A Models-Aided Approach for the Engineering of Complex Socio-Technical Systems

Focus on Dynamic Phenomena

Along System Lifetime
(from Scoping Studies to Deconstruction)

From a System Owner Standpoint

Thuy NGUYEN
EDF R&D

Overview

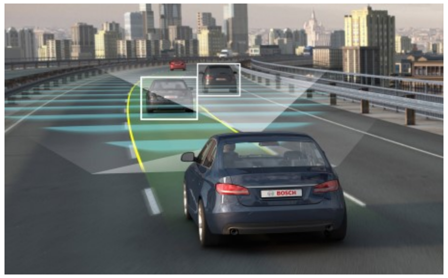
Large and Complex Technical Systems ...

- **... cannot be understood in all necessary aspects by single or few individuals, teams or disciplines**

Socio-Cyber-Physical Systems

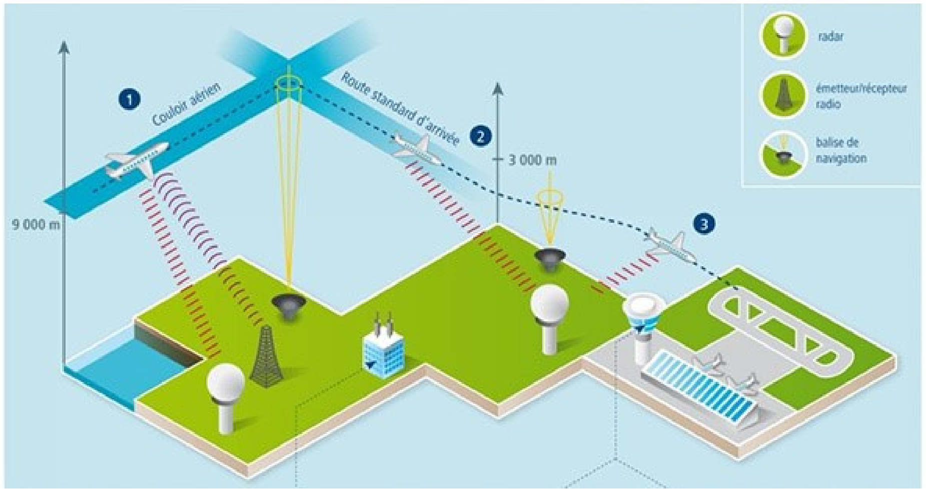
- **Most technical systems are socio-cyber-physical systems (SCPS)**
  - Human aspects
  - Computation & communication
    - Important, but not predominant
  - Process, geographic proximity, physical connections, ...

You ignore it at your peril

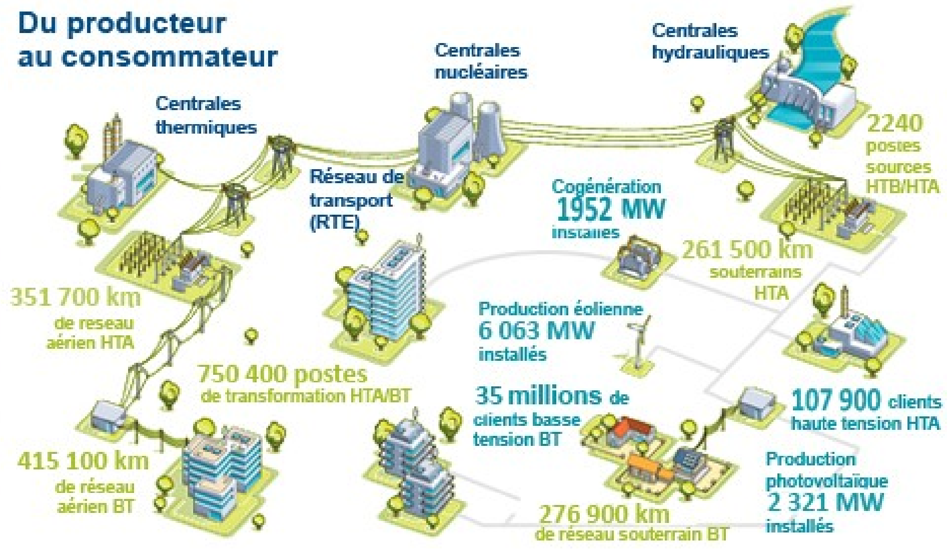


Systems of Systems

- **Some are also systems of systems (SoS)**
  - Multiple, loosely integrated SCPS, that may have different owners and operators, their own lifecycle, and may be included in, or removed from, the whole at different times



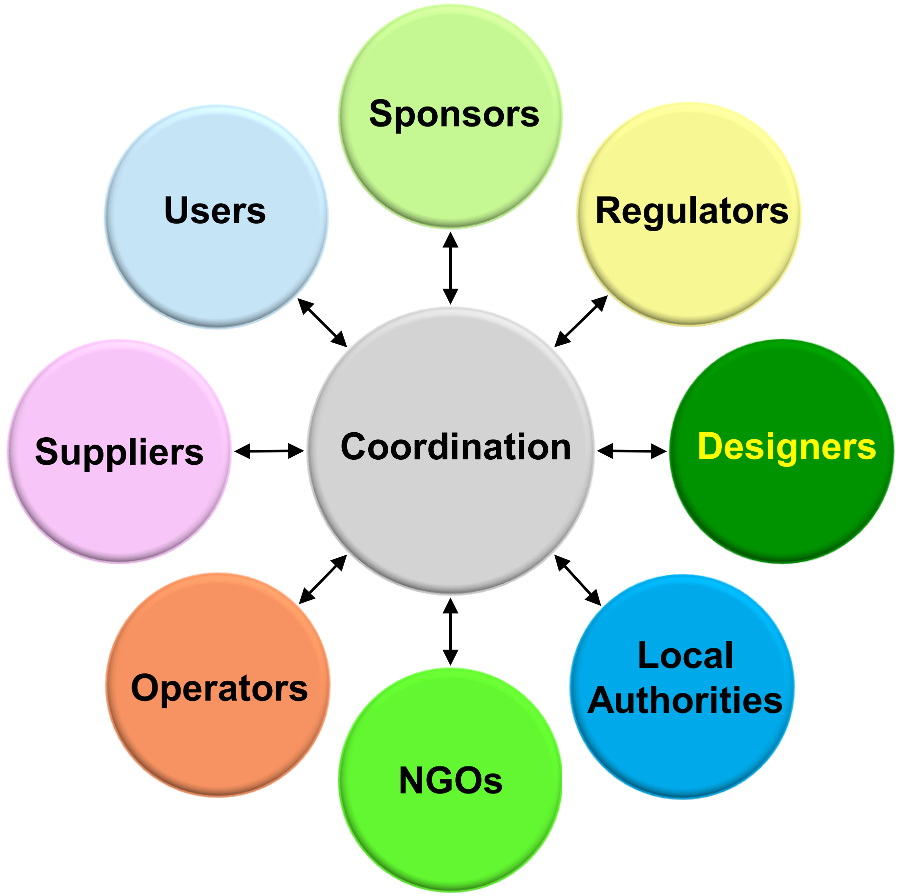
Air Traffic Control

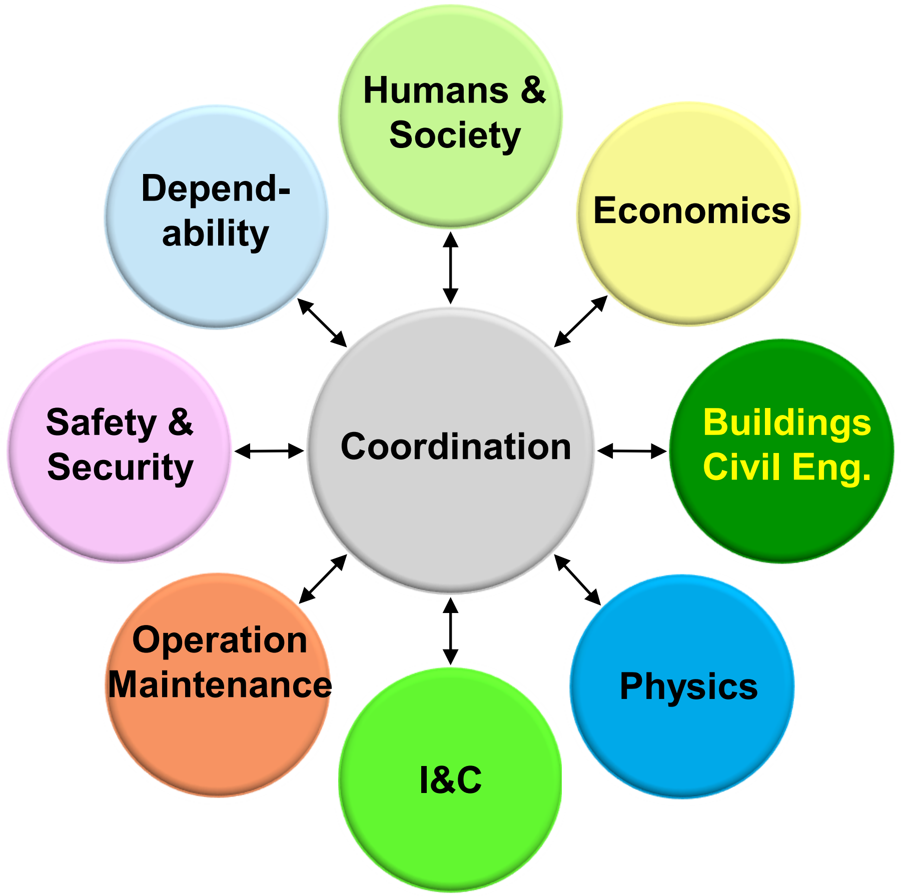


Du producteur au consommateur
Centrales thermiques
Centrales nucléaires
Centrales hydrauliques
Réseau de transport (RTE)
2240 postes sources HTB/HTA
Cogénération 1952 MW installés
261 500 km souterrains HTA
351 700 km de reseau aérien HTA
Production éolienne 6 063 MW installés
750 400 postes de transformation HTA/BT
35 millions de clients basse tension BT
107 900 clients haute tension HTA
415 100 km de réseau aérien BT
Production photovoltaïque 2 321 MW installés
276 900 km de réseau souterrain BT

Efficient & Safe Design, Construction and Operation ...

... Require the Cooperation and Coordination of Multiple Teams and Varied Disciplines, from many Different Viewpoints





and also Geography, Weather & Climate, Prospective studies, Construction, Logistics, ...

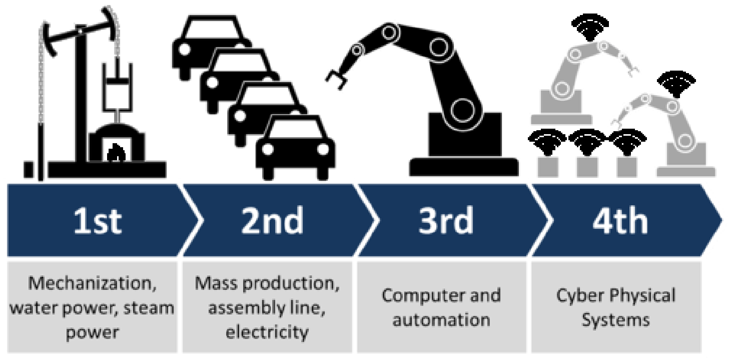
Their Engineering is a Daunting Challenge

- **Finding a common ground for multiple stakeholders**
  - With different viewpoints and sometimes antagonistic objectives
- **Understanding between engineering disciplines**
  - With different technical cultures and often antagonistic constraints
- **Coordination of numerous teams, but just as necessary**
  - Too much ➔ paralysis, too little ➔ chaos
- **Engineering efficiency**
  - Ability to innovate and find optimal solutions in spite of complexity and demanding safety, budgetary and time constraints
- **Coverage of complete system lifecycle**
  - From scoping and conceptual studies to deconstruction, through design, construction, operation, outage management and renovation
- **Maintaining engineering knowledge over system lifetime**
  - Several decades for large & complex SCPS, indefinitely for some SoS

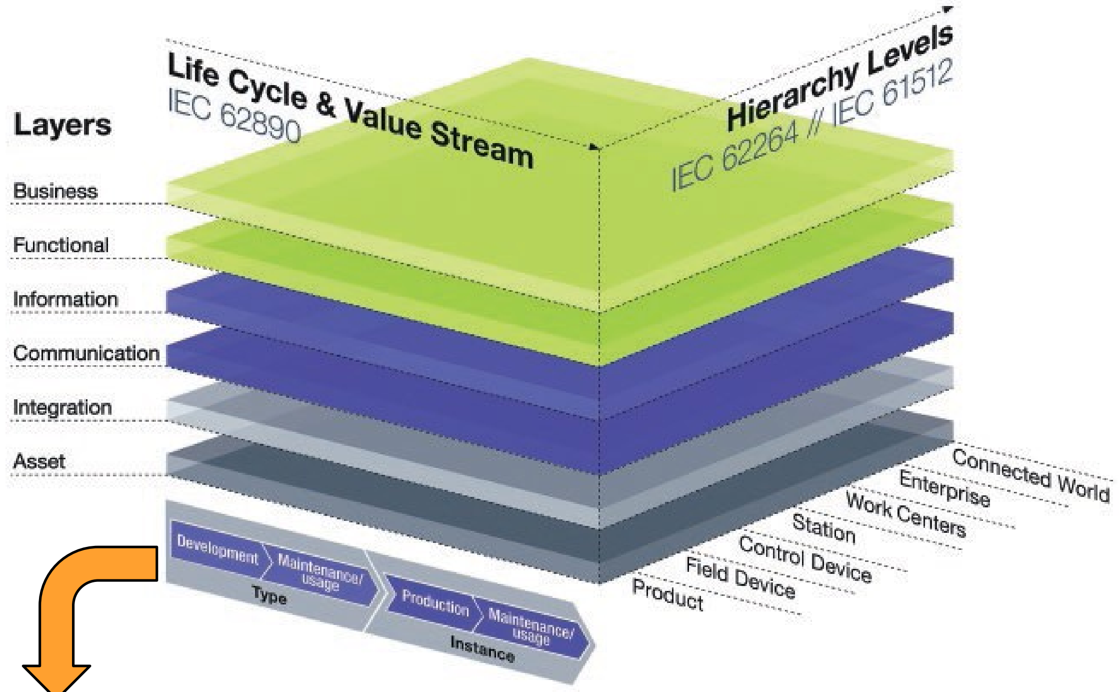
Proposed Approach

- **Focused on the mastery of dynamic phenomena and the explicit justification of engineering decisions**
  - Combined use of advanced modelling and structured justification frameworks
- **Advanced modelling of dynamic phenomena aims at providing unambiguous descriptions of key overall objectives (WHAT) and their solutions (HOW)**
  - Simulation and analysis facilitate understanding and help master complexity
  - With thrifty modelling, a model can support multiple engineering activities along lifecycle
    - Better chance of maintaining models consistent with the real system all along lifecycle
- **Justification frameworks provide means for clearly explaining the WHY of the solutions chosen all along lifecycle**
  - Developing an explicit justification usually helps in identifying weaknesses and enables fruitful discussion between concerned parties
    - When it is not too late nor too expensive to make changes in solutions or requirements
  - More informative knowledge representation than simple traceability links
  - May be closely integrated with modelling

An Intrustry 4.0 Perspective



1st
Mechanization, water power, steam power
2nd
Mass production, assembly line, electricity
3rd
Computer and automation
4th
Cyber Physical Systems



time

prospective studies, conceptual design

What assets will be needed, what is possible, considering wide uncertainties?

development (initial, retrofits, upgrades)

requirements
specification
architecture
detailed design
validation
integration
unit verification

Coordinate teams and disciplines
Manage complexity and conflicts

construction & commissioning

Optimise huge schedules with complex constraints and unplanned events

operation & maintenance

optimisation of operation

optimisation of monitoring & maintenance

Understand current system state, operate & maintain as well as possible

deconstruction outages

Optimise huge schedules with complex constraints and unplanned events

Maintain engineering knowledge regarding the system along its lifetime

Overview

MASE: Models-Aided Systems Engineering

- **MBSE (Model-Based Systems Engineering) to facilitate communication**
  - Between teams, disciplines, stakeholders, generations (in the case of very long lifetimes)
- **But also to support varied engineering activities all along lifecycle ➔ digital twin**
- **In reality, for SCPS and SoS: Models-Aided Systems Engineering (MASE)**
  - MBSE mainly for cyber aspects (code generation)
- **MASE for large, complex systems needs extensive tool support**
  - Models exploration: browsing, queries
  - Verification & validation: functional validation of requirements, step-by-step verification of solutions and implementations, failure analyses (FMECA, STPA, ...), probabilistic safety or dependability analyses, ...
    - Simulation (possibly on a massive scale)
    - Static analysis, formal verification of desirable or required properties
    - X-in-the loop testing
  - Optimisation: during design, construction, operation, outages, deconstruction
  - Aids for operation (in normal and abnormal situations, maintenance, ...)
  - Training: automatic generation of training scenarios and verification of trainees' actions

MASE and Aids for Operation

- **Data validation & reconciliation: use of digital models and mathematical methods to correct observation data in industrial processes**
  - Errors due to sensors response times, lack of precision, calibration or failures
  - To reduce margins of uncertainty
- **Data assimilation: combination of digital models with observation data**
  - To determine the most likely current state of the system
  - To interpolate sparse observation data
  - To determine initial conditions for digital forecast models ➔ what-if aids to help determine an appropriate course of action to reach a desired system state
  - To determine the causal factors that led to the current system state ➔ diagnostic & prognosis aids to optimise maintenance and outages
  - To determine digital model parameters based on observed data
  - ...
- **Optimisation of complex schedules**
  - Very large number of tasks with many constraints and subject to many external events
- **...**

Modelling ...

"Essentially, all models are wrong, but some are useful"
(George Box, 1919-2013)

- **Model: simplified representation of phenomena or objects of the real world**
  - Restricted to chosen phenomena and objects, and to their significant environnement
  - Restricted to chosen aspects
  - Simplified representation of these aspects

- **Many different types of models**

Dynamic phenomena

  - Requirements models
  - Functional models
  - Probabilistic models
    - Safety, availability, ...
  - Economic models
    - Costs & revenues
  - Operational procedures
  - Tasks scheduling models
  - Process & multi-physics models
    - E.g., finite elements models, Modelica or Simulink models
  - ...

Other forms of modelling

  - 3D models
  - Geographic models
  - Engineering databases
    - Static characteristics of components and system designs
  - ...

... the Dynamics of a System

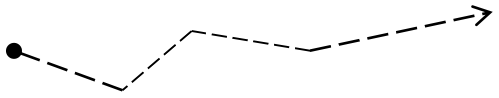
- **In the face of**
  - Normal stimuli: initial and boundary conditions
  - Time flow
  - Operational objectives: start-up and shut-down, load following, maintenance, testing, commissioning, modification, ...
  - Internal events: component failures, ...
  - External agressions: ambient conditions, seismic conditions, fire, flooding, failure propagation by physical invasion, malicious attacks, ...
- **Characterised by**
  - Nominal behaviour
    - In the various situations the system may face
  - Performance
    - Response times, accuracy, cost, ...
  - Quality of service
    - Fault tolerance, probabilities of failure on demand or in continuous operation, availability, ...
  - Acceptable failure behaviours

Various Forms of Modelling for Dynamic Phenomena

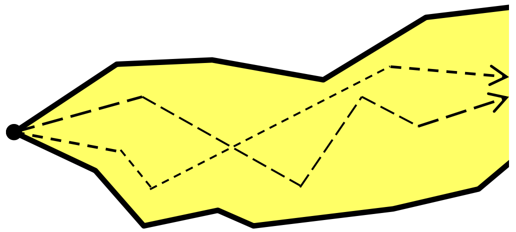
- **Non-formal or semi-formal modelling**
  - Natural language or drawings ➔ often ambiguous
    - Examples: SADT or SysML
  - Limited tool support ➔ a problem when dealing with complex systems
  - Individual models tend to address and be useful for only for a limited part of the lifecycle

Well-defined syntax & semantics enabling extensive tool support

- **Deterministic formal modelling**
  - Given initial and boundary conditions, only one possible behaviour
    - Examples: Modelica or finite elements models for physics, functional block diagrams for I&C, ...
  - Detailed and accurate ➔ only for downstream engineering activities
  - In general, specific to a discipline


Deterministic Model

- **Constraints-based formal modelling**
  - Envelopes of expected behaviours: avoid over-specification, enables simplification and abstraction
  - To model requirements, assumptions and preliminary solutions ➔ for engineering activities along the complete lifecycle
  - Can also represent uncertainties and human variability


Constraints Model

For Large and Complex Systems

- **Decomposition** (***divide et impera***) **➔ modular modelling, models composition, management and enforcement of interfaces**

- **Abstraction ➔ selective focus on aspects of interest for a given engineering activity**

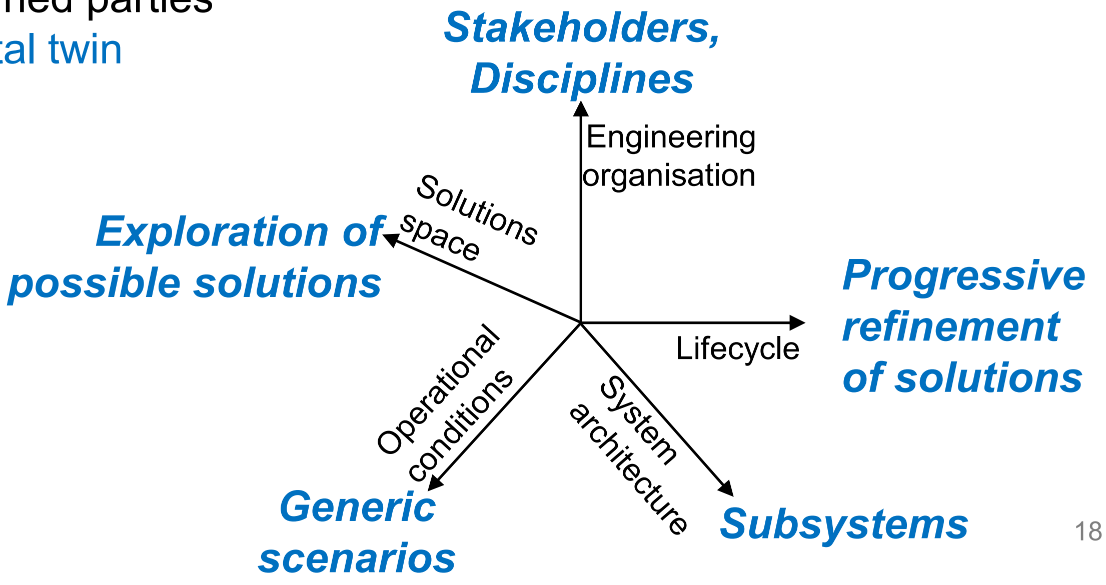
- **Coverage of system lifecycle ➔ progressive discovery of the system and its operation, variety of disciplines and activities**

- **Optimisation ➔ Exploration of multiples solutions, finding optimal trajectories in behavioural envelopes**

- **Verification & validation ➔ Formal verification, massive simulation, coverage testing, regression testing, X-in-the-loop testing**

Modelling on Multiple Axes

- **Conventional deterministic modelling: 1 system aspect - 1 all-encompassing model**
  - Big models are often expensive and difficult to develop and debug
  - Models for different aspects of the same system cannot or are very difficult to combine
    - Buth recent FMI (Functional Mockup Interface) standard is a big improvement
- **Modular contraints modelling: 1 system - multiple small models**
  - Each representing a specific aspect or part of the system, the viewpoint of a particular team, discipline or stakeholder, a particular solution, a generic simulation scenario, ...
  - Easier to develop and debug
  - Can be combined at will to suit the specific needs of a particular engineering activity and to take account of the concerned parties
    ➔ configurable, protean digital twin

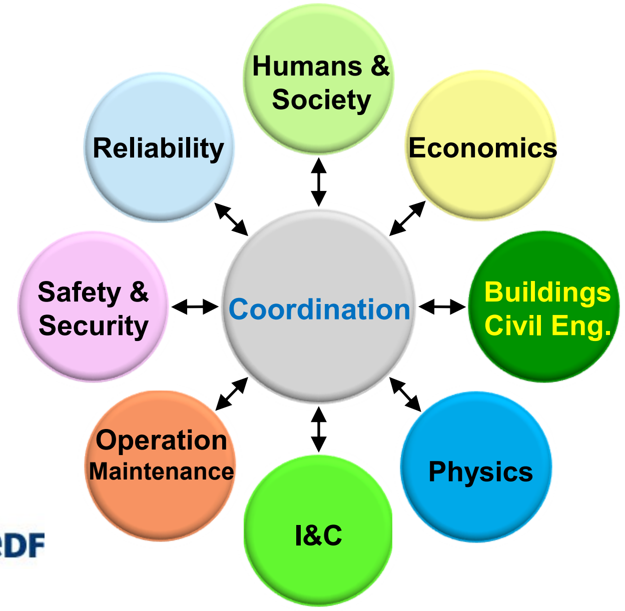


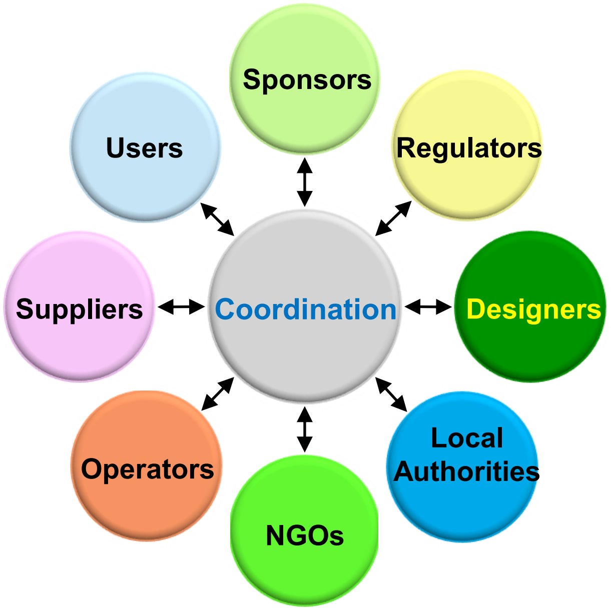
Multiple Teams and Disciplines

- **Stakeholders and disciplines often have their own methods and modelling languages for deterministic modelling**
  - And must be able to keep using them

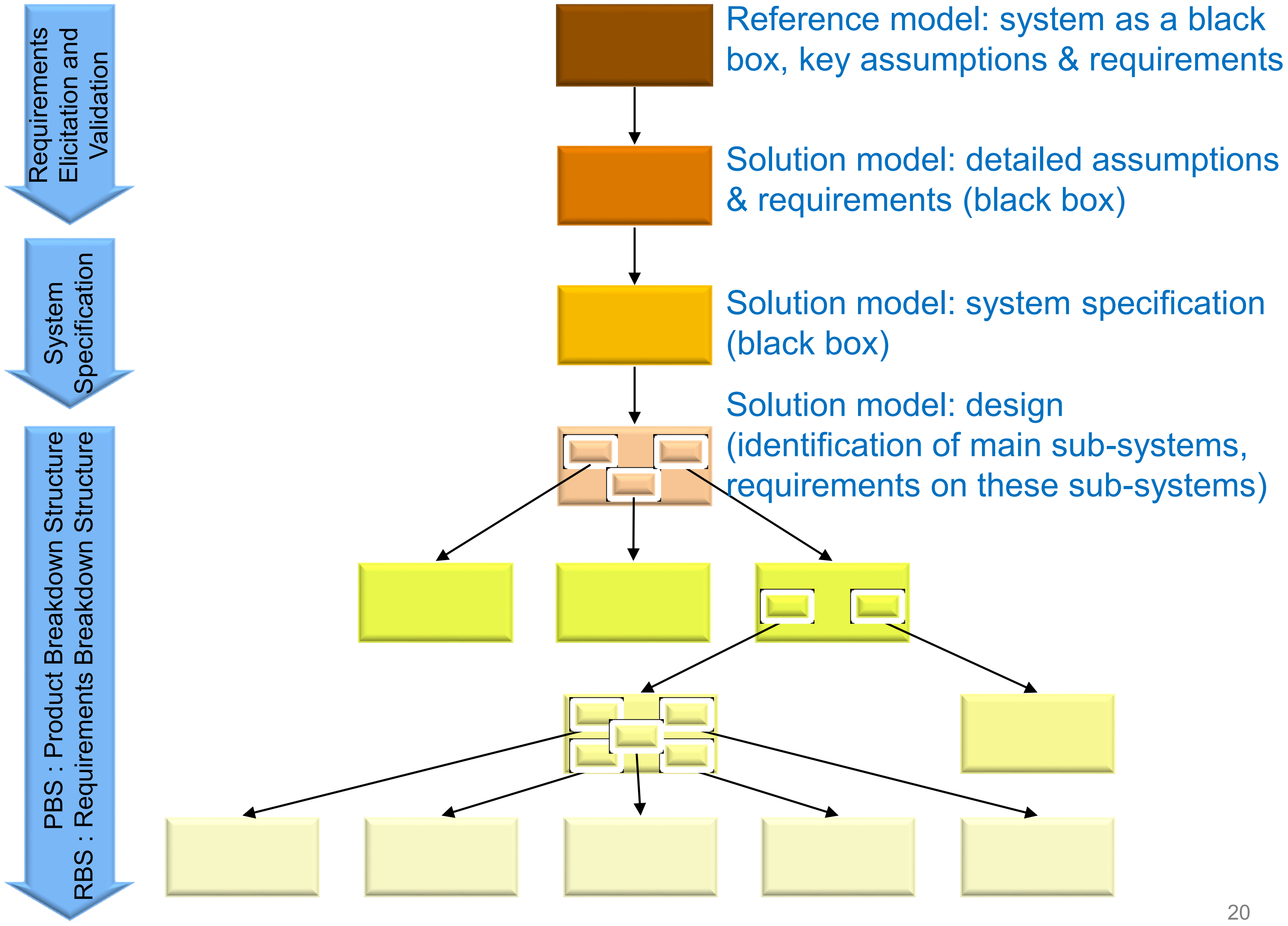
- **A common constraints language may be used to model what each stakeholder, discipline or team**
  - Expects from, and guarantees to, others (contracts)
  - Assumes from system environment
  - Requires of the system

- **Constraints models must be able to interface and interact with disciplinary models**





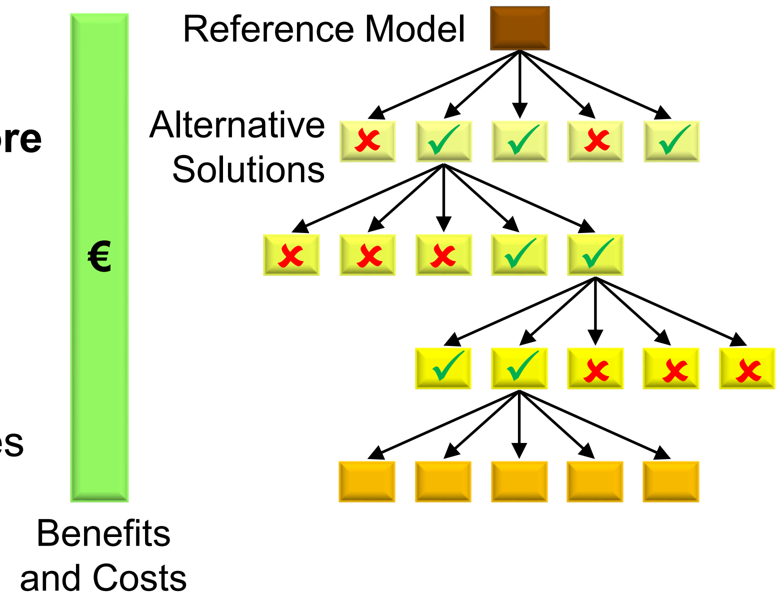
Progressive, Step-by-Step Refinement of Solutions

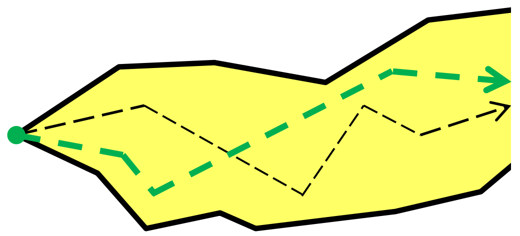


Requirements Elicitation and Validation
System Specification
PBS : Product Breakdown Structure
RBS : Requirements Breakdown Structure
Reference model: system as a black box, key assumptions & requirements
Solution model: detailed assumptions & requirements (black box)
Solution model: system specification (black box)
Solution model: design (identification of main sub-systems, requirements on these sub-systems)

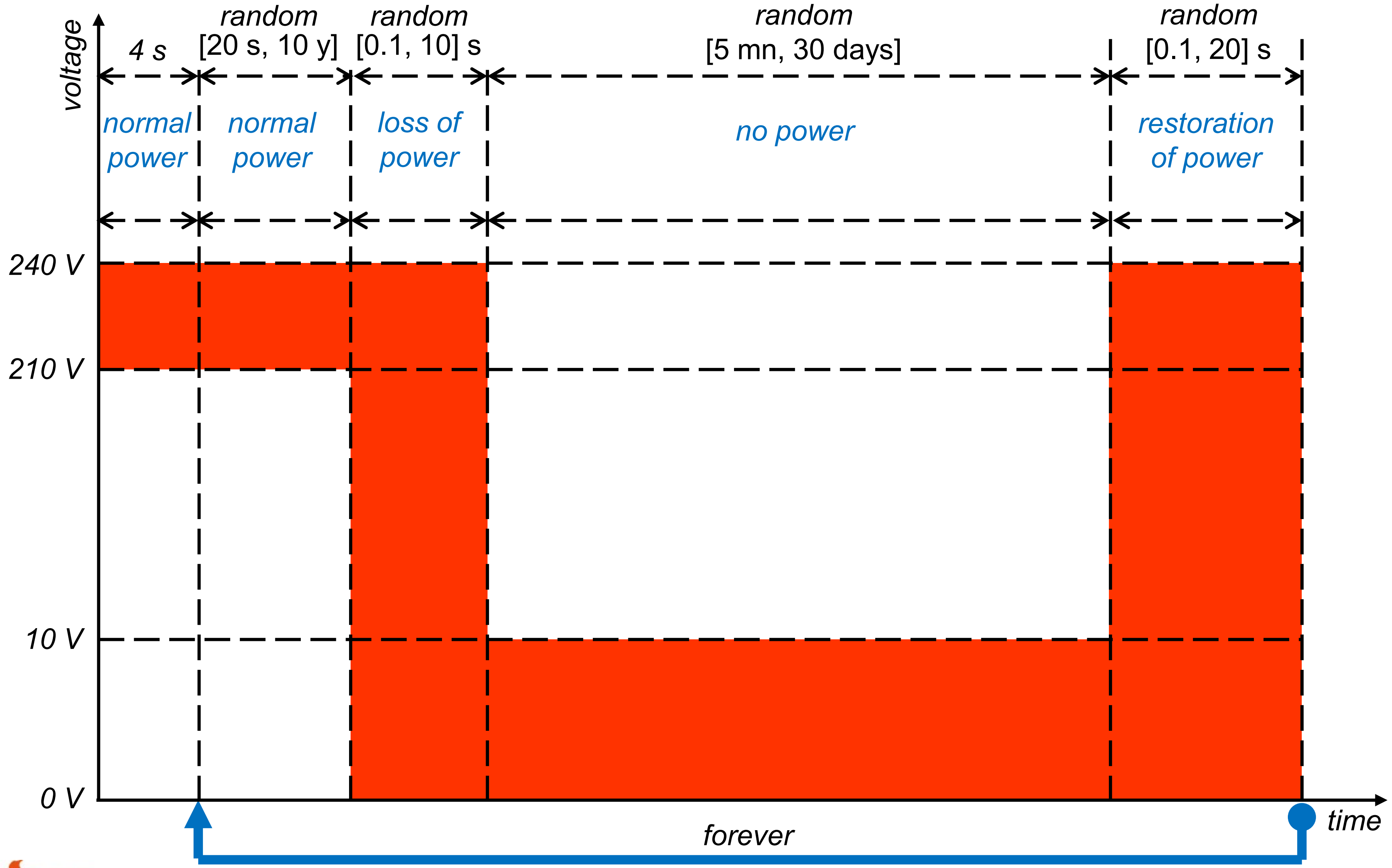
Optimisation: Exploration of Possible Solutions

- **Competition, financial constraints, deadlines, …**
- **Need to optimise and innovate, and thus to explore multiples solutions**
  - Preferably early in, and then at each stage of, the engineering process
  - Manually developed solutions
  - Possible application of more systematic approaches such as genetic approaches
- **Also, need to find (near) optimal solutions in envelope models**
- **Diverse evaluation criteria**
  - Satisfaction of requirements
  - Cost of construction and profitability of operation (including maintenance)
  - Safety and security justification
  - …




Optimal solution in a constraints envelope

Generic Scenarios - Example (Power Supply)



Overview

- **Motivation**
- **Modelling of Dynamic Phenomena**
  - Modelling
  - Methodology
- **Justification Frameworks**

General Approach

1. **Develop a Reference Model for the system**
   - This constraints model will be used as the touchstone for the verification of solutions
   - It views the system as a black box and specifies the key system requirements

2. **Validate the Reference Model**
   - To make sure it correctly represents the intentions of the authors

iterate

3. **Develop a constraints-based Solution Model, and verify it against the Reference Model and the previous Solutions Models**
   - It may still consider the system as a black box and add more detailed, less essential requirements or specification items
   - Or it may start decomposing the system into subsystems and specify their requirements and interactions: it can serve as a Reference Model for each subsystem

4. **Multiple alternative Solution Models may be developed to explore the space of possible solutions**

5. **At some point, deterministic Disciplinary Models are available and may be integrated**
   - Constraints models may be used to coordinate multiple, different disciplinary models

Reference Model

A. **Identify the environnement entities that interact with the system**
   - Other technical systems, human actors, the physical environment

B. **Identify situations**
   - System states, environment entities states, operational goals, transitions
   - Need to also address abnormal situations

C. **Identify flows between the system and its environment**
   - Fluids, information, events
   - May depend on situations (e.g., agression and invasion in some abnormal situations)

D. **Model the assumptions made by the system regarding its environment**
   - May also depend on situations

E. **Model the requirements placed on the system**
   - Some requirements may be placed on the system by its environment, others are placed directly on the system
   - May also depend on situations

In the case of Multiple Stakeholders

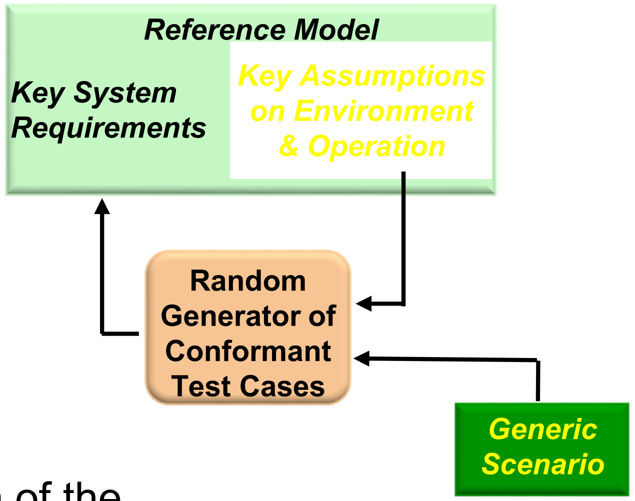
- **Since the Reference Model must be verified by traditional means and will be the touchstone for all further Solution Models, it needs to be as simple as possible**
  - It should focus on the "raison d'être" of the system, on the main issues and on the make or break requirements
  - Further models will represent solutions, even when they express more detailed system requirements or specifications
- **Different stakeholders may have very different expectations on the system, but not all stakeholders need to contribute to the Reference Model**
- **Also, in general, different teams and disciplines from the same stakeholder may need to contribute to the Reference Model**
- **Each may develop their own partial model**
  - The complete Reference Model is the combination of these partial models

Validation of the Reference Model

- **Reviews and inspections**
  - Coverage and correct representation of relevant statements of input documents
  - Compliance with modelling rules
  - ...
- **Tool-supported analysis can help detect contradictory constraints**
  - For which there is no possible solution
  - Often due to conflicting stakeholders expectations
- **Simulation**
  - To verify that the model behaves as intended
  - To help understand the specified assumptions and requirements, and their effects
  - To verify that no situation will lead to unacceptable behaviour
  - To help stakeholders decide whether proposed tradeoffs are acceptable

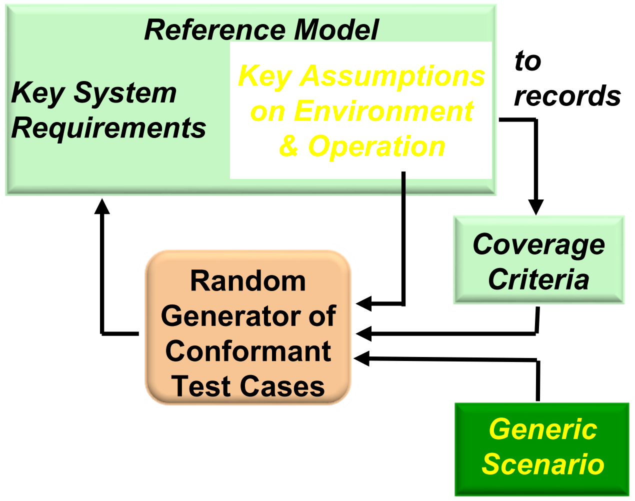
Minimal Models & Scenario Models

- **The system views the entities of the environment preferably through contracts**
  - That formally specify their mutual interfaces
- **Initially, Minimal Models should be used for the environment entities**
  - Just satisfying their respective contracts: thus, one does not make implicit assumptions
  - Focus on the system under study, reduction of complexity and necessary computing power
  - More accurate models may be used later to detect possible emergent behaviours
- **Generic Scenario Models may be used to guide test case generators towards cases of particular interest**
  - Expressed as additional assumptions
- **Test cases may be generated automatically**
  - With tools such as StimuLus (from ArgoSim)
  - Test cases generated randomly, but consistent with assumptions and definitions
  - Automatic test case generation and automatic verification of the satisfaction of requirements enable massive simulation and the exploration of very large numbers of situations


Reference Model
Key System Requirements
Key Assumptions on Environment & Operation
Random Generator of Conformant Test Cases
Generic Scenario

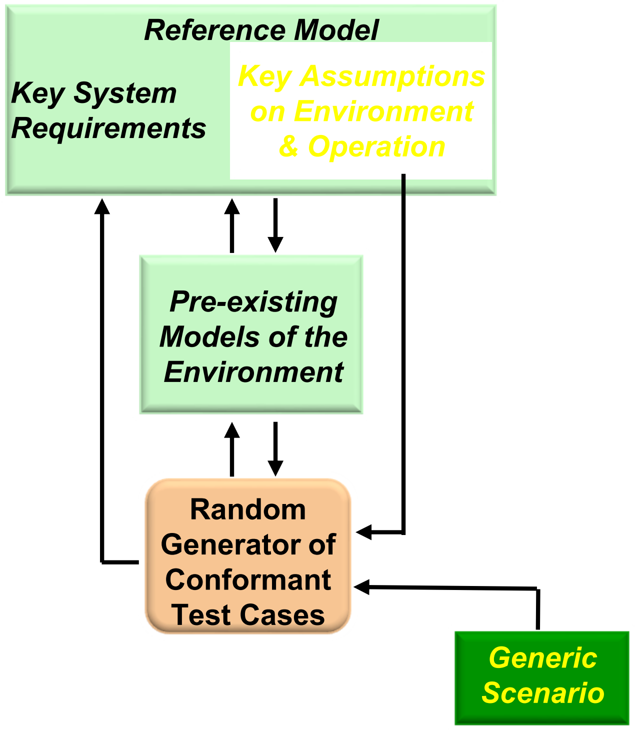
Test Coverage

- **Many test cases are necessary to ensure that a model is adequately challenged and to gain adequate confidence in the model**
- **Test coverage criteria and objectives to be collectively satisfied by the test cases may be specified**
  - Many criteria are possible, e.g., visiting each state of each automaton, taking each state transition of an automaton, etc.
- **Criteria and objectives could be formally specified as requirements**
  - Not with respect to the system under study, but to the recorded simulation results
- **As simulation records accumulate, the test case generator may be guided to improve coverage**


Reference Model
Key System Requirements
Key Assumptions on Environment & Operation
to records
Coverage Criteria
Random Generator of Conformant Test Cases
Generic Scenario

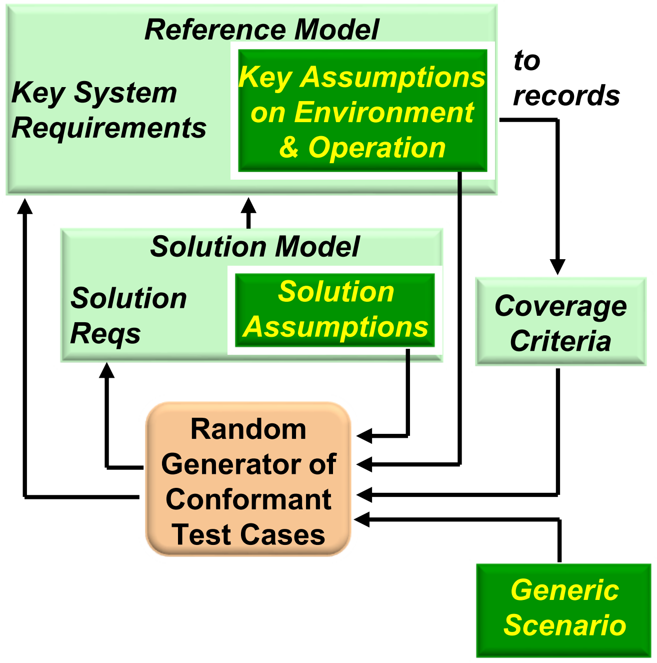
Pre-Existing Models for the Environment

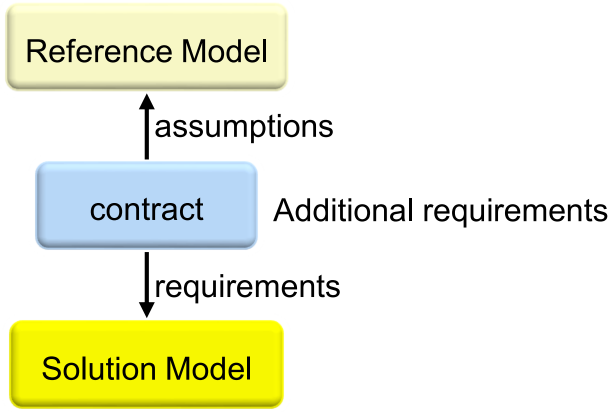
- **When all or part of the system environment is well-understood et already modelled in detail (in FORM-L or not), the validation of the Reference Model may make use of these models**
  - E.g., in the case of a partial system upgrade or a new system to be integrated in an existing SoS


Reference Model
Key System Requirements
Key Assumptions on Environment & Operation
Pre-existing Models of the Environment
Random Generator of Conformant Test Cases
Generic Scenario

Solution Models – Detailed System Requirements

- **Reference Model limited to key requirements**
- **More detailed, less essential requirements may be specified in subsequent constraints models viewed as Solution Models**
  - The system is still a black box
  - Additional requirements may reflect the needs of second rank stakeholders, construction, operation, maintenance, modification, non-essential interactions with the environment, ...
- **For simulation purposes, the Reference Model views the additional requirements as assumptions**
  - The test case generator produces compliant behaviours to be checked against reference requirements
- **The Solution Model views them as requirements**
  - To be satisfied by further, more detailed solution models
- **This dual view (assumptions - requirements) may be modelled with contracts or with model extensions**





Solution Models – System Specification

- **System specification: constraints-based system description as one solution to system requirements**
  - A project owner issues a tender specifying the system requirements
  - Different bidders reply, each with their own system specification
  - The system is still viewed as a black box, or as a dark grey box
- **Need to determine whether a system specification complies with the requirements**
  - Often far from straightforward

Solution Models – System Overall Design

- **Once the system specification is verified, a system architecture can be developed**
  - Identification of main components
  - Assumptions on components behaviours and interactions
    - Allocation of system requirements or specification to components
  - Contracts may be established between the architecture and its components, so that the assumptions made by the architecture are requirements for the components
  - Contracts may also be established between components
  - Enables early failure and probabilistic analyses (safety, dependability)
- **Verification with components behaviours consistent with assumptions**
  - Contract or extension between the system specification model and the architecture model
  - Minimal component models: no need to wait for detailed design solutions
- **Some components may be considered as systems of their own, and the same process is applied iteratively**

Solution Models – Detailed Design & Implementation

- **As design becomes more detailed, the precise behaviour of individual components can be represented by deterministic behavioural models**
  - E.g., in MODELICA for physical processes, in functional diagrams for I&C functions
- **Detailed design decisions need to comply with the overall design**
  - Model-in-the-Loop verification
- **Accurate simulation helps make sure that the assumptions made at earlier stages are appropriate**
  - Identification of possible emergent behaviours
- **At a further point, a component may be represented by an implementation**
  - Software-in-the-Loop, Hardware-in-the-Loop verification
- **In both cases, automatic generation of test cases and verification of test results**

Overview

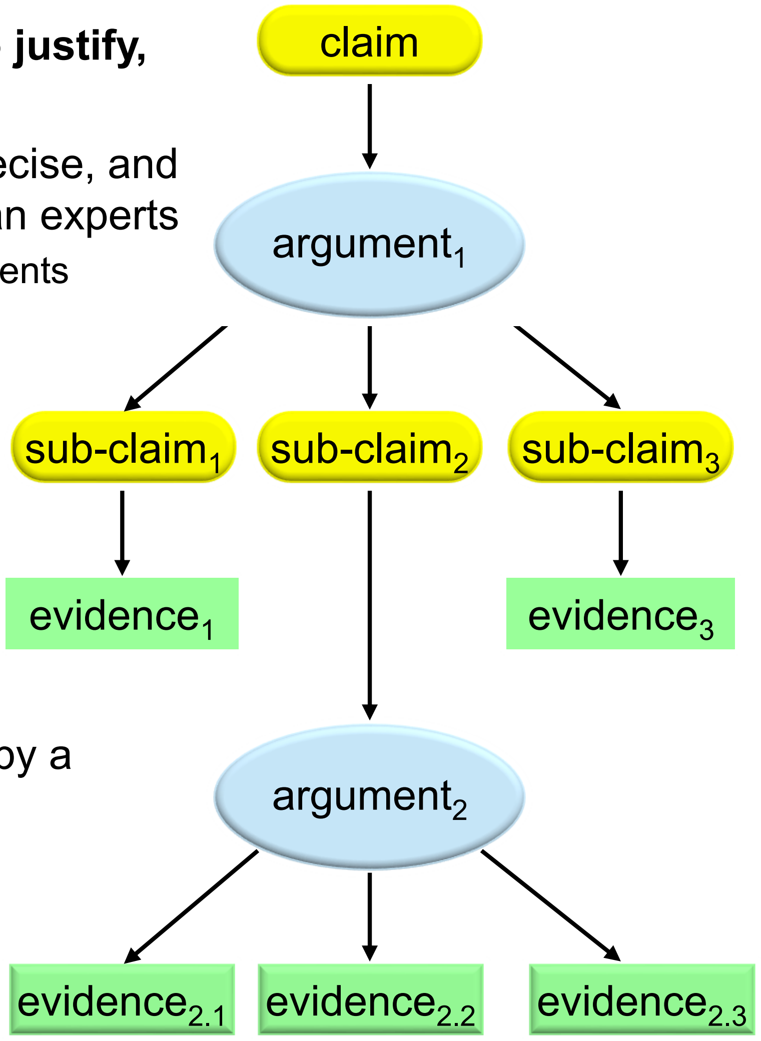
- **Motivation**
- **Modelling of Dynamic Phenomena**
  - Modelling
  - Methodology
  - The FORM-L language
- **Justification Frameworks**

Justification Frameworks

- **System modelling describes the system and its environment as they are (or as they will or should be) in objective and quantitative terms**
  - Composition, Interfaces, Requirements & Assumptions, Situations, ...
- **It addresses the WHAT, the WHEN, the WHERE, the HOW WELL and the HOW**
- **It only gives a partial answer to the WHY**
- **A justification framework aims at organising and structuring the justification that a system complies adequately to requirements that are sometime vague and ambiguous**
  - Including standard or regulatory requirements
  - Objective / quantitative aspects, but also subjective / qualitative aspects
- **It provides a more complete answer to the WHY**
- **Important justification elements may be provided by the modelling**

Claim, Argument, Evidence (CAE)

- **A property, the satisfaction of which one seeks to justify, is represented by a claim**
  - At the beginning, some claims are vague and imprecise, and are left to the understanding and judgment of human experts
    - This is often the case of regulatory or standard requirements
  - E.g., defence against digital common-cause failure
- **In the simplest cases, the claim can be justified by a simple piece of objective evidence**
- **In more complex cases, a reasoning (argument) is applied to transform the claim into one or more sub-claims**
  - Each sub-claim needs in turn to be justified, either by a piece of evidence or by another reasoning
  - And so on
- **A claim that is not supported by any argument and associated evidence is an assumption**



A Well-Studied Approach

- **ISO - CEI - IEEE 15026-2 (2011) Systems and software engineering - Systems and software assurance -- Part 2: Assurance Case**

- **EURATOM project HARMONICS (2011-2014)**

  Harmonised
  Assessment of the
  Reliability of
  MOdern
  Nuclear
  Instrumentation &
  Control
  Systems

  - VTT (Finland), EDF (France), ADELARD (UK), ISTEC (Germany), SSM (Sweden)

- **Mainly used for safety justification**

- **But applicable to the justification of other types of properties**

Benefits

- **Complex chains of reasonning made explicit**
- **Much like the mathematical demonstrations**
  - Where a theorem (claim) is progressively broken down into lemmas (sub-claims)
  - And where each reasonning step is explicit so that it can be understood, verified and if necessary, challenged
- **The main difference is that justification frameworks make room for human judgment**
  - Some requirements are inherently impossible to satisfy in the absolute
    - Example: independence of two systems
  - Other are impossible to prove formally
    - Example: failure rates of high quality digital systems
- **Developing an explicit justification usually helps in identifying weaknesses**
- **Expression and explanation of solution strategies and principles early in lifecycle, enabling timely discussions between concerned parties, when solution details are not frozen yet**
  - Applicable throughout the engineering process

Different Types of Argument - 1/2

- **Concretion**
  - Transformation of an imprecise or ambiguous claim into one or several precise and unambiguous sub-claims that collectively provide an interpretation of the original claim
  - Initial requirements (and also regulatory and standard requirements) are often voluntarily vague, expressing ideal objectives rather than precise criteria that would obscure the objective and not necessarily be adequate for all situations
- **Substitution**
  - Transformation of a claim on an object into another claim on the same object, or into the same claim but on another object
  - A side sub-claim should justify that the substitution is legitimate
  - Example: test on a test specimen rather than on the specimen(s) that will be in operation
    - A side sub-claim should justify that the test specimen is adequately similar to the specimen(s) to be used in operation, at least for what concerns the test being considered

Different Types of Argument - 2/2

- **Decomposition**
  - Decomposition of the claim or of the object that is the target of the claim
    - The classical divide-and-conquer approach
  - A side sub-claim should justify that the satisfaction of the other sub-claims implies the satisfaction of the claim
- **Calculation**
  - Applies to quantitative claims, and combines decomposition with a computation formula
  - A side sub-claim should justify that the formula is adequate
  - Examples: calculation of maximum response time, of reliability, ...
- **Evidence incorporation**
  - When a single piece of evidence is sufficient in itself to justify the claim
  - Example: the measurement of a static feature

Strategies for Safety Justification

- **In general, three main classes of claims**

- **Conformance to rules (indirect properties)**
  - International standards
  - National regulations or practices
  - Development codes

- **Direct, application-specific properties**
  - Behavioural properties
  - Probabilistic properties
  - Structural properties

- **Tolerance with respect to postulated residual vulnerabilities**
  - Justification that they cannot lead to unacceptable, or simply undesirable situations

Conclusion

- **Informal approaches are useful in the initial steps, when the stakeholders and disciplinary teams need to understand each other's main points**
  - But provide little help in mastering complexity and little continuity along lifecycle
- **Formal constraints modelling provides objective, unambiguous views on the system, its environment and its operation all along lifecycle**
  - Forces to add precision when necessary, to make sure that there is no ambiguity
  - Sometimes at the expense of simplicity and clarity
    - Simulation and analysis can help understand complex modelling
- **Justification frameworks provide explicit rationales behind engineering decisions**
  - Richer semantics than with simple traceability links
  - Can represent subjective human judgment
- **The combination constitutes a powerful means for systems engineering and for representing the engineering knowledge regarding a given system**
  - Important for long-lived systems that will need to be maintained, retrofitted and upgraded by future generations

Thank you for your attention

Any questions?