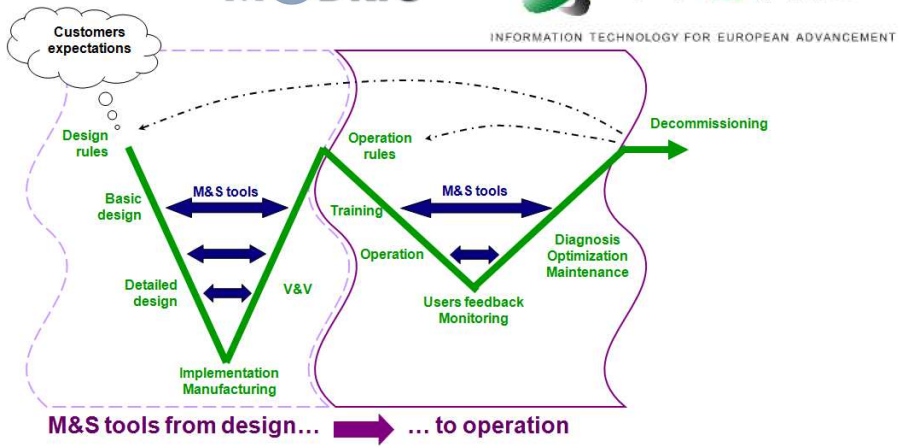


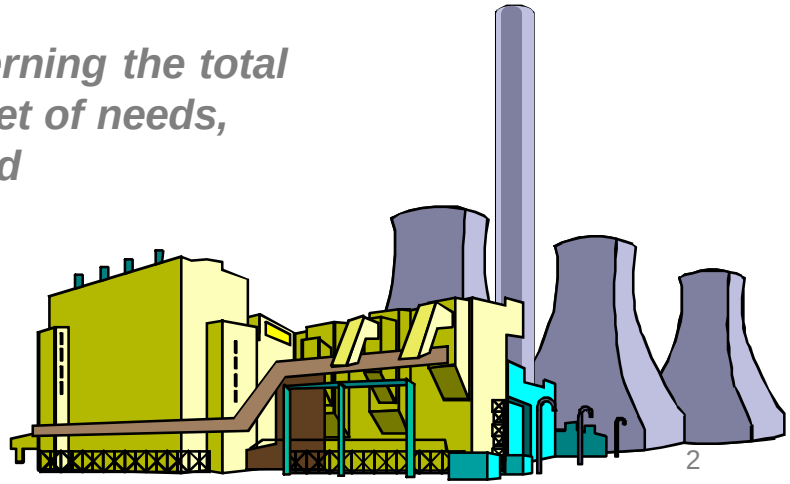

# The Benefits and Challenges of Massive Behavioural Simulation

in the Engineering of Complex and Critical Cyber-Physical & Human Systems

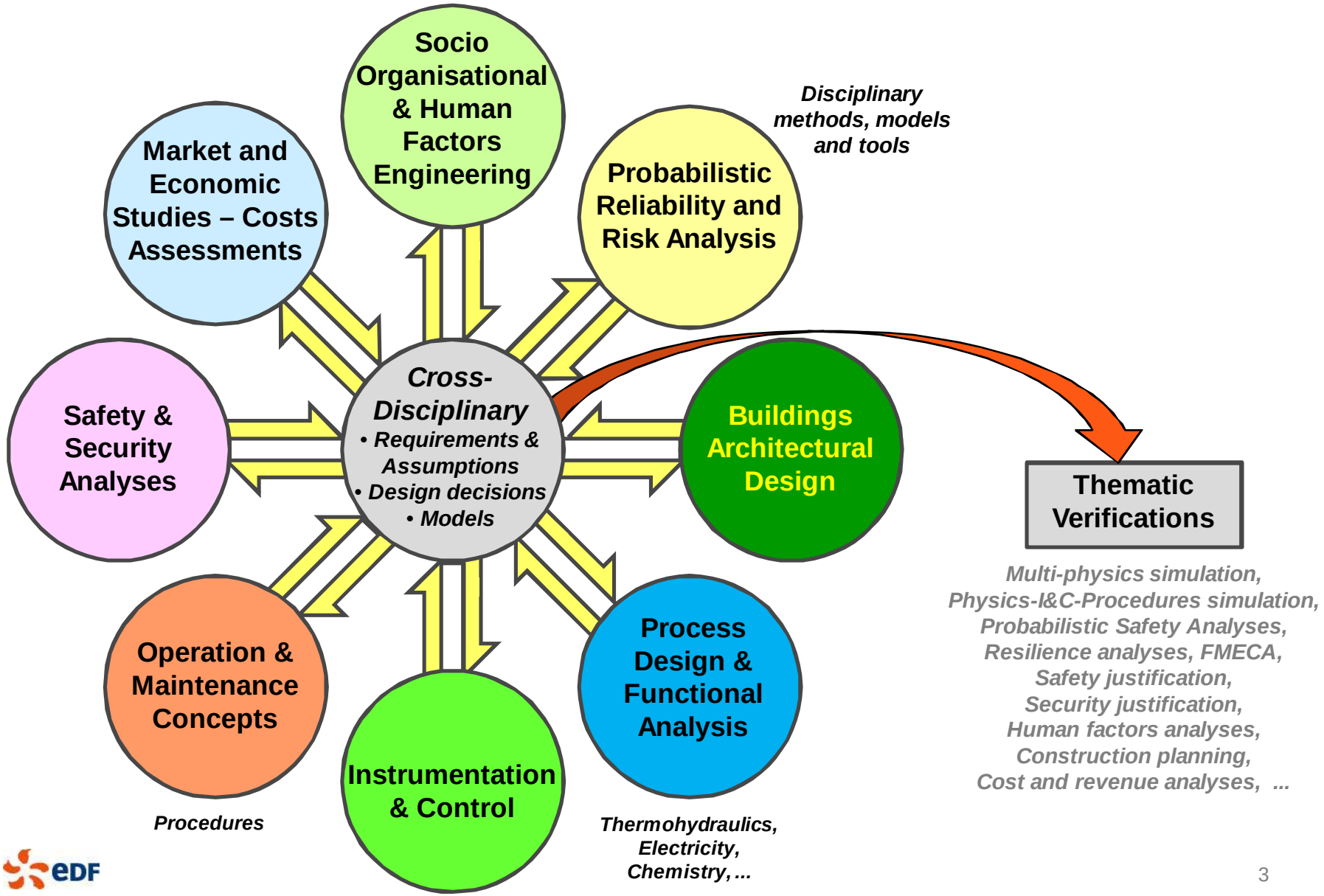
*N. Thuy - EDF R&D*
*October 14-16, 2015*
*Sim@SL*

# Cyber-Physical & Human Systems (CPHS)

- **A *cyber-physical system* integrates computation, networking and physical processes**
- **Many such systems also have an important human aspect**
  - For operation and maintenance
  - As a means to address unexpected situations, or situations too complex to be handled only via automated means
- **Large, complex systems often have important societal implications**
  - Transportation, energy or communication systems, public service infrastuctures, …
  - Need to ensure safety, dependability, performance, adaptability, …
  - High cost, high risk, high profile, high pressure to meet deadlines
- **They require an *interdisciplinary approach governing the total technical and managerial effort to transform a set of needs, expectations, and constraints into a solution and to support that solution throughout its life***
  - *INCOSE, Systems Engineering*



# Need for Multi-disciplinary, Multi-team Approaches



# System Behaviour

- **Functions**
  - What to do, and what not to do
- **Performance factors**
  - E.g., response time, accuracy
- **Quality of Service (QoS)**
  - E.g., fault tolerance, limits to failure probabilities
- **Behaviour in case of error or failure**

- **Taking into account assumptions made on system environment**

- **Possibly depending on operational states of the system and of its environment, and on operational goals**

- **Modelling & simulation as a support to behaviour-related activities**

# Key Questions

- **Are the specified behavioural requirements appropriate for all situations?**
  - Will they, in some cases, lead to unacceptable consequences?
  - Are the tradeoffs between conflicting stakeholders expectations, acceptable?

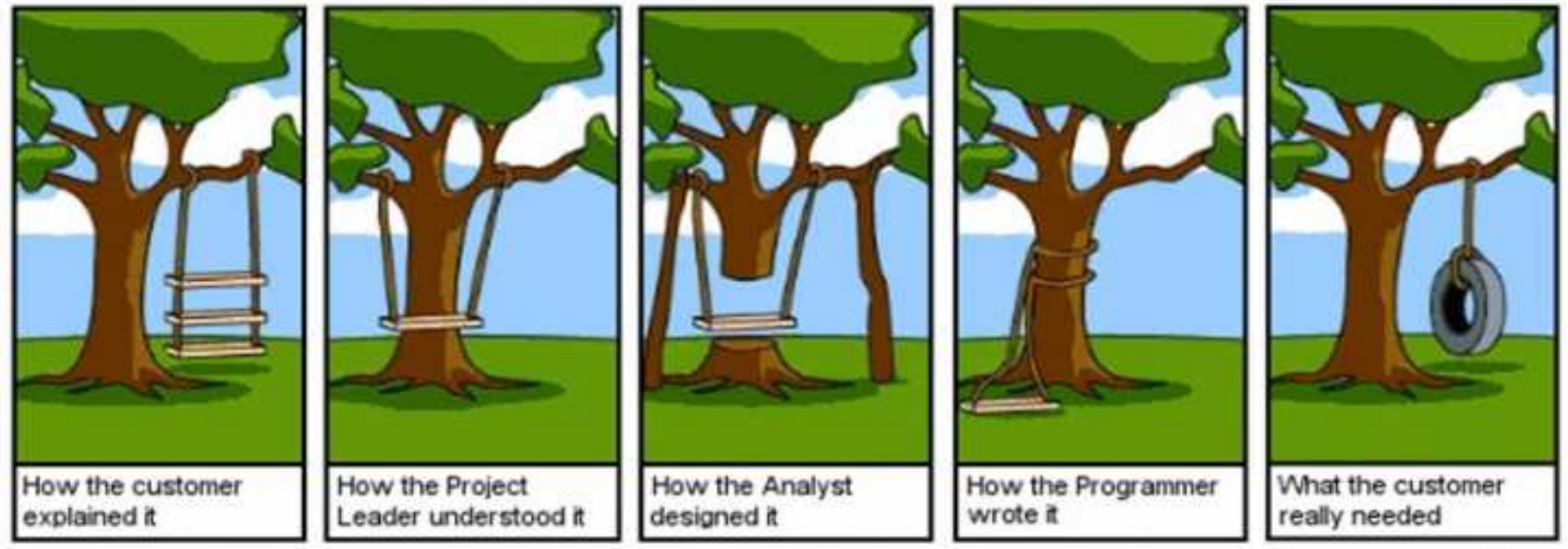
- **Does system design comply with requirements?**
  - Including in the presence of component failures or human mistakes, in all foreseen situations
  - Application of failure analysis methods such as FMECA, STPA, …
- **Do we have an optimal design solution?**
  - Need to evaluate (possibly many) design alternatives
- **Does system implementation comply with specification and requirements?**
  - Need to support testing, inspection, and possibly formal verification
- **During operation, what is the best course of action for operators to bring the system from its current state (e.g., an incident or accident state) to a desired state?**
  - Need to explore possible courses of action
- **Is there adequate support for training, operation, maintenance, modification?**

# Behavioural Requirements Specification is not Always Adequate

"Weaknesses in requirements are one of the most significant contributors to systems and software failing to meet the intended goals. A better analysis is needed to understand the software's interfaces with the rest of the system and discrepancies between the documented requirements for a correct functioning system."

[OECD COMPSIS Project Report – Nov 2011]

- **True for all industrial sectors, even for highly dependable systems**
  - Errors are revealed late during development, or worse, during operation



# This Could Lead to Unacceptable Consequences

**To prevent spurious deployment while airborne, hydraulic circuits of thrust reversers are required to be de-energised when wheels are not on the ground**

***Small airport, No local control tower, Snowing, Poor visibility (Real accident)***

# This Could Lead to Unacceptable Consequences

**Wheels on the ground → Thrust reversers are re-energised and in operation**

***Small airport, No local control tower, Snowing, Poor visibility (Real accident)***

# This Could Lead to Unacceptable Consequences

**Pilots see a snow plough on the airstrip → They disengage the thrust reversers and try to take-off**

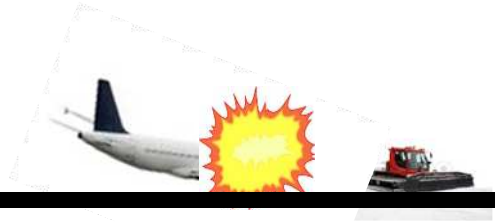
***Small airport, No local control tower, Snowing, Poor visibility (Real accident)***

# This Could Lead to Unacceptable Consequences

**Wheels no longer on the ground → Thrust reversers are de-energised. Reverser on one side is fully stowed, but not on the other side**

***Small airport, No local control tower, Snowing, Poor visibility (Real accident)***

# This Could Lead to Unacceptable Consequences



**Aerodynamic pressure reopens the thrust reverser → Airplane is thrown off balance, pilots do not have time to react**

***Small airport, No local control tower, Snowing, Poor visibility (Real accident)***

# Validation of Specified Behavioural Requirements

- **The system may have many internal states**
  - Covering installation and commissioning, normal operation (e.g., start-up, nominal operation, shut-down), maintenance and routine inspection & tests, subsystems states, deviations from nominal operation, failure states
- **The system environment may be composed of multiple entities (each with its own states) and may generate many possible events**
  - Including agressions (e.g., fire, flooding, seismic events) and malicious attacks
  - Need to explicitly state the assumptions made regarding system environment
- **The current operational goal may change in time, and may even change during a course of action**
  - Operators and maintenance personnel usually follow specified procedures
  - They also apply lessons learned during their training

# Validation of Specified Behavioural Requirements

- **Behavioural requirements are often different in different situations**
  - Many cases need to be considered before and during design
- **Design decisions will lead to additional requirements**
  - Including behavioural requirements
  - E.g., for the control, monitoring and maintenance of designed-in equipment

  ➔Fallacy of the cascading V-shaped lifecycle
- **Also, need to convince stakeholders that the tradeoffs made in requirements with respect to their expectations are acceptable to them**
  - Tradeoffs are necessary due to conflicting stakeholders expectations

# Verification of Design and of Implementation

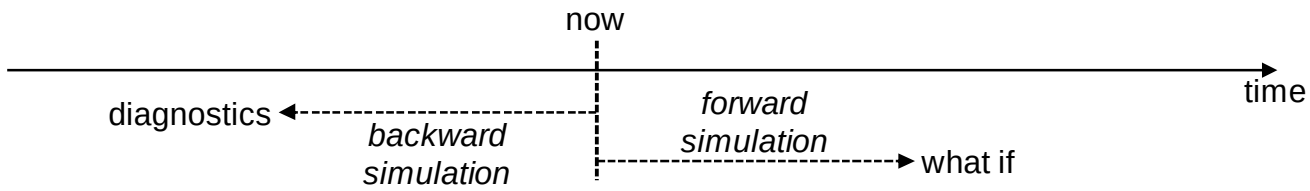
- **Design errors should be detected as early as possible in the engineering lifecycle**
- **Design verification against requirements at different stages**
  - Overall design: decomposition into subsystems, and allocation of system requirements
  - Detailed design: precise, deterministic description of subsystems behaviour
- **Closed-loop testing of implemented items**
  - Modelling and simulation of the environment of the tested item
  - Model-in-the-loop, Software-in-the-loop, Hardware-in-the-loop
- **Testing at various stages of the development process**
  - Unit testing
  - Integration testing
  - System validation testing
  - Factory acceptance testing, On-site acceptance testing

# Looking for an Optimal Design

- **Pressure from competition, budget constraints, planning constraints, multiple suppliers solutions, …**
- **Need to innovate and thus to evaluate multiple designs**
  - Preferably at overall design stages
  - Manually developed alternatives
  - Possible application of genetic approaches
- **Various assessment criteria**
  - Compliance with behavioural requirements
  - Cost and profit during operation (including maintenance)
  - Justification for safety and security
  - Socio-human factors
  - …

# Support for System Operation

- **Verify and validate operational procedures**
- **Training for operation**
  - Self-training (in the absence of a coach): cases generated on the fly (based on broadly defined generic scenarios), with automatic assessment of
    - Correct implementation of stated operational goals
    - Satisfaction of required system invariants
    - Compliance with specified operational procedures
  - Coverage of training courses
- **Diagnostics: help determine the cause of a deviation from nominal behaviour**
  - Possibly with the application of investigation procedures
- **What-If analysis: helps determine an appropriate course of action in order to bring the system into a certain state while avoiding other states**
  - Multiple assessment criteria
  - Faster-than-real-time simulation



# Failure Analysis

- **Example: FMECA (Failure Modes Effect and Criticality Analysis)**
  - Make sure that the consequences of component failures are acceptable
- **Principle**
  - For each component in the design, for each possible failure mode of that component,
  - For various situations (state of the system, state of its environment, operator objective, timing of the failure),
  - Evaluate system behaviour, and check that in each case, consequences are acceptable
    - Need to take into account fault propagation
- **Possible extensions**
  - Consideration of multiple component failures, as determined by probabilistic analyses
  - Consideration of common-cause failures (due to errors in design, construction, operation, maintenance)
- **Currently, manual analysis**
  - (Tens of) thousands cases to consider, multi-thousand page reports, high risk of error, difficult to keep up-to-date, done late in the design process just for confirmation, difficult to use, extremely time consuming, extremely expensive (multi m€)
  - Doable only for highly critical systems

# For All Activities

- **Behavioural modelling and simulation can be of great help, but most often, a very large number of cases need to be considered**
  - Typically, several tens of thousands for one activity, and up to a few millions
- **Need to simulate human actions**
  - Based on specified procedures
- **Need also to perform simulation according to various sufficiency criteria**
  - Functional and / or structural coverage criteria
  - Boundary and out-of-boundary testing
  - Fault-injection
  - Statistical testing, Monte Carlo testing

➔**Massive simulation**

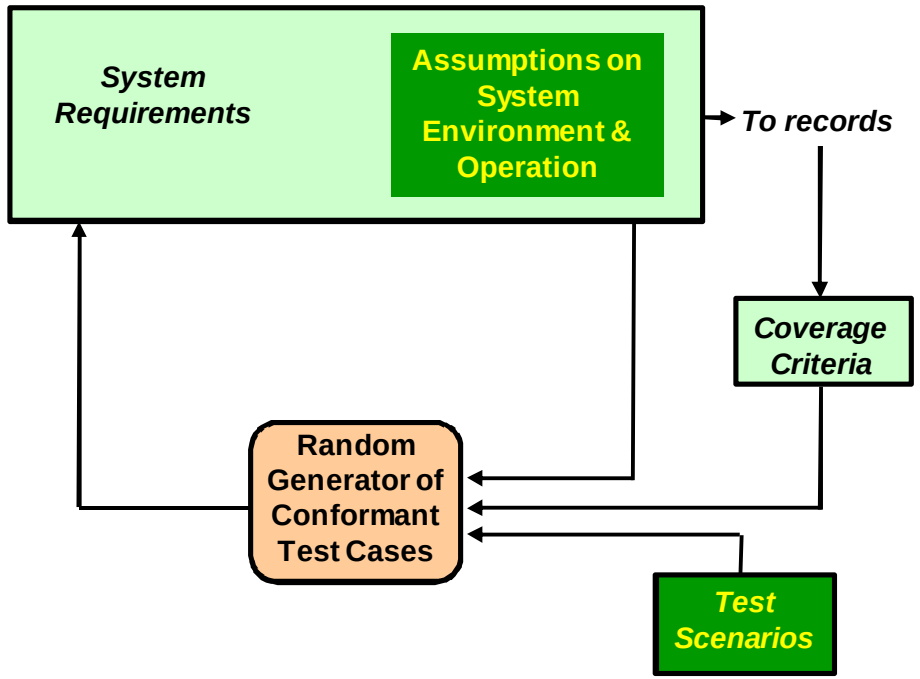
  - Simulation runs are independent from one another and can all be executed in parallel

# The Challenges of Massive Simulation

- **Generation of a large number of test cases**
- **Results assessment for an equally large number of test cases**
- **Minimising the need for different models, languages and tools when addressing different activities**
  - Modelling thriftiness
    - E.g., FMECA with the same models as for overal design evaluation
  - Models inter-operability
  - Consistency with real design / system
- **Ascertaining the correctness of models and simulation results**
- **Also (not addressed here)**
  - Perenniality of models and simulation infrastructure
- **Assumption: computing power is not an issue**
  - High-Performance Computers (HPS) have typically ten or hundred thousand nodes
  - One million runs could be performed in a matter of minutes

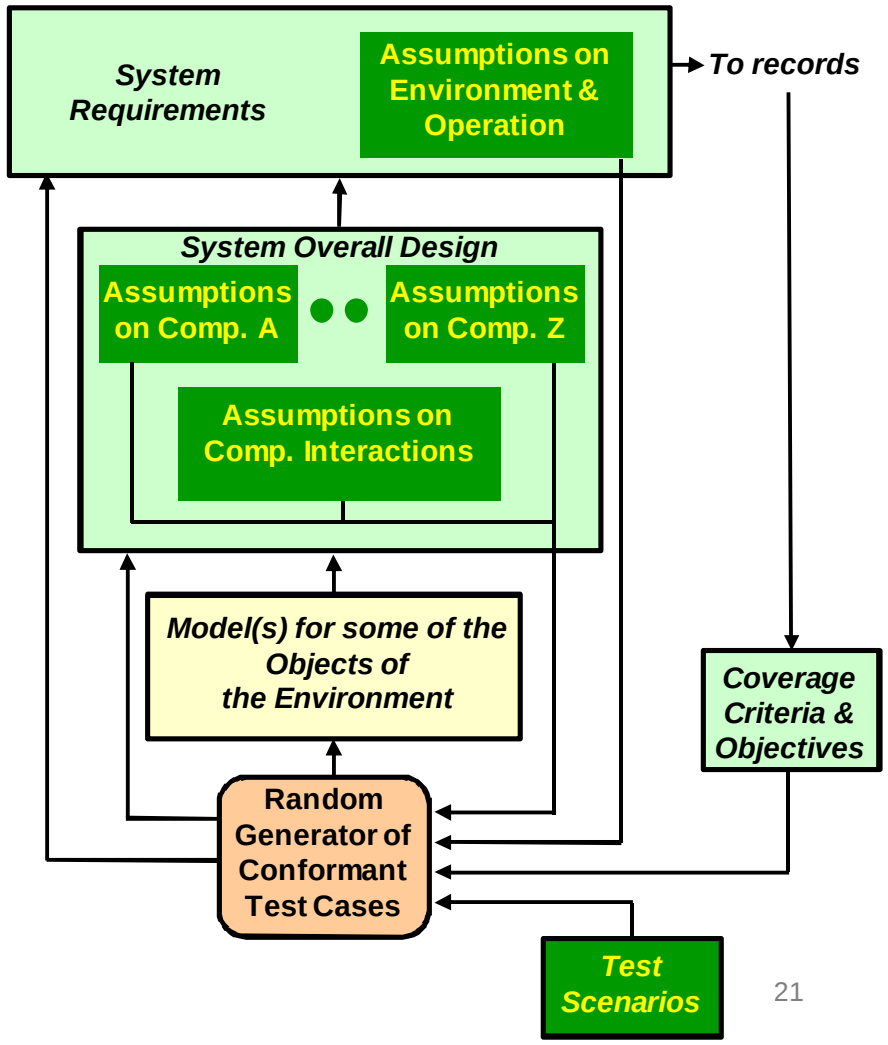
# Generation of Large Numbers of Test Cases

- **Use of an automatic test case generator**
- **But test cases should satisfy certain constraints**
  - Assumptions made regarding system initial conditions, system environment and operators actions
  - User-specified generic test scenarios
  - Satisfaction of specified test coverage objectives
- **Generic test scenarios can be specified in the form of additional assumptions**
- **Such generators are now commercially available**
  - Example: StimuLus (from ArgoSim)



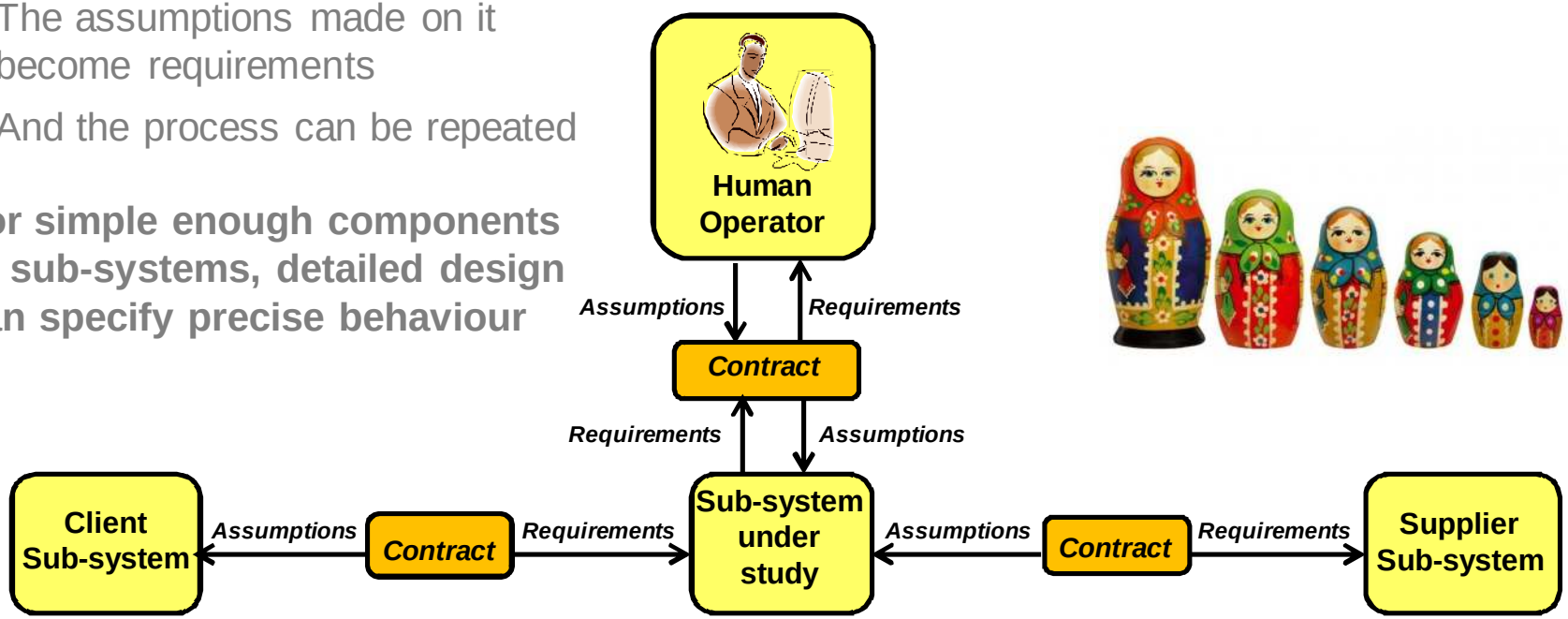
# Analysis of Large Numbers of Test Results

- **Approach: formal modelling of the behavioural requirements of a system (viewed as a black box)**
- **Can be used to verify that requirements specification is appropriate**
  - The approach also models the assumptions made regarding the environment and the operation of the system
- **Can also be used to automatically check that simulation results for other systems engineering purposes are OK (or not)**
  - Including behavioural requirements specification for subsystems



# Support for Gradual and Modular Design Approaches

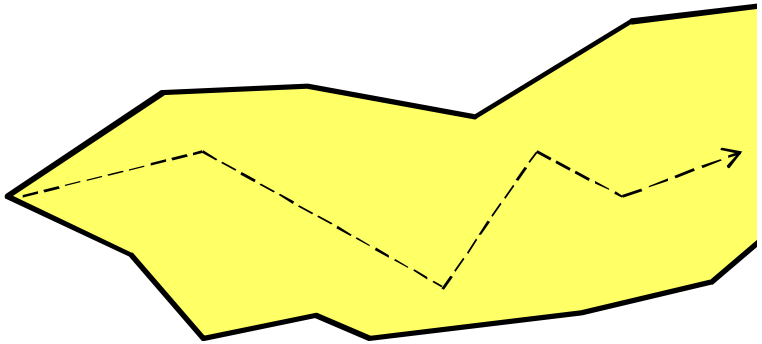
- **The overall design of a system identifies its components**
  - … and specifies the assumptions made regarding their interactions and behaviours
  - Allocation of system requirements to components
- **Some of these components may be considered as (sub-)systems**
  - Once the overall design is verified, a sub-system can be viewed as a system of its own
  - The assumptions made on it become requirements
  - And the process can be repeated
- **For simple enough components or sub-systems, detailed design can specify precise behaviour**



# Modelling Thriftiness

- **Two main categories of behavioural models**
  - An imperative model completely specifies the behaviour of a system
    - Given initial and boundary conditions, only one behaviour is possible
  - A constraint-based model just specifies the constraints to be satisfied
    - Usually, given initial and boundary conditions, many possible behaviours are allowed
    - Techniques exist to detect over-constrained models (i.e., models that have no solution)
- **Imperative models are used to describe detailed designs and solutions**
  - Examples: MODELICA models, functional diagrams
- **Constraint-based models are preferable for expressing high-level behavioural requirements, to describe high-level overall designs or to specify generic scenarios**
  - Example: FORM-L (FOrmal Requirements Modelling Language) models

Imperative behavioural model

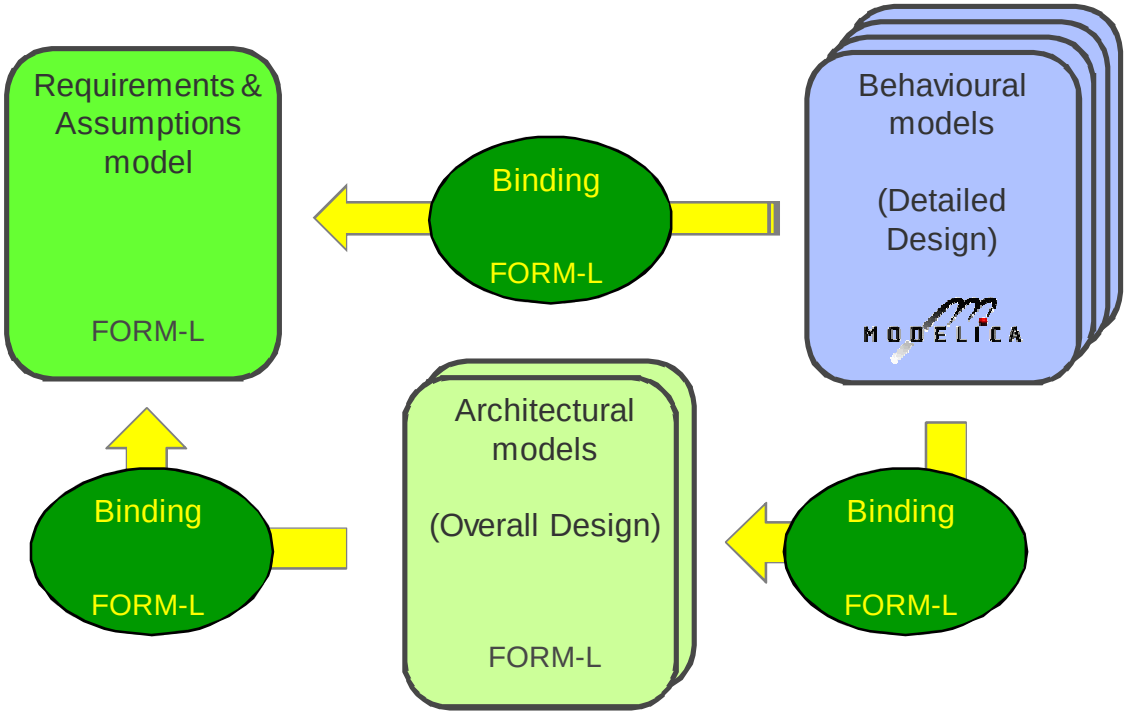


Constraint-based behavioural model

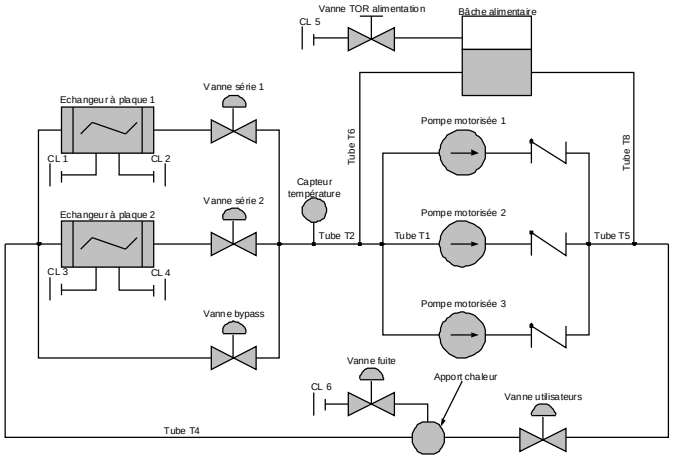
# Models Composability and Inter-Operability

- **Different types of models are developed by teams from different disciplines and with specific constraints**
- **Need to perform co-simulation without modifying any of these models**
  - Models may represent different parts of the system and its environment
  - A part may be represented by multiple models, at different development stages

➔**'Bindings' between models**



# MODELICA




- **Modelica® is a non-proprietary, object-oriented, equation-based language to conveniently model complex physical systems**
  - Containing, e.g., mechanical, electrical, electronic, hydraulic, thermal, control, electric power or process-oriented components
- **It is supported by several toolsets, some commercial (like Dymola), others open-source (like OpenModelica)**
- **Models have an internal textual representation, but most are developed graphically**
- **It is widely used in many industrial sectors**
- **Reusable components libraries facilitate models development**

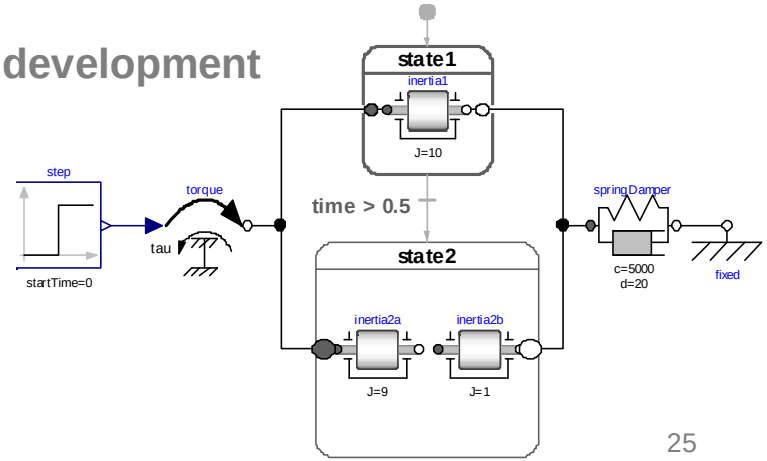
Normal shaft

$$J \cdot \frac{d\omega}{dt} = \tau_a + \tau_b$$

Broken shaft

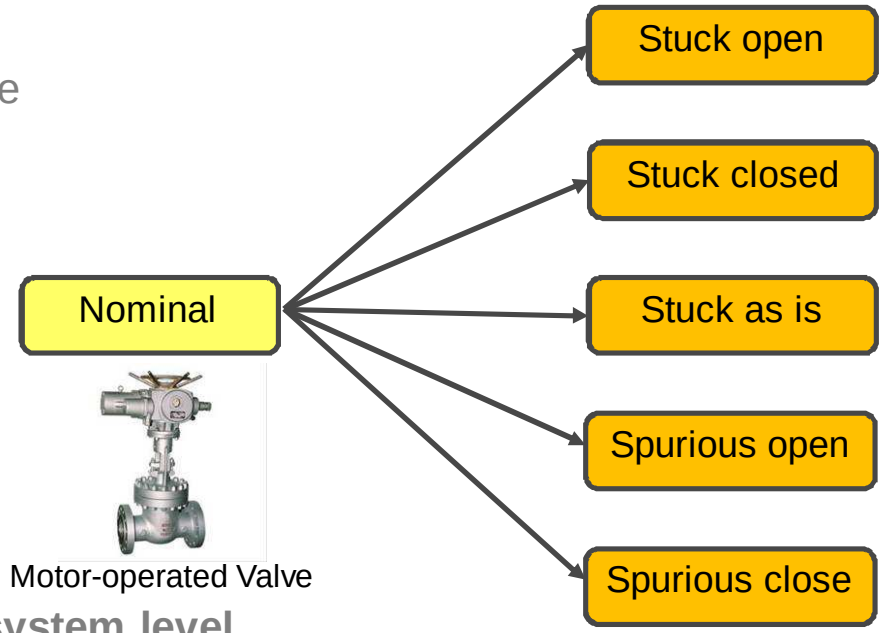
$$J_a \cdot \frac{d\omega_a}{dt} = \tau_a$$

$$J_b \cdot \frac{d\omega_b}{dt} = \tau_b$$



# Multi-Mode Modelling

- **MODRIO is extending MODELICA to better support systems engineering**
- **One such extension is multi-mode modelling**
  - The nominal, downgraded or failure modes of a component can be represented by a stochastic state automaton
  - Each mode can be represented by a separate model
  - When a component fails and in which mode is determined by the test case
  - The simulation infrastructure automatically switches between models
- **Such modelling could be done once and for all in components libraries**

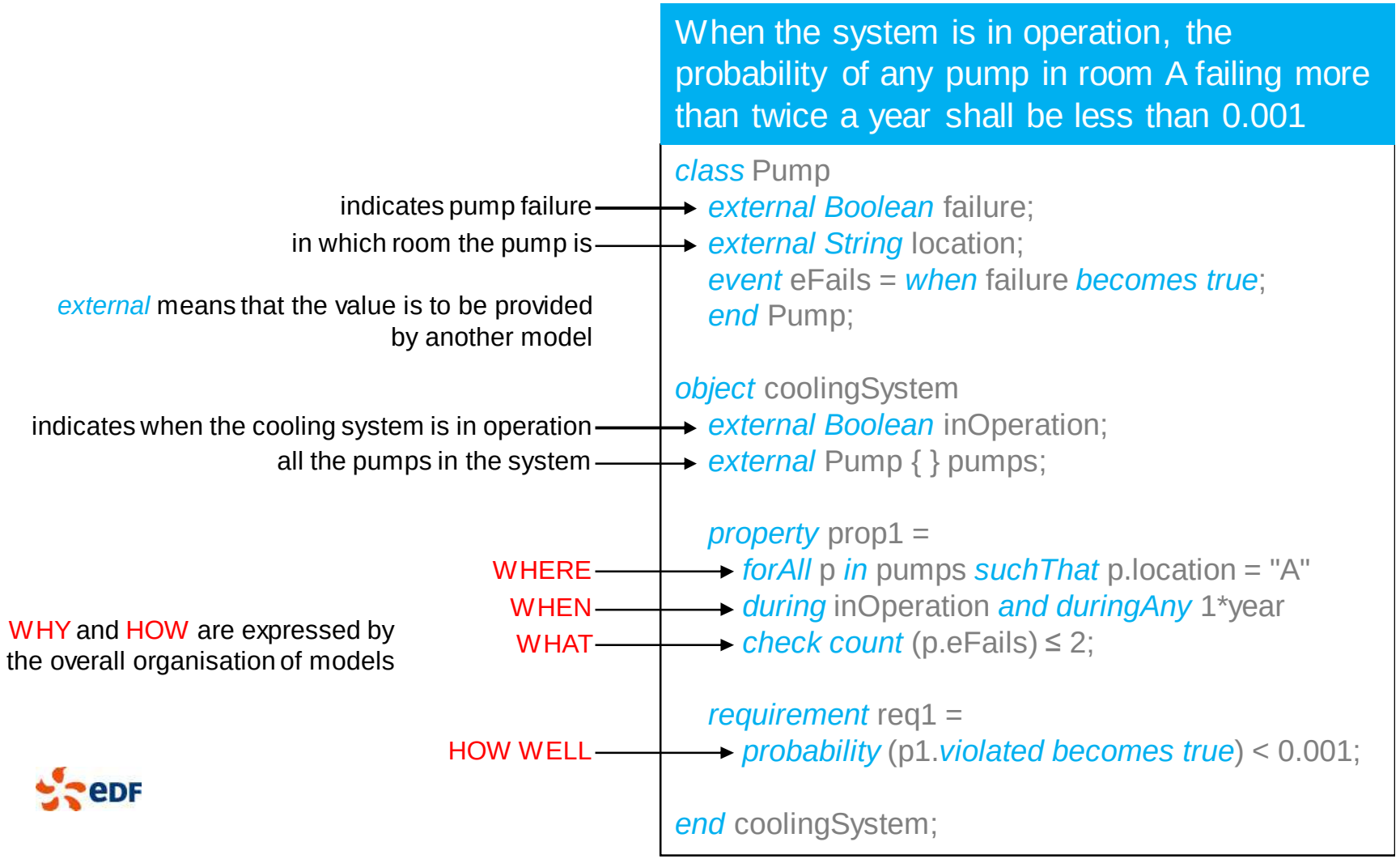


Motor-operated Valve

- **Multi-mode modelling can also be used at system level**
  - E.g., to represent major state transition between a normal, mono-phasic state and an incidental, di-phasic state

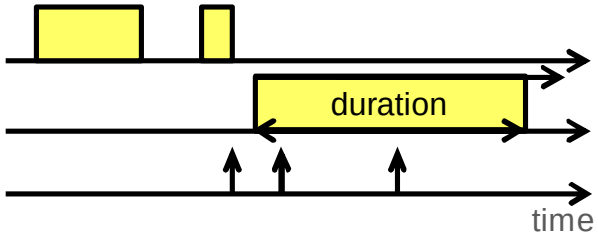
# FORM-L

- **Also developed in the framework of MODRIO, to express**
  - Behavioural requirements and assumptions
  - Overall design decisions, such as allocation of requirements to components



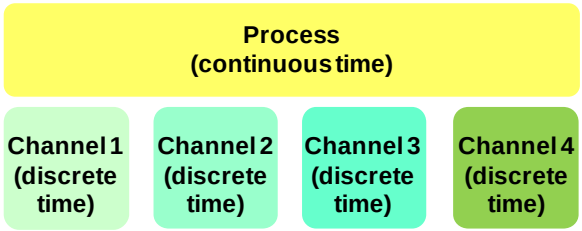
# WHEN

- **Time locators**
  - During time periods
  - During "sliding time windows“ of fixed duration
  - At particular instants



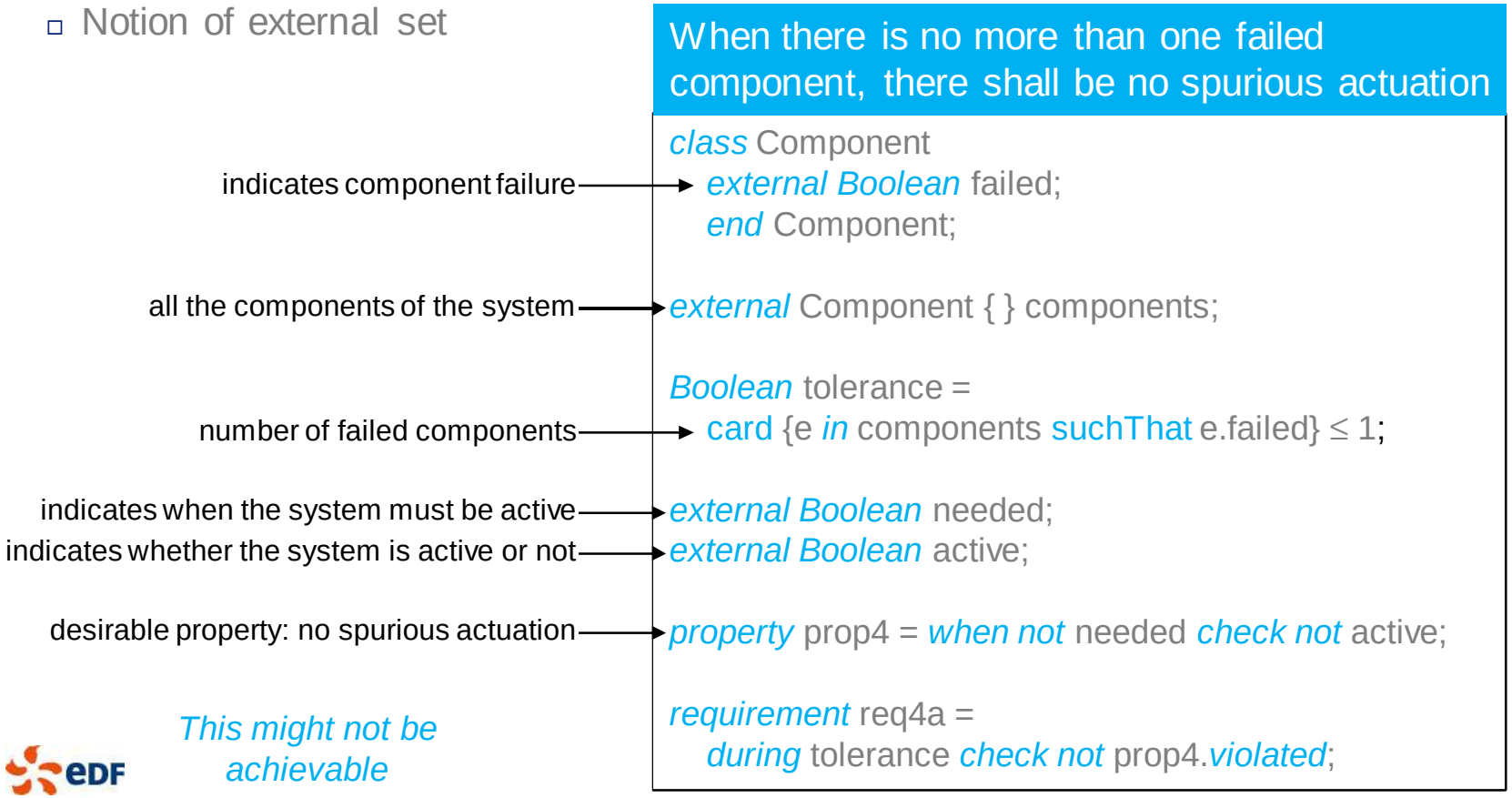

- **Finite state automata and statecharts**

- **Time domains**
  - A single continuous time domain
  - Optional, possibly multiple, discrete time domains
    - Each such domain has its own clock
    - Usually periodic, but could be quasi-periodic, intermittently periodic, multi-periodic or not periodic at all
  - Need to model time domain interfaces
    - How an object in one time domain can perceive an event in another domain



# WHERE

- **Specification of which parts or objects of the system are concerned with a property**
- **Sometimes, need to refer to objects not known individually at requirements specification time**
  - Notion of external set

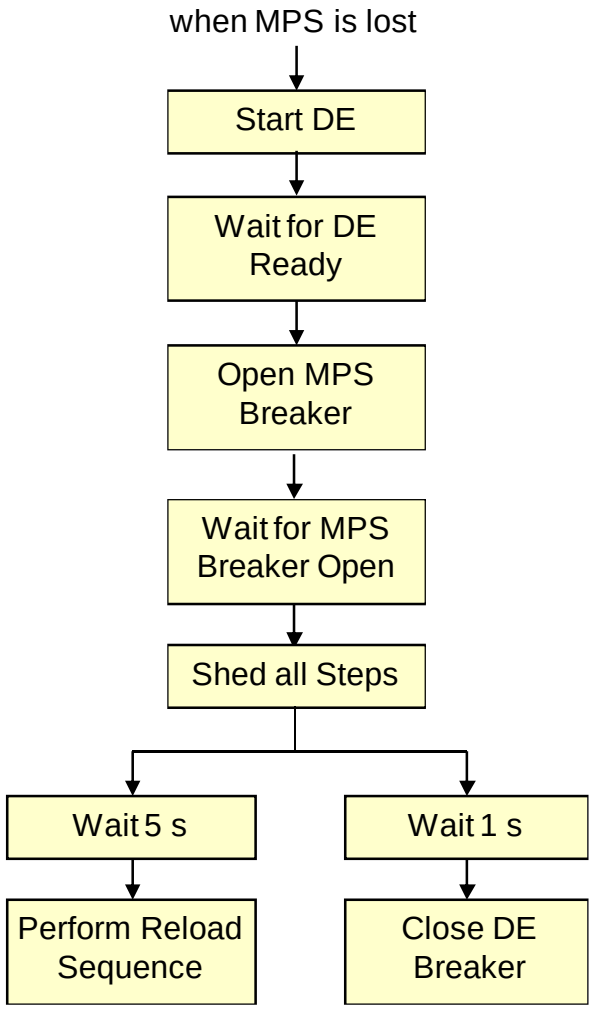



*This might not be achievable*

# WHAT

- **Specification of constraints to be satisfied or imperative actions to be performed**
- **Constraints**
  - Boolean conditions
  - Duration of Boolean conditions
  - Events occurrences
- **Elementary actions**
  - Checking of a constraint
  - Assignment to a variable
  - Raising of an event
  - Control of the simulation
- **Composite actions**
  - Sequential actions
  - Parallel actions
  - Iterative actions
  - Conditional actions

# WHAT - Example




## Imperative actions (in a discrete time domain)

```
when eMPSLoss then sequence
   raise eStartDe;
   wait eDeReady then raise eMpsBrkOpenCmd;
   wait eMpsBrkOpen then raise eShedAllCmd;
   simultaneous
      wait 5*s then raise eReload:
      wait 1*s then raise eDeBrkCloseCmd;
      end;
   end;
```

## Constraints specification

```
when eMPSLoss then sequence
   within 100*ms check eStartDe;
   within 5*s check eDeReady;
   within 100*ms check eMpsBrkOpenCmd;
   within 1*s check eMpsBrkOpen;
   within 100*ms check eShedAllCmd;
   simultaneous
      wait 5*s then within 100*ms check eReload:
      wait 1*s then within 100*ms check eDeBrkCloseCmd;
      end;
   end;
```

# Models Correctness – 1/2

- **Understandability and verifiability by application domain experts**
  - Modelling language with appropriate concepts, and clear textual / graphical syntax
    - With support for natural language or diagramatic "boiler plates“
  - Availability of suitable libraries (object templates, functions) and modelling patterns
  - Simulation with adequate human interface to facilitate models understanding and verification
- **Modular organisation of complex models**
  - Separate verification of model modules
  - Reuse of verified model modules
  - Generic modelling patterns
  - Modularity also facilitates models understanding and maintenance
- **Static models analysis**
  - To detect intrinsic flaws (e.g., that a requirements model is too constrained and has no solution, inconsistencies in the information flow, instrinsic incompleteness, ...)
  - To identify aspects that need particular attention (complexity)

# Models Correctness – 2/2

- **"Debugging" means to help understand unexpected simulation results**
- **Taxonomy of errors that could lead to incorrect models or simulation results**
  - Transcription errors
  - Forgetting to address certain situations
  - Ignorance of situations that need to be addressed
  - Inadequate treatment of certain situations
  - Misunderstanding of certain modelling language / tool features
  - Tools errors
  - …
- **Application of a rigorous modelling methodology addressing each type of error**
- **Also, explicit justification in a structured assessment framework**
  - E.g., based on a Claim-Argument-Evidence (CAE) approach

# Conclusion

- **Massive behavioural simulation can be of great help**
  - And could ‘democratise’ expensive but useful techniques such as FMECA or STPA
- **Work is on-going within the MODRIO project so that MODELICA infrastructures provide a direct support for FORM-L**
  - Also support from FIGARO (probabilistic analyses) and StimuLus (test case generation)
- **Work is also on-going to further facilitate the use of FORM-L**
  - Graphical FORM-L
  - Boiler plate sentences and diagrams that appear as (any) natural language but that represent well-formed FORM-L templates
- **An ITEA3 project proposal is being prepared (SAFE-INNOV)**
  - To better integrate behavioural modelling and simulation in systems engineering and systems operation activities
  - To address “systems of systems” and prospective analyses

# Thank you for your attention

# Any questions?