An Overview of Applied Systems-theoretic Method

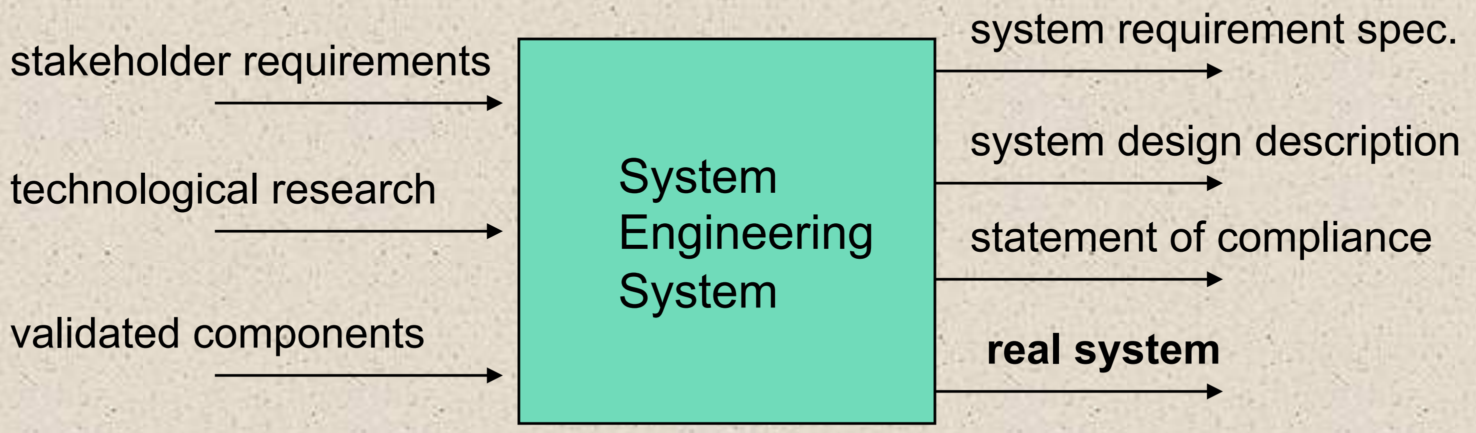
Doing Systems Engineering

As with all other engineering activities, systems engineering requires an appropriate application of:

- a rigorous and comprehensive theoretical method
- an efficient process (project lifecycle management)
- appropriate tools (automation of design 'documentation', verification and validation techniques)

This presentation is concerned with **systems engineering method** - in particular the practical use of mathematical systems theory.

The System Engineering System



The System Engineering System is the human + technology endeavour, used to design and deliver a solution that best satisfies the needs of the stakeholders.

The System Engineering System (2)

The System Engineering System (SES) will be complex, and the process will be complicated. It will, itself, need to be 'system engineered' ! The aim is to have an optimal means of delivering real system solutions.

The SES customer will be the Project Team. It will be implemented by Project Management.

Application of Theory

Particular engineering disciplines apply methods that make use of particular theoretical paradigms, for example:

- **Electronic Engineering**: Electrical Circuit Theory, Logic Circuit Design
- **Mechanical Engineering**: Machine Theory, Kinematics
- **Control System Design**: State-space Analysis
- **Engine Design**: Thermodynamics, Heat Engine Cycles

What, then, for **Systems Engineering** - what is (or should be) the theoretical basis?

The use of systems-theoretic method

Is based on mathematical systems theory, and is used to formally specify:

- complex systems constructs
- complicated behaviours
- hardware, software, bio-ware
- hybrid technological implementation
- identification of optimality criteria and constraints
- conformance and compliance testing.

Systems Formal Specifications

Design and behavioural descriptions using set theory language (syntax) and mathematical systems-theoretic constructs (semantics).

Formal proofs (of correctness) using axiomatic systems-theoretic formulations, and propositional calculus.

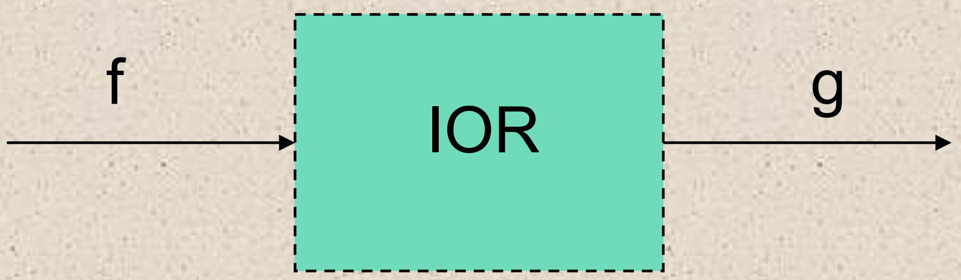
Supporting graphic modelling: system block diagrams (Connectivity Diagrams); behavioural modelling (Venn diagrams); directed-graph order relationships (Hasse line diagrams).

Requirements Specification

SDR = (IOR, TYR, TR, STR)

where:	IOR	is the input/output ,
	TYR	is the technology ,
	TR	is the trade-off , and
	STR	is the system test

IOR Input Output Requirement



For every possible input trajectory (f) , all eligible output trajectories (g) are specified in terms of an 'eligibility function' (i.e. $g \in ER(f)$) .

IOR Elaboration/Simplification

The IOR input/output requirement is subsequently (formally) expressed in an elaborated form:

$$IOR_{original} = HIMIO(IOR_{elaborated}, HI, HO)$$

Q. If there is a functional design solution for $IOR_{original}$, is there a functional design solution for $IOR_{elaborated}$?

Q. If there is *no* functional design solution for $IOR_{original}$, then is there *no* functional design solution for $IOR_{elaborated}$?

TYR **Technology Requirements**

A technology requirement is specified in terms of a particular set of (complex) system models that describe the technological solutions that are *mandated*, or (as is more usual) *prohibited* - in terms of acceptable system architectures (component choices and interconnections).

The technology is specified by an ordered list of viable system architectures.

TR **Trade-off Requirement**

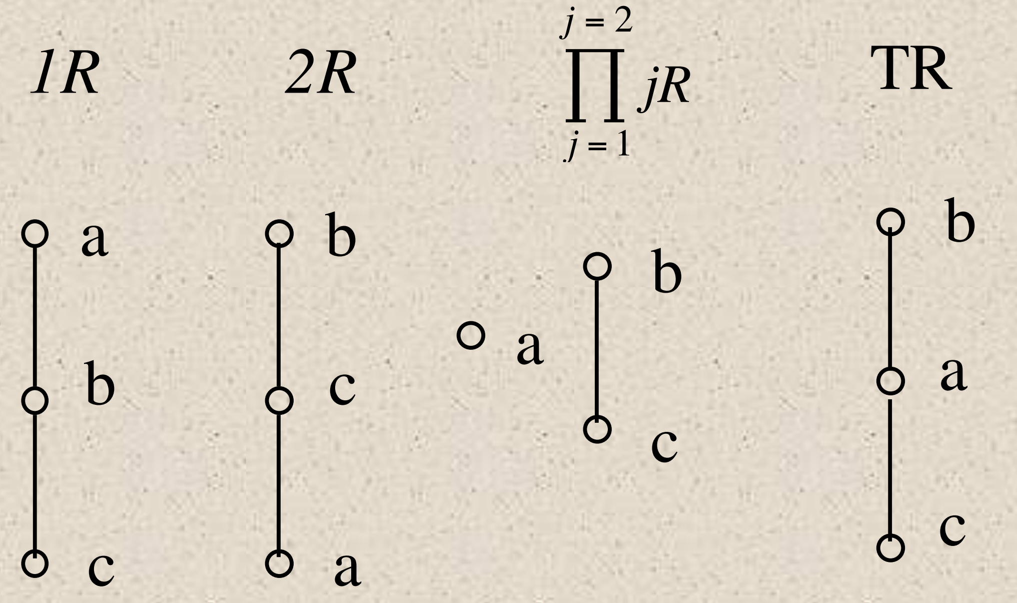
The trade-off requirement is defined in terms of a comparative order over a set S of (feasible) candidate solutions:

$$TR = RFO(S, TF)$$

such that a preference between any two candidate system solutions can be expressed - the objective being to be able to identify the 'best' (or, at least, an optimal) system solution.

Q. How, then, do we determine the function TF - what is it's relationship to the various evaluation criteria for optimality identification? How does TF embody the evaluation criteria?

TR Trade-off Requirement (2)



The trade-off requirement TR operates on the product-order of the evaluation criteria -it adjudicates between ‘non-comparable’ or ‘equivalence’ relationships.

STR System Test Requirement

The system test requirement STR is used to specify those tests of system compliance (satisfaction of IOR and TYR requirements) that are required to be conducted by, or on behalf off, the system customer/owner.

System tests may involve dynamic experimentation (trials scenarios), physical inspection, etc.

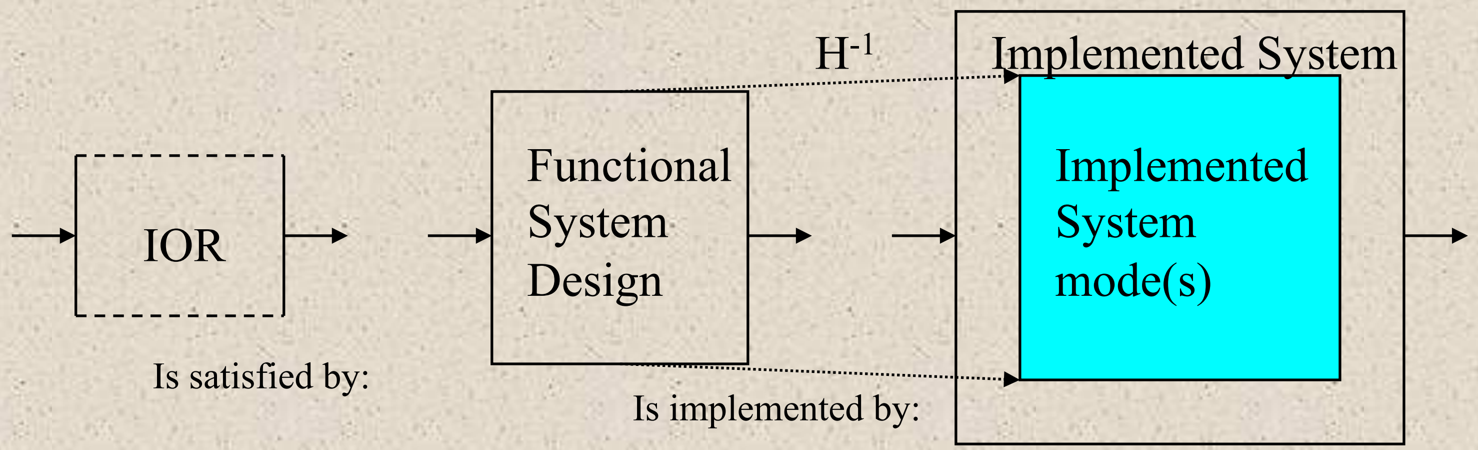
System Construct Modelling

System definition : Z = (SZ, IZ, OZ, NZ, RZ)

where:	SZ	is the set of system states ,
	IZ	is the set of system inputs ,
	OZ	is the set of system outputs,
	NZ	is the 'next-state' function, and
	RZ	is the 'read-out' function

The system model may representative of a continuous phenomena (e.g. as a set of differential equations) or a discrete phenomena (e.g. as a discrete sequential machine).

Implementation



A satisfaction of the input/output requirement is provided by some Functional System Design.

The Functional System design is satisfied by means of a mode (or modes) of behaviour of the system implementation - which, itself, satisfies the technology by choice of components and architecture.

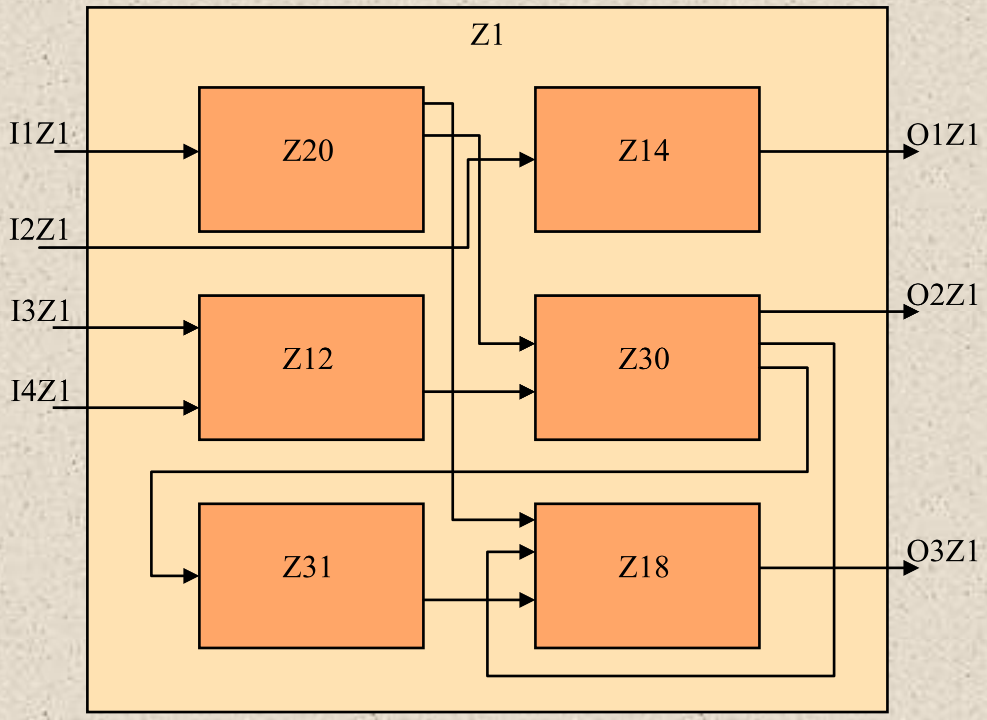
Design Re-use and Read-across

$$\begin{tikzcd}
b \arrow[rr, leftrightarrow, dashed, "\text{I/O equivalence}"] & & c \\
& a \arrow[ul, dashed, "H^{-1}"] \arrow[ur, dashed, "H^{-1}"'] &
\end{tikzcd}$$

Systems b and c both have *full functional equivalence* to system a. Also, an input/output equivalence between b and c is demonstrated.

Q. Are systems b and c functionally equivalent?

Implementation Design Complexity



System (organizational) complexity - expressed in terms of a multiplicity of interconnected components.

Complexity - Architectural Forms

Components and interconnections - taxonomies

Sub-systems

System polymorphism.

Adaptive (and auto-adaptive) systems.

Programmable systems.

Systems of systems.

Complexity - Realisation

Different technologies: hardware/software/bio-ware

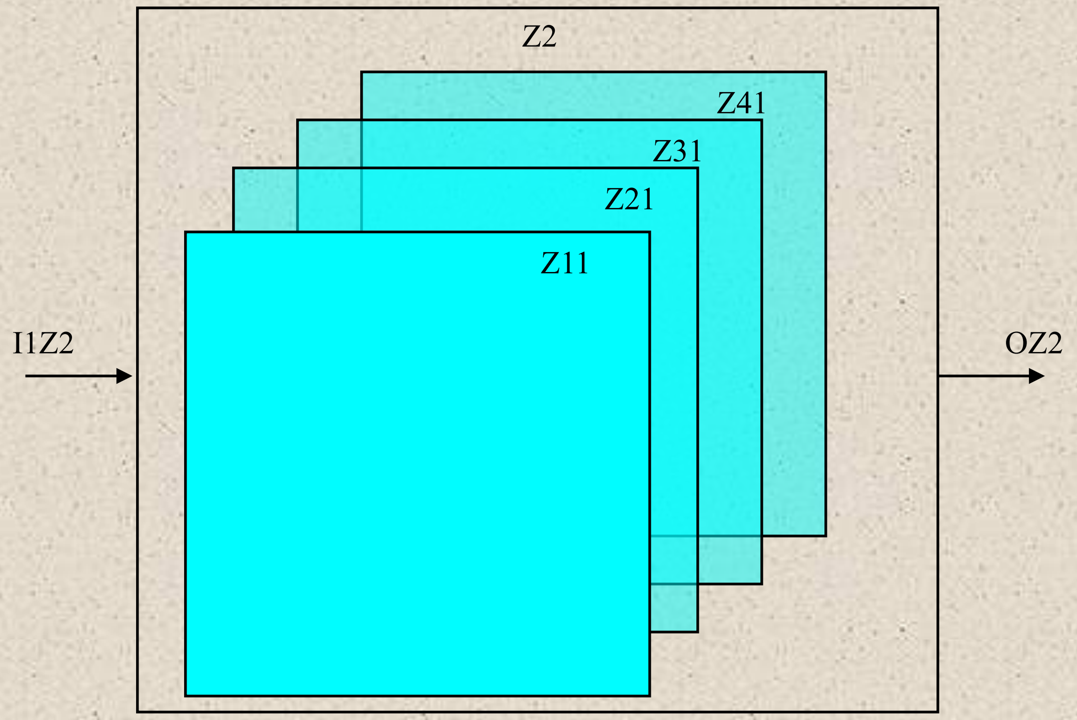
Hybrid implementation - mix of technologies

Adds to design complexity (more inputs, outputs and interconnections)

Adds to complication (non-operational modes of behaviour).

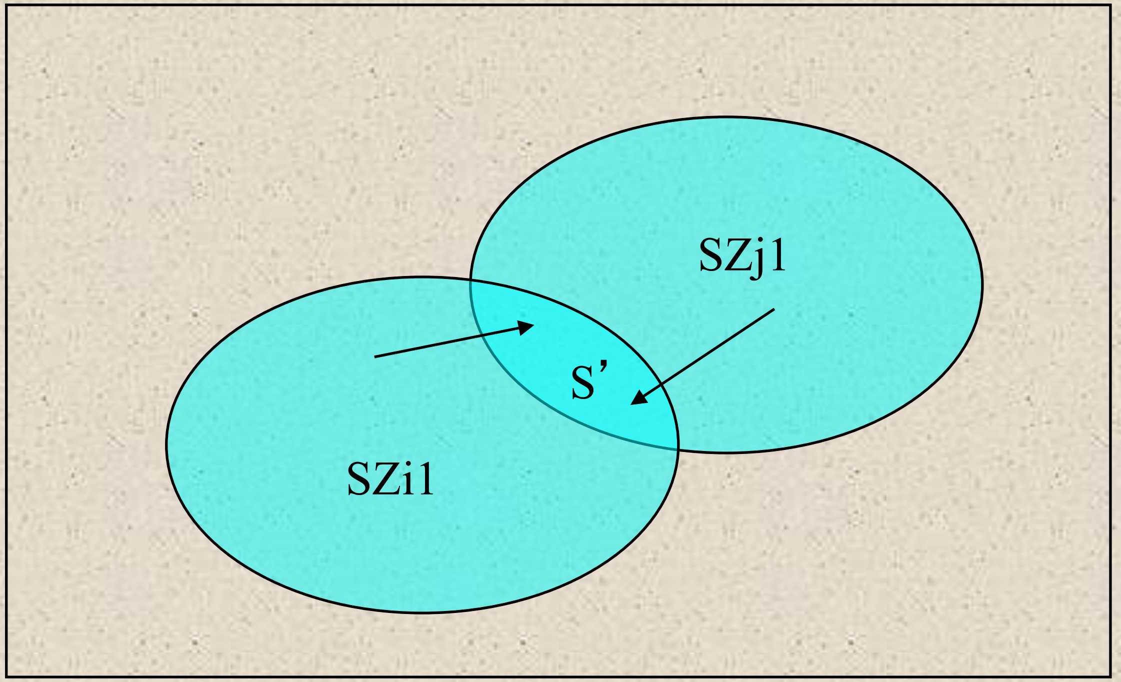
Q. What techniques can we use to capture and model these characteristics of complex real-system implementation?

Implementation Design Complication



System complication: multiple modes of behaviour.

Behavioural Design and Analysis



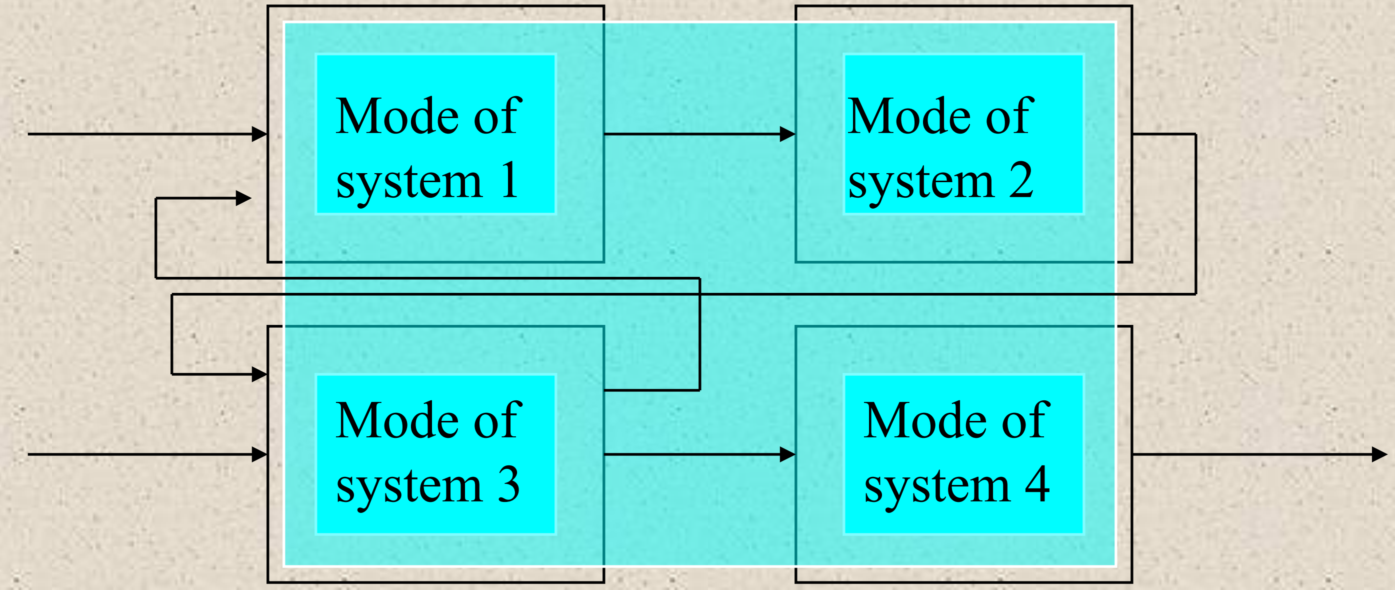
A formalism is required in order to express the way in which modes are invoked and sustained - and how transfer between modes can occur.

Behavioural Mode Implementation

The modes of behaviour of the implementable design solution are required to satisfies the required functional system design (for IOR input/output compliance).

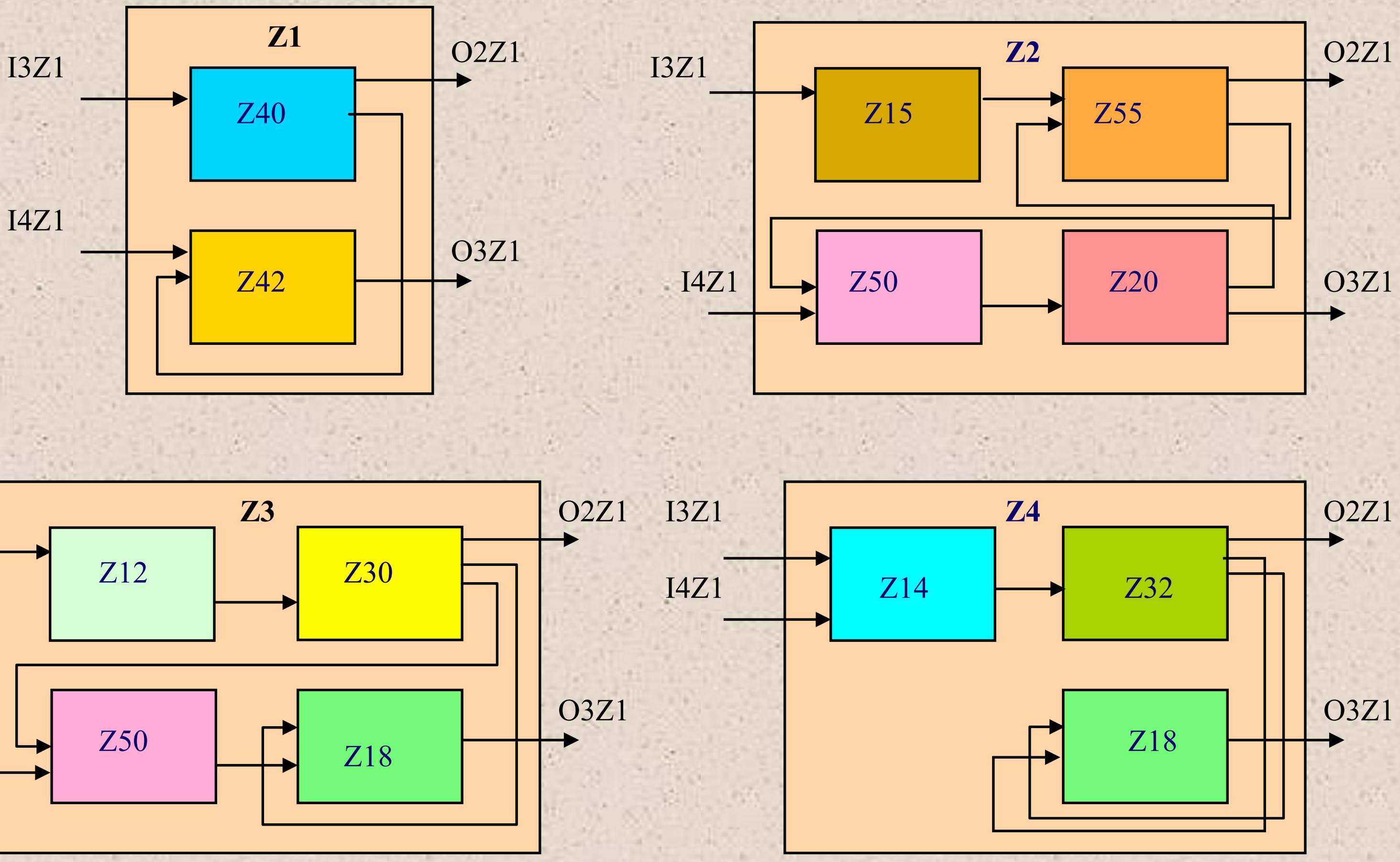
Q. What techniques can be applied to contribute towards a comprehensive proof that this primary objective can be achieved?

Behaviour - Context Dependency

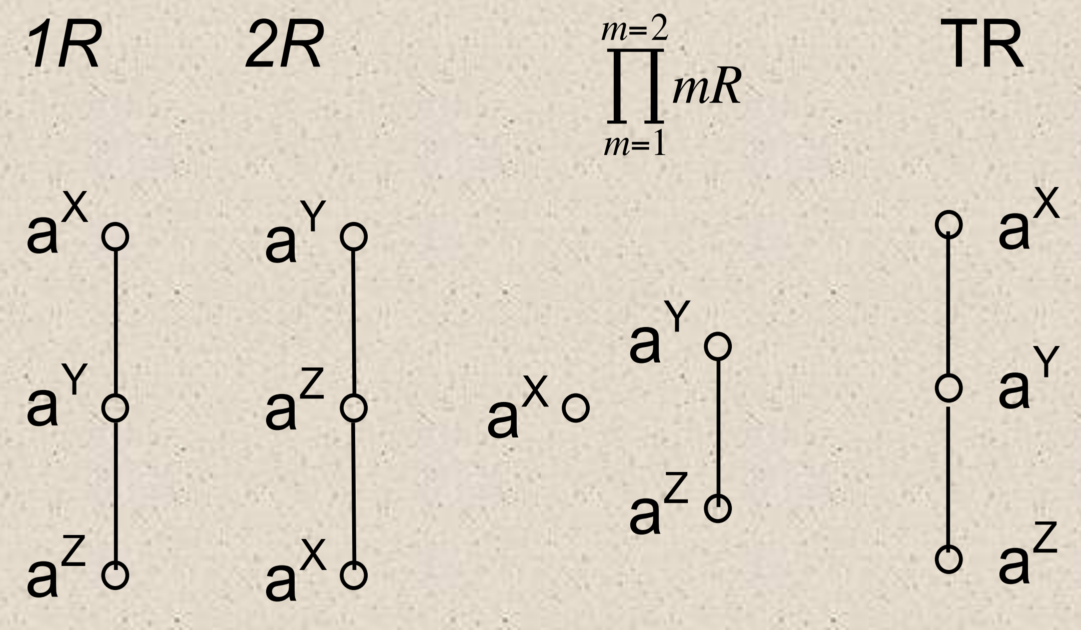


The objective is to ensure that the required *overall* mode of behaviour can be invoked and sustained - by proper implementation of the complex system architecture.

Which Candidate System is <u>Best</u>?



Optimality of a candidate system



For example, a system a with 2 components (n = 2) - the overall trade-off TR is required to be decomposed to the components (for *1R* and *2R*, in this case).

Optimal Systems Design

It is proposed that the reason that complex systems design optimisation is so rarely pursued, as an objective, has little to do with the ‘intractability’ of the problem.

For most practical cases a true optimal solution can be accurately identified, in a reasonable time-scale - even if the problem is perceived to be of type NP-complete (re. the ‘Knapsack Problem’).

Optimality Requirements

We need to start with a clear rationale for the optimal design requirements of a complex system:

- The specific evaluation criteria.
- The trade-off function.
- The implementation constraints.

Evaluation Modelling

Identification of the specific evaluation criteria, and their relative (trade-off: TR) importance, assists the SE team is selecting and implementing the appropriate design models. For example:

- Performance model;
- Cost (LCC, ‘cost of ownership’) models;
- Reliability model;
- Risk model.

The Constraints ‘Problem’

It is the constraints that makes optimisation of complex systems difficult. Without constraints the problem would be ‘trivial’ - we would simply identify the optimal solution for each component according to each component’s trade-off function (determined, for example, by use of the *nk*-STFE formulation).

Optimisation - Context Dependency

The concept of implementation context, and an extensive systems hierarchy, raises important issues, in particular:

- Of properly relating optimisation criteria and trade-off functions between different levels of hierarchy.

- Of allowing for possible combinatorial constraints of component architectures at each level.

Summary

The fundamental issues reviewed within this presentation of theory-based systems engineering method are concerned with the following:

- **Compliant system functionality for <u>complicated</u> system behaviours.**

- **Optimal implementation of a <u>complex</u> system solution.**