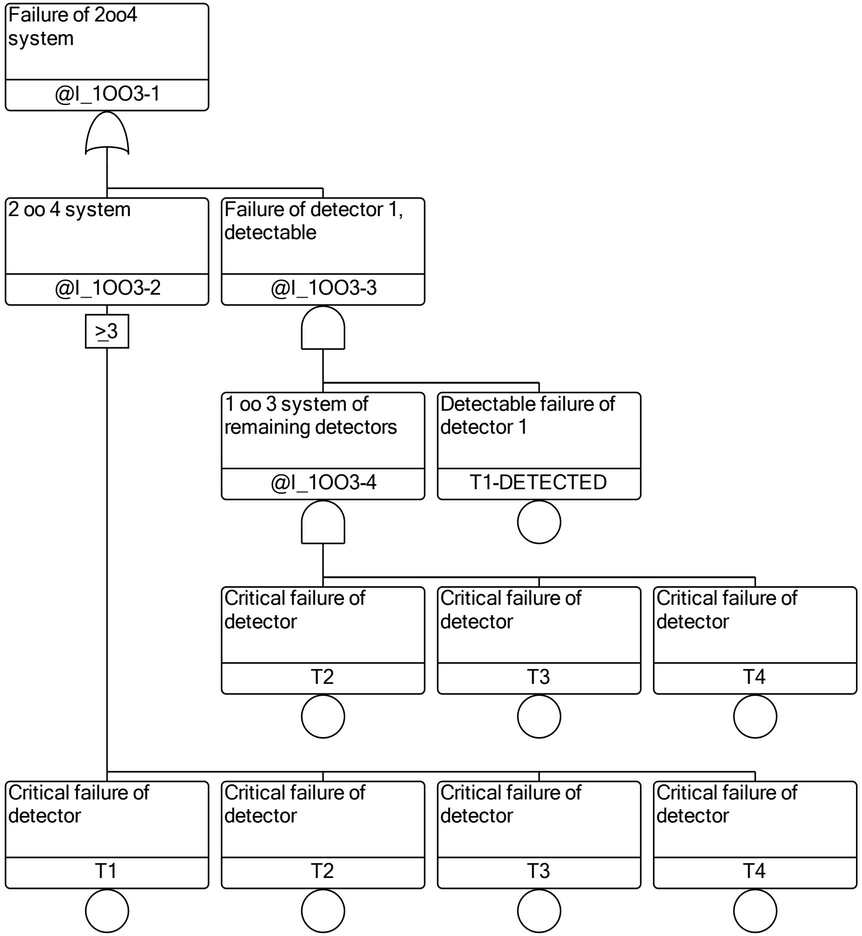
Examples from real-life models

- Instrumentation & Control: intelligent voting
- Spent Fuel Pool: water level and temperature control
- Heterogenous power production: resource scheduling

- We present the control logic and how it can be encoded in a knowledge base.

I&C: Intelligent Voting

- Detectable failures
- Modified voting logic



RISK SPECTRUM

I&C: Intelligent Voting

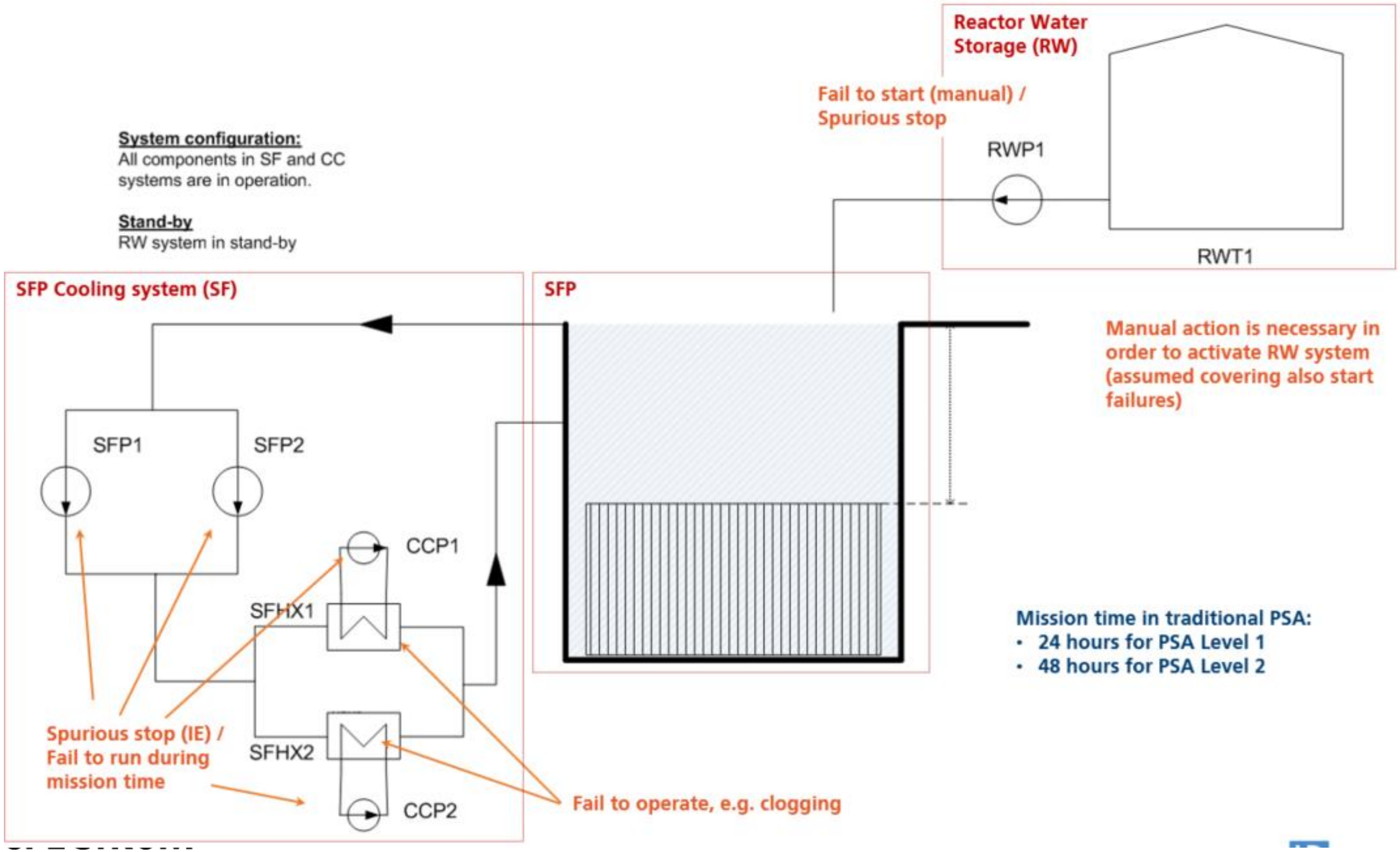
A component class for a voting system

- Modeling:
  - Drag and drop the component to the model scheme.
  - Drag and drop detector components.
  - Connect the voting component to detectors.
- Automatic fault tree generation will include intelligent voting.

```
(* All detectors functioning - original voting *)
IF NOT (THERE_EXISTS detector INCLUDED_IN linked_detectors
        SUCH_THAT detectable_failure(detector))
   AND NOT (THERE_EXISTS AT_LEAST detectors_required_all_OK(voting_system)
            detector INCLUDED_IN linked_detectors
            SUCH_THAT NOT critical_failure(detector))
THEN
    voting_system_loss ;

(* One detector defect - degraded voting defined in the voting system *)
IF (THERE_EXISTS detector INCLUDED_IN linked_detectors
        SUCH_THAT detectable_failure(detector) AND
            FOR_ALL other_detector INCLUDED_IN linked_detectors
            SUCH_THAT other_detector <> detector WE_HAVE
            NOT(detectable_failure(other_detector)) )
   AND NOT (THERE_EXISTS AT_LEAST detectors_required_one_failed(voting_system)
            detector INCLUDED_IN linked_detectors
            SUCH_THAT NOT critical_failure(detector))
THEN
    voting_system_loss ;
```

RISK SPECTRUM

Spent Fuel Pool: Water Level and Temperature Control



Spent Fuel Pool: Water Level and Temperature Control

Reactor Water System (RW)

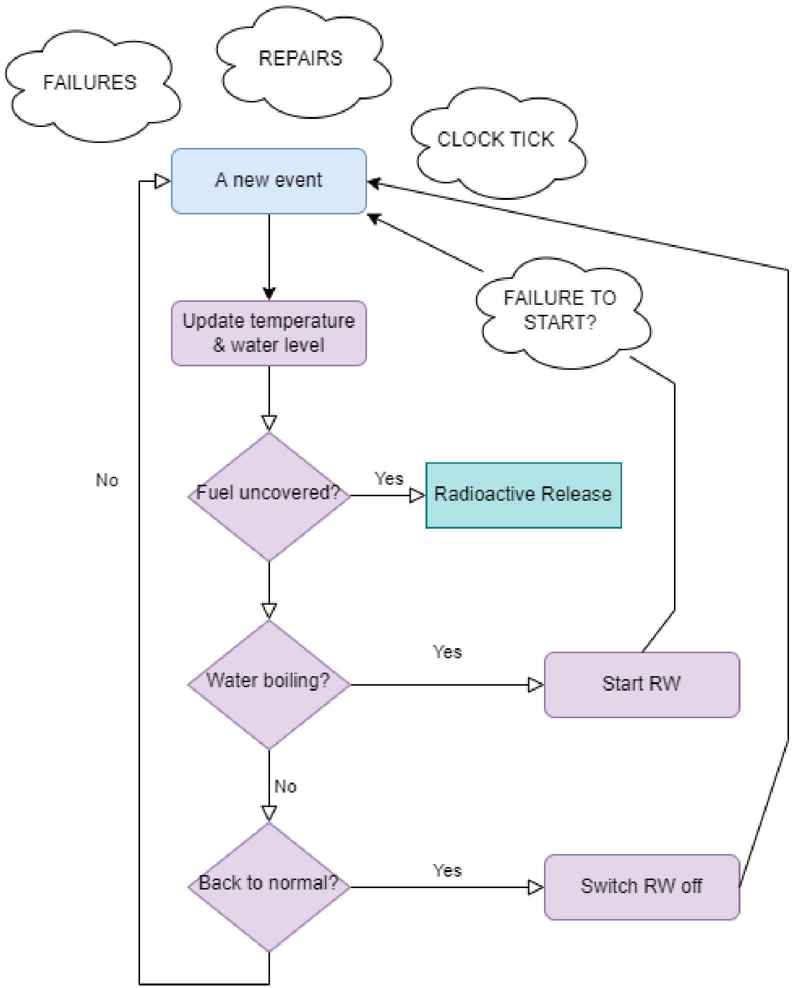
- A hybrid analysis – discrete failures together with continuous variables
- Solved by Monte Carlo simulations
- Time discretization: clock ticks
- Discrete events can occur also between ticks

RISK SPECTRUM

Spent Fuel Pool: Water Level and Temperature Control

Control System for RW

- Event-driven
  - Failures, repairs
  - Clock
- State evaluation
- Decisions
- Commands to other components (RW)



RISK SPECTRUM

Spent Fuel Pool: Water Level and Temperature Control

Plant state update

```
(* A simple linear approximation of the heat exchange *)
IF last_event_date <> CURRENT_DATE
THEN T <-- MIN( T + (heat(spent_fuel) - heat_removal(cooling)) *
           (CURRENT_DATE - last_event_date), 100 );

(* If the water is boiling since the last event then update the water level *)
IF water_boiling = TRUE AND T = 100 AND last_event_date <> CURRENT_DATE
THEN L <-- MAX( L - boil_off_factor * (CURRENT_DATE - last_event_date), 0 );

(* RW system is filling the pool with water if running *)
IF state(RW) = 'RUNNING' AND last_event_date <> CURRENT_DATE
THEN L <-- MIN( L + flow_rate(RW) * (CURRENT_DATE - last_event_date), L_MAX );

IF T = 100
THEN water_boiling <-- TRUE
ELSE water_boiling <-- FALSE;
```

Spent Fuel Pool: Water Level and Temperature Control

Control of RW

- The control component can check and set the state of RW.
- State: ‘required’
- Actual start is handled by RW with corresponding failure modes.

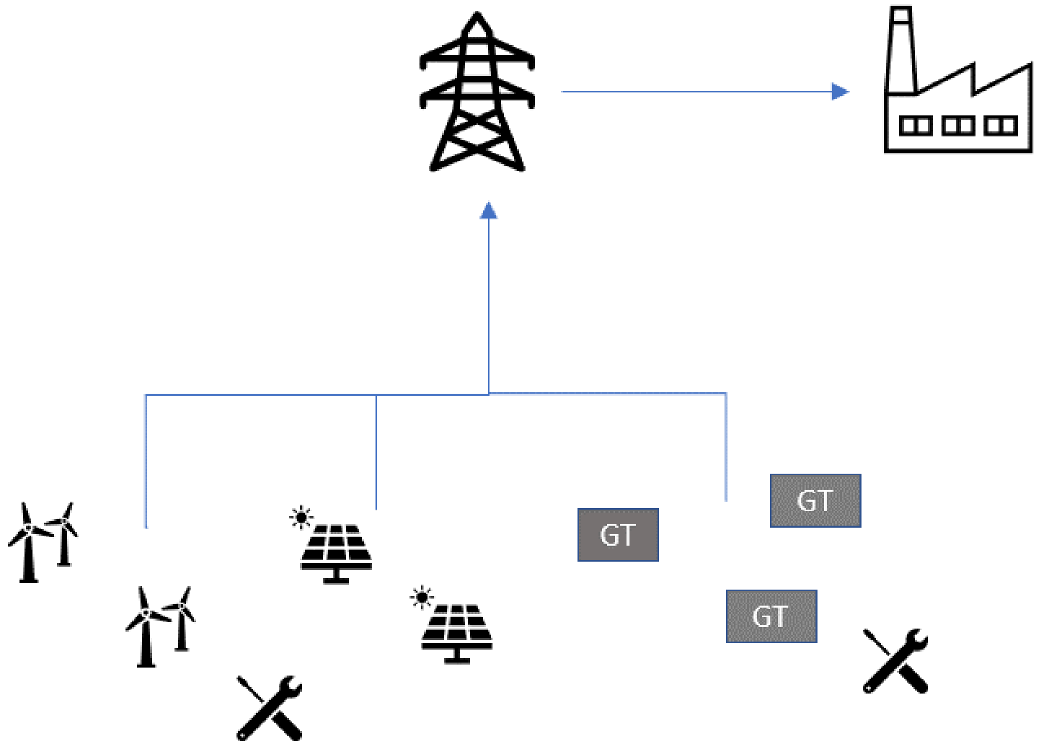
```
(* Send a start signal to the RW system if the
temperature reaches the critical level*)
IF T(pool) >= T_critical AND state(RW) = 'STAND_BY'
THEN state(RW) <-- 'REQUIRED';

(* Stop the RW system if the temperature is OK
and the pool is full *)
IF T(pool) <= T_normal AND L(pool) = L_MAX(pool)
                       AND state(RW) = 'RUNNING'
THEN state(RW) <-- 'STAND_BY';
```

RISK SPECTRUM

Heterogenous Power Production: Resource Scheduling

A heterogenous power plant

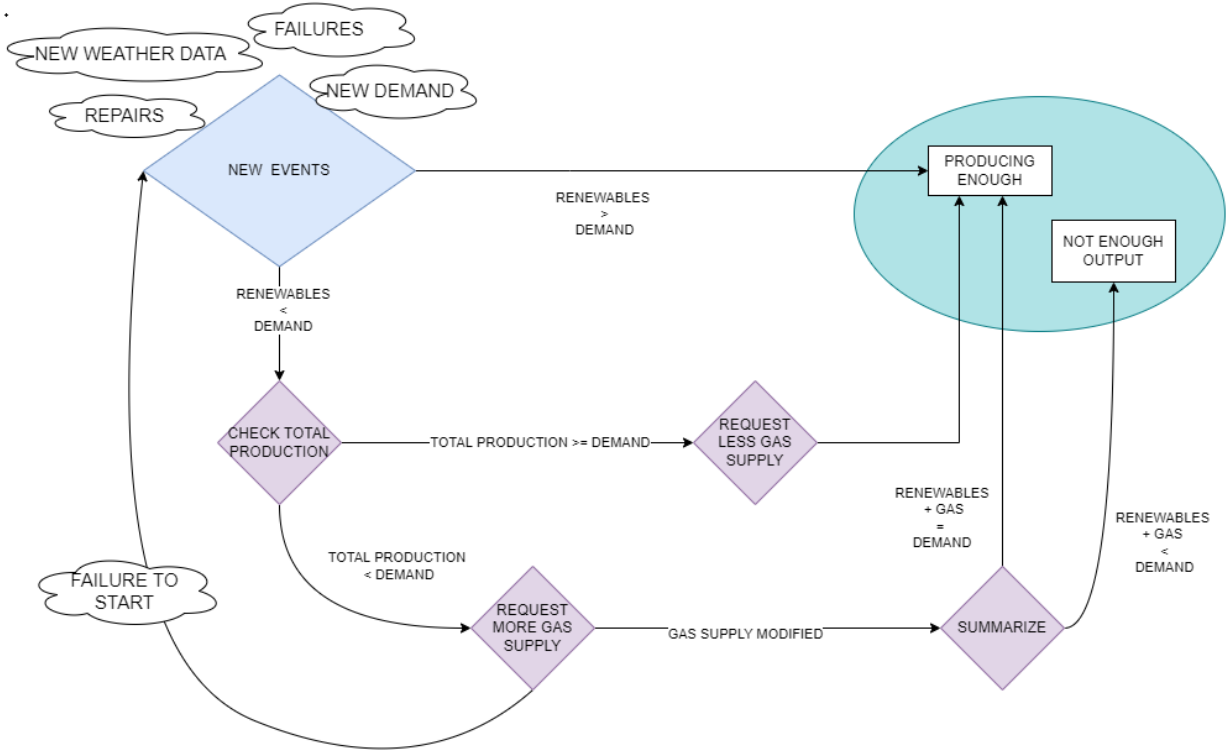
- Consumer: a variable demand
- Renewables: wind, photovoltaic
- Gas turbine back-up
- A repair policy



Heterogenous Power Production: Resource Scheduling

A control flow chart for production scheduling

- Event driven:
  - Changing demand
  - Changing production
  - Failures/repairs
- Gas turbines started only if needed
- Variable set-point



RISK SPECTRUM

Heterogenous Power Production: Resource Scheduling

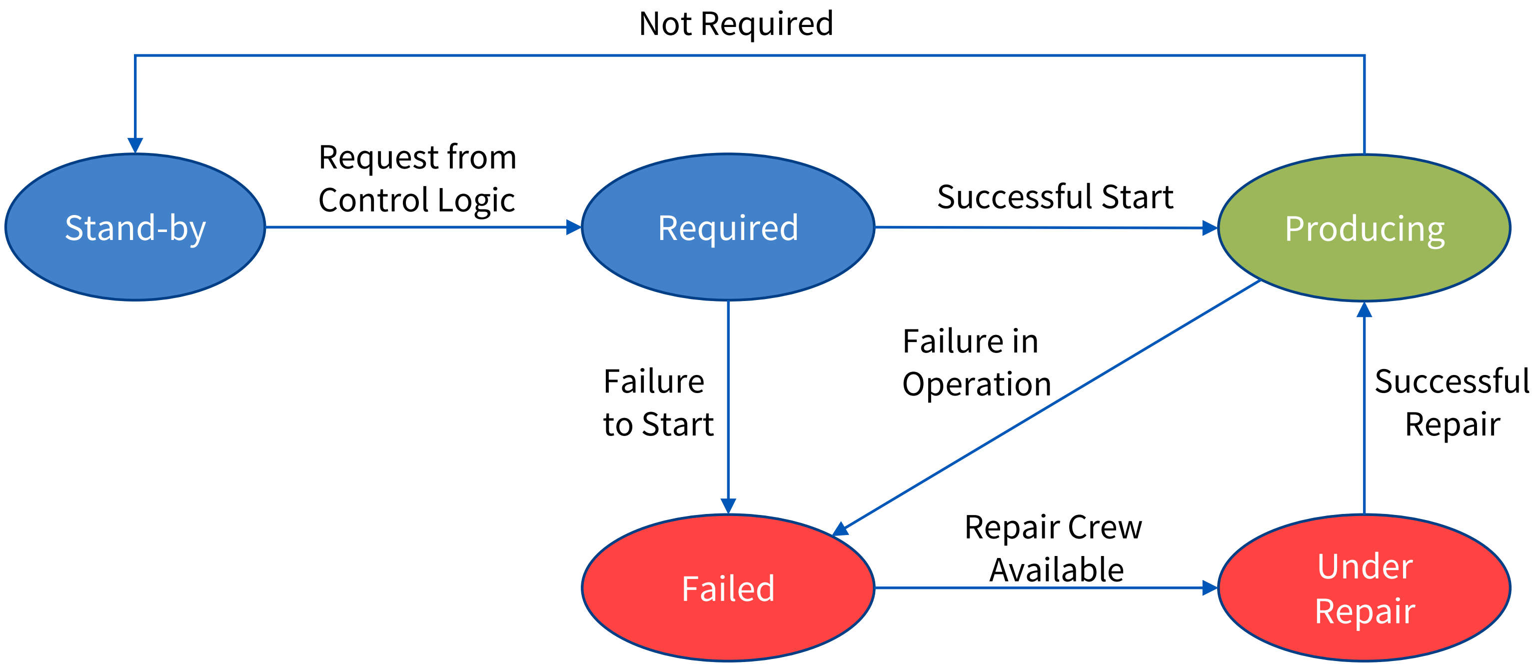
Control station: starting and stopping gas turbines

- Power of first order quantifiers:
  - One set of rules
  - Any number of gas turbines
- Communication:
  - Sending start requests
  - Setting the production point
  - Switching off (directly)

```
(* when the output drops too much then try to increase production
from running backups *)
station0
STEP request_backups GROUP Simu_group
GIVEN x A backups
IF current_output ...

(* when the output drops too much then start one backup *)
station1
STEP request_backups GROUP Simu_group
GIVEN x A backups
IF current_output + expected_new < demand AND state(x) = 'standby'
THEN state(x) <-- 'required', (* Raises an ocurrence for start-up *)
requested_output(x) <--MIN(maximal_output(x), (demand - (current_output + expected_new))),
expected_new <-- expected_new + requested_output(x);

(* when output from renewables is sufficient then switch off a backup*)
station2
STEP decrease_backups GROUP Simu_group
GIVEN x A backups
IF current_output - current_output(x) >= demand AND state(x) = 'producing'
THEN state(x) <-- 'standby',
current_output <-- current_output - current_output(x);
...
```

Heterogenous Power Production: Resource Scheduling

A state machine for each power producing component



RISK SPECTRUM

Heterogenous Power Production: Resource Scheduling

Gas turbine: production determined locally

- Depending on the state
  - A local property
- Requested output
  - Set by the control logic
- Maximal output
  - A local property

```
CLASS gas_turbine EXTENDS prod_system;
      ATTRIBUTE
      requested_output DOMAIN REAL DEFAULT 10;
      state DEFAULT 'standby';

      INTERACTION
      (* If a gas turbine produces it will provide requested but not more than its maximum *)
      gas1
      STEP default_step decrease_backups GROUP Simu_group
      IF state = 'producing'
      THEN current_output <-- MIN( requested_output, maximal_output );

      (* If a gas turbine is not producing, nothing is provided *)
      gas2
      STEP default_step decrease_backups GROUP Simu_group
      IF NOT(state = 'producing')
      THEN current_output <-- 0;
```

RISK SPECTRUM

Heterogenous Power Production: Resource Scheduling

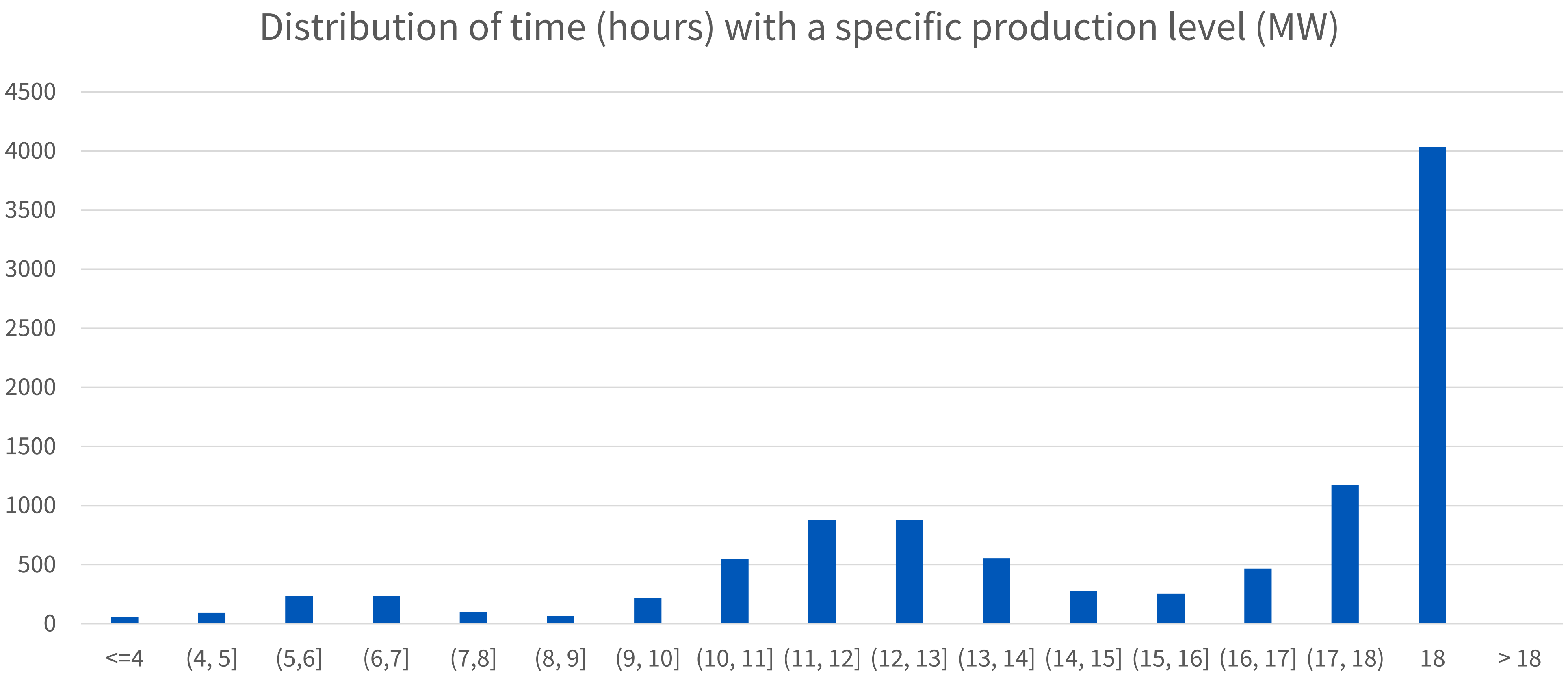
Failure modeling

- A standard failure mechanism in Figaro
  - A failure time distribution
  - Effects of a failure
- Different failure modes
- Repair modeling
  - FIFO per repair crew

```
OCCURRENCE
(*  Occurrences for prod systmes, i.e. renewables like wind, sun, and
 gas turbines. *)

fail_in_op
GROUP Simu_group
IF state = 'producing'
MAY_OCCUR
FAULT fail DIST EXP(fail_rate)
INDUCING state <-- 'failed',
FOR_ALL r A rep DO rank <-- max_rank(r) + 1 ;

fail_to_start
GROUP Simu_group
IF state = 'required'
MAY_OCCUR
FAULT fail DIST INS(fail_start_prob)
INDUCING state <-- 'failed',
FOR_ALL r A rep DO rank <-- max_rank(r) + 1
OR_ELSE
TRANSITION startup
INDUCING state <-- 'producing' ;
```

RISK SPECTRUM

Heterogenous Power Production: Simulation Results



RISK SPECTRUM

Conclusions

Control Logic Modeling in RiskSpectrum ModelBuilder

- Model-Based Safety Assessment
- Figaro - a generic and powerful modeling language
- Can incorporate various features for control logic modeling
  - Variable number of ‘related’ components
  - State machines
  - Flow charts
  - Time discretization for continuous variables
- Applicable to both fault tree generation and simulations

RISK SPECTRUM