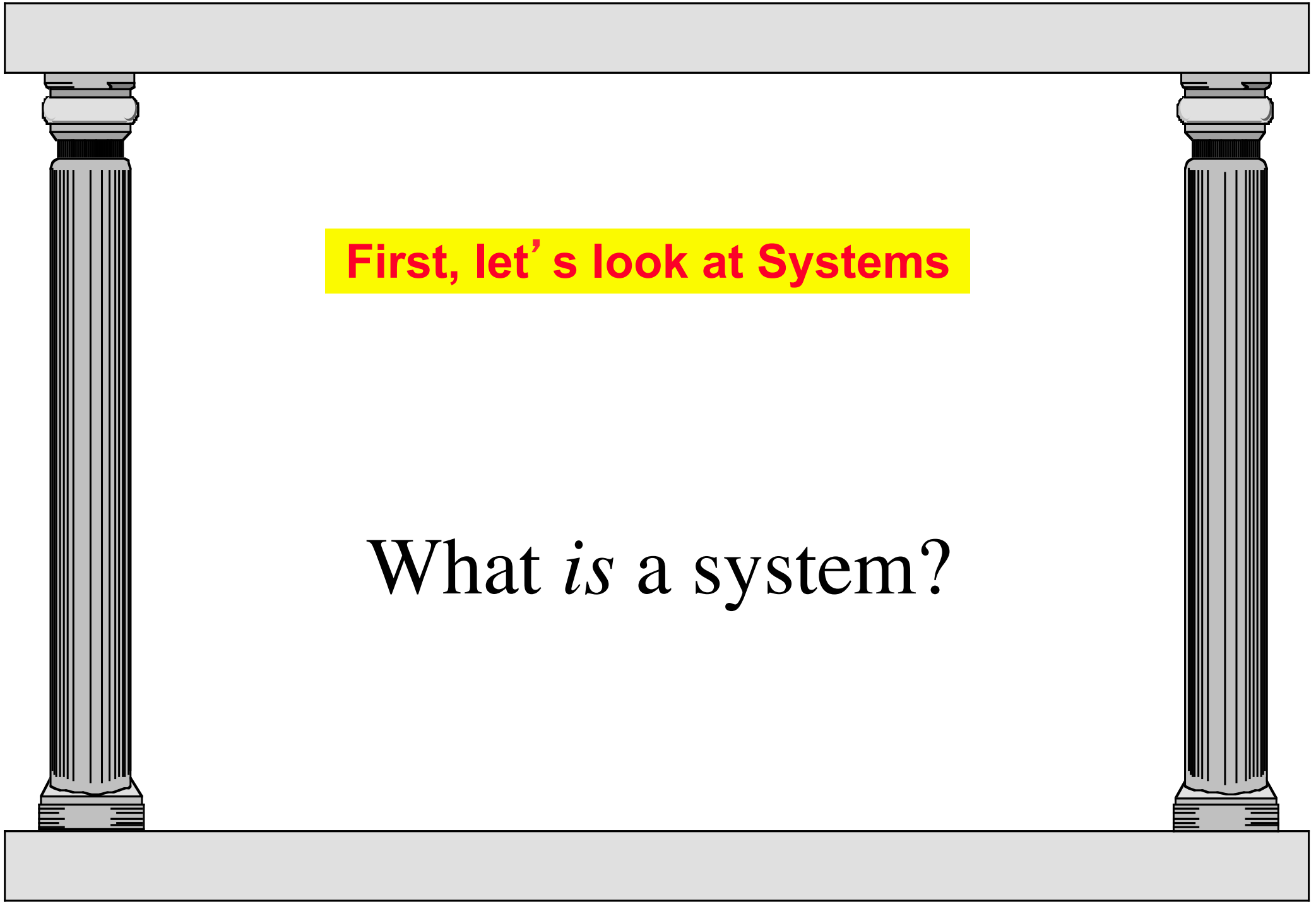
# UK—SE PRINCIPLES, PRACTICES & PROSPECTS



**First, let's look at Systems**

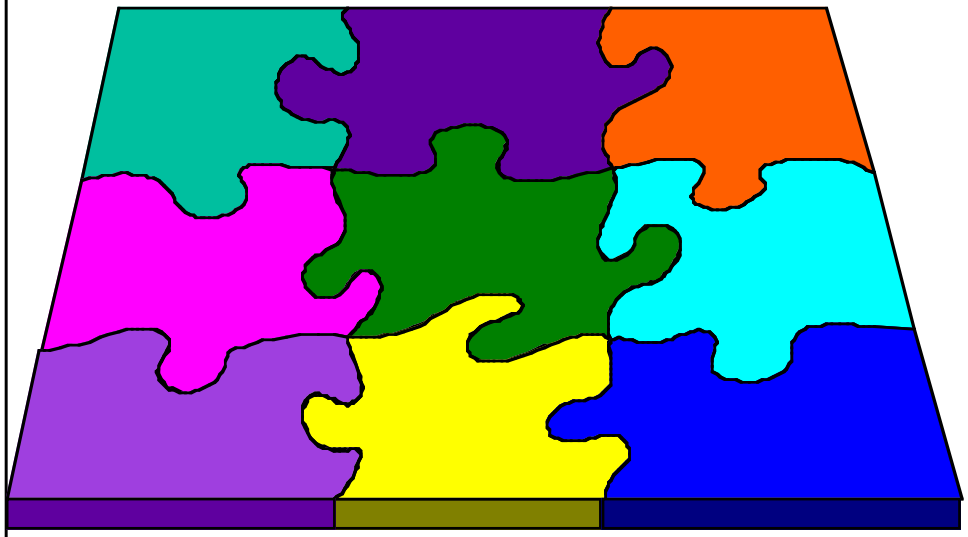
What *is* a system?

# What is a system—INCOSE?

- **A system is an interacting combination of elements viewed in relation to function".**
- Definition excludes solar system—has no function—it just *is*.
- Functional disaggregation of the solar system then meaningless.
- System disaggregation into planetary systems, Jovian, Saturnian, Martian, etc., sensible, simple.
- Should we feel comfortable with any definition that excludes such an obvious example?

# What *is* a System? DKH

- "An open set of *complementary*, *interacting parts* exhibiting properties, capabilities and behaviours emerging both from the parts and from their interactions"



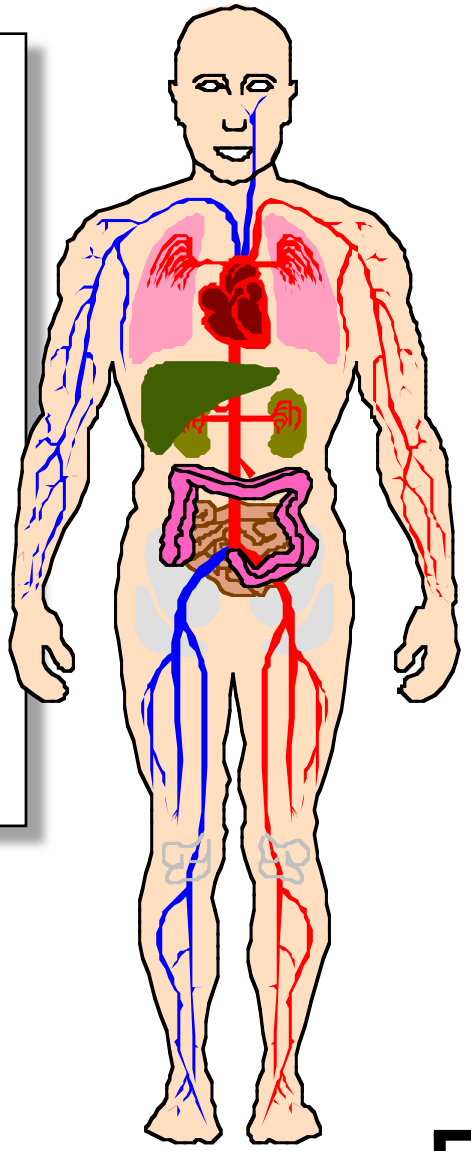
*Complementary Parts, but whole picture emerges only when all the parts are there* ***and*** *in the correct place*

# Systems Ideas

- **Emergence**, defining **Hierarchy**, resulting from…
- …**Interaction**
- **Containment**, nesting, Babushka Russian Dolls
- **Completeness**, yet…
- …**Openness**
- **Complementation**, hence **variety, cohesion, synergy** and…
- …**Dynamic stability**
- **Entropy, internal energy**…hence efficiency, effectiveness, net contribution and quality

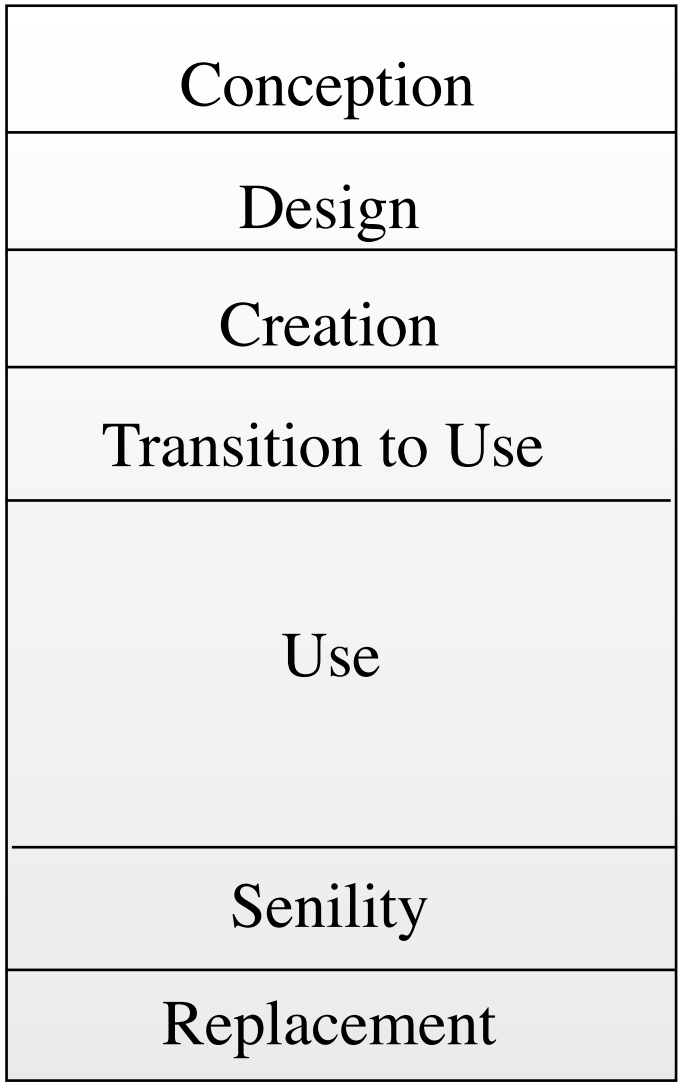
# Whole *System* Principles

- **First Principal of Systems**: —
  - The properties, capabilities and behaviour of a system derive from its parts *and* from interactions between those parts, *and* from interactions with other systems.
- Corollary to the First Principle
  - Altering the properties, capabilities or behaviour of any of the parts, or any of their interactions, affects other parts, the whole system, and interacting systems.



# Whole *Life* Principles

7 Ages of *System*




- Each Age is the Product of Previous Ages.
- Success depends fundamentally on proper Conception and Design for **all** Successive Ages
- (N.B. It is now EU Law to provide for the disposal of systems as part of their conception)


About
Systems
Engineering

# Why should we be interested in Systems Engineering—1?

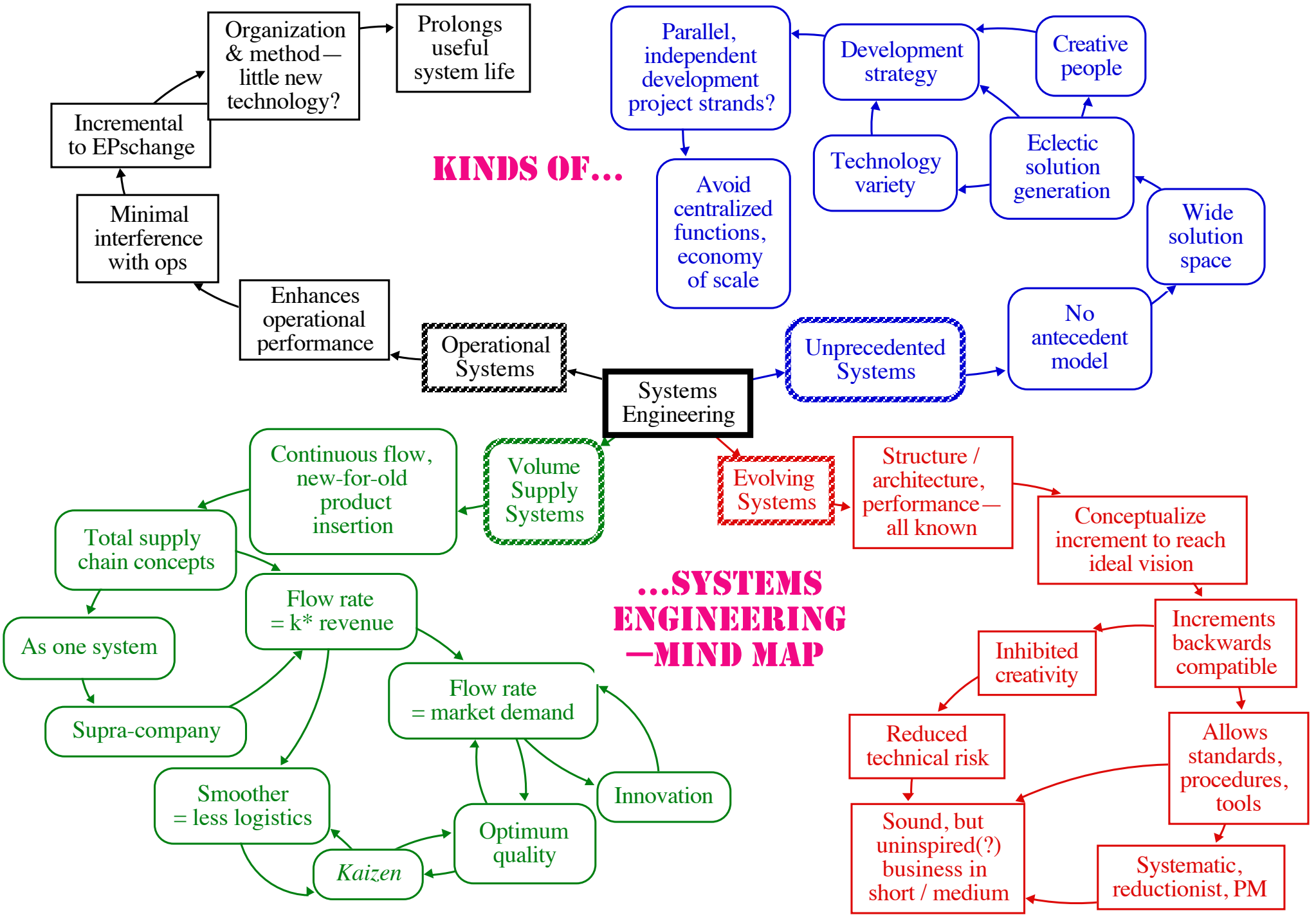
- Fashions sweep through the UK
  - diversification, consolidation, core business, Outsourcing, Market Testing, Total Quality Management, simultaneous/concurrent engineering, Business Process Re-engineering, PFI…
  - can't all be right, mutually interfere, *and* look only at parts of overall system as though separate
  - each new fad heralded as the silver bullet—we really must stop!
- Systems engineering looks at the whole system, does not isolate the parts, is a continuing lifetime process, gets it right!

# Why should we be interested in Systems Engineering—2?

- Grouping many parts into fewer systems allows us to get a grip on complex issues
- The more parts, the better the grip
- So, systems engineering comes into its own just when conventional "break' em apart" ideas fail

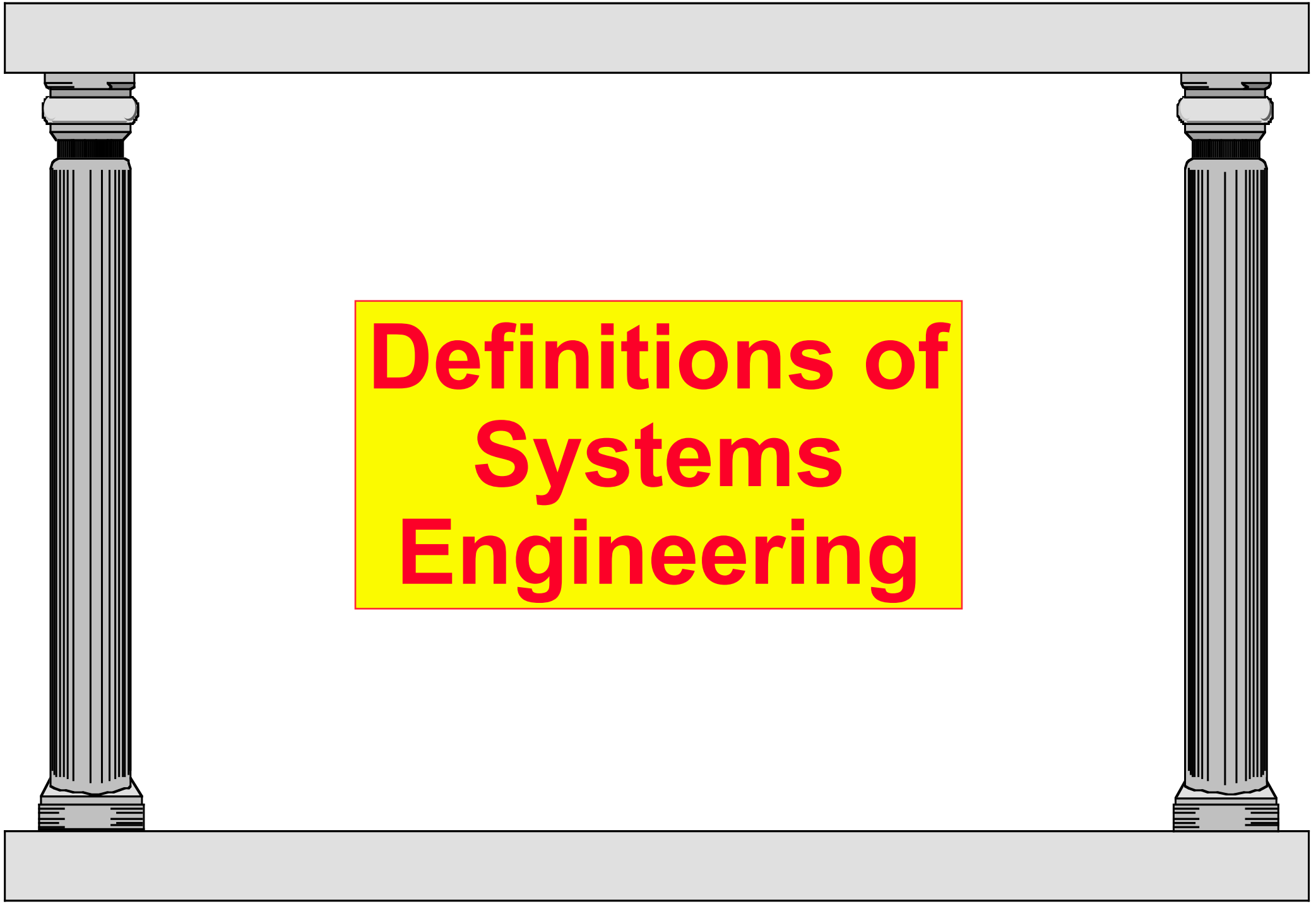
# Why should we be interested in Systems Engineering—3?

- Because understanding the issues seems to be easier for numerate people
- Engineers can see the point of addressing the whole system
  - parts of an orbiting satellite must work synergistically: Each makes contribution to performance, mass, consumption, failure…Each therefore interacts with all others
- *Some* engineers can see that the whole process is also a single system:—
  - parts of a process must work synergistically. Each makes a contribution to product performance, cost and timeliness. Each therefore interacts with all others—for optimum time, cost and quality
  - present practice is to break process into separate , fixed phases


KINDS OF…
Systems Engineering
Operational Systems
Organization & method—little new technology?
Prolongs useful system life
Incremental to EPschange
Minimal interference with ops
Enhances operational performance
Unprecedented Systems
Parallel, independent development project strands?
Development strategy
Creative people
Avoid centralized functions, economy of scale
Technology variety
Eclectic solution generation
Wide solution space
No antecedent model
Volume Supply Systems
Continuous flow, new-for-old product insertion
Total supply chain concepts
As one system
Supra-company
Flow rate = k* revenue
Smoother = less logistics
Kaizen
Optimum quality
Flow rate = market demand
Innovation
…SYSTEMS ENGINEERING —MIND MAP
Evolving Systems
Structure / architecture, performance—all known
Conceptualize increment to reach ideal vision
Increments backwards compatible
Inhibited creativity
Reduced technical risk
Sound, but uninspired(?) business in short / medium
Allows standards, procedures, tools
Systematic, reductionist, PM

# SE Topics

- UK views of SE
- SE Models
  - waterfall, spiral, concurrent, evolutionary acquisition
- Systems Engineering Implementation
  - phasing, specification, configuration, interfaces, budgeting, tradeoffs, integration & test
- Contemporary UK SE Practices
  - seeking the ideal process
  - best practice
- Critique of UK SE practices
  - Procrustean approach—one process, toolset for all systems —prescriptive, predictive
  - reductionist
- A UK-SE Goal-Strategy


Definitions of
Systems
Engineering

# Draft IEE Guide to the Practice of Systems Engineering

- **Systems Engineering**:—"The structured and ordered creation of an Application System to achieve the required *emergent properties*"
- **Aim**:—To establish and deliver an Application System with the *emergent properties* and through-life support facilities required by the customer and satisfying end-user needs
- **Objectives** to achieve the Aim:—
- Create in an ordered, structured manner, an Application System with the emergent properties required by the customer
- Apply structured and ordered processes to the planning and development of the Application, Engineering and Supporting Systems
- Use well-defined procedures and established principles in the development of the Application, Engineering and Supporting Systems

# EnTra Definitions

**Mission of Systems Engineering:—**

- To establish, through a structured approach, an integrated and adaptable system with the current and continuing effectiveness required for customer and user needs

**Key functions:**

- Risk (technical risk and commercial potential)
- Interactive dynamic environmental systems test
- Progressive harmonization
- Systems engineering management—monitoring, adapting and redesign
- Systems architecture
- Configuration management
- System bounding

# MILSTD-499B (Defunct)—Systems Engineering

Systems engineering process operates over eight primary system life cycle functions to define, design and verify system products and processes to satisfy customer needs and requirements:—

1. Hardware 2. Software 3. People 4 Facilities
5. Data 6. Materials 7. Services, and 8 Techniques

*The eight primary* functions *are*:—

1. Development 2. Manufacturing 3. Verification
4. Deployment 5. Operations 6. Support 7. Training, and 8 Disposal

# MILSTD-499B, Systems Engineering

Update was to include:—

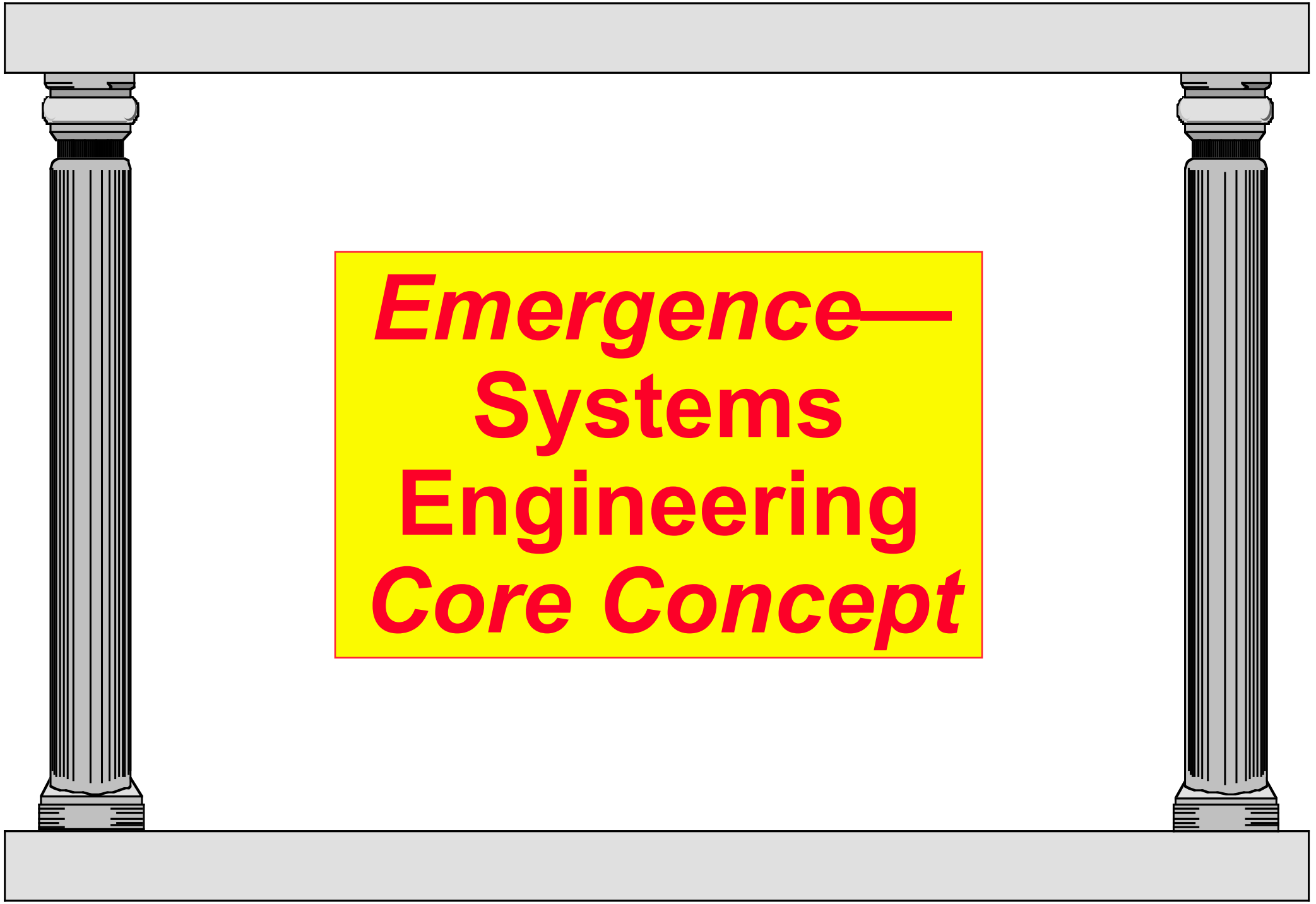
- Translating Identified Needs into Design Requirements, using structured, disciplined application of the systems engineering process into design requirements
- Transitioning Technology from the Technology Base to Product and Process Applications
- Establishing a Technical Risk Management Programme. Risks to be identified, quantified and handled through the acquisition cycle.
- Verifying that the Item Design Meets Established Requirements. The progressive verification of the system from parts, materials, and subprocesses up through the total system products and processes is required

# IEEE1220

- New IEEE Systems Engineering standard
- Drawn up by software engineers
  - shows heritage in invalid functional decomposition, typical of S/W engineering
- Likely to be proposed by US as ISO standard

# Current MOD Definition

- "…the **set of activities**, which **control** the design, implementation and integration of a complex set of interacting components or systems in order to meet the needs of all users and stakeholders within the constraints arising from the systems operational and development environment."
- According to this, then,
  - MOD views SE is a means of control(?) {Control leads to reduction—the antithesis of systems engineering e.g. Downey}
  - the product may be a system - or not - while the process does not appear to be a system, just a set?
  - SE doesn't have any lifecycle responsibility?
    - » i.e. ends at integration?
  - SE is not concerned with manufacture/assembly?
- *Read definitions carefully—they are full of implications. Read what they do* **not** *say, too!*
- Above definition describes MOD programme management


Emergence—
Systems
Engineering
Core Concept

# The Core Concept of *Emergence*

- Complex systems
  - Consist of many, varied interacting parts
  - Properties of whole system are difficult/impossible to predict
  - Cannot be created piecemealwith any hope of meeting requirement
- Complex systems exhibit emergent properties
  - properties of the whole system, not exclusively attributable to the parts
- Performance, Availability, Survivability
- (In)compatibility
- Whole system phenomena; C of G, M of I, Entropy

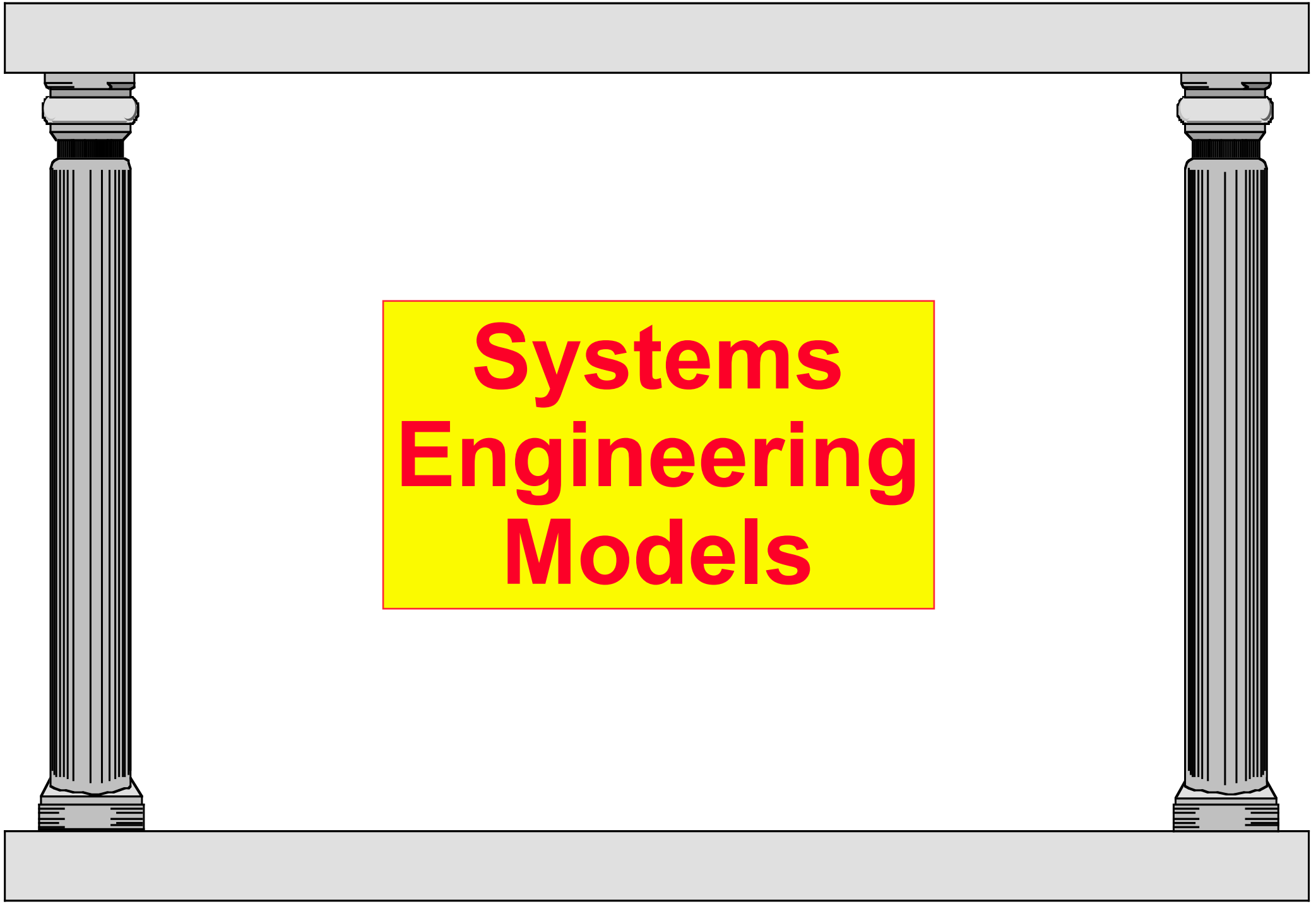
# The Central Concept of *Emergence*

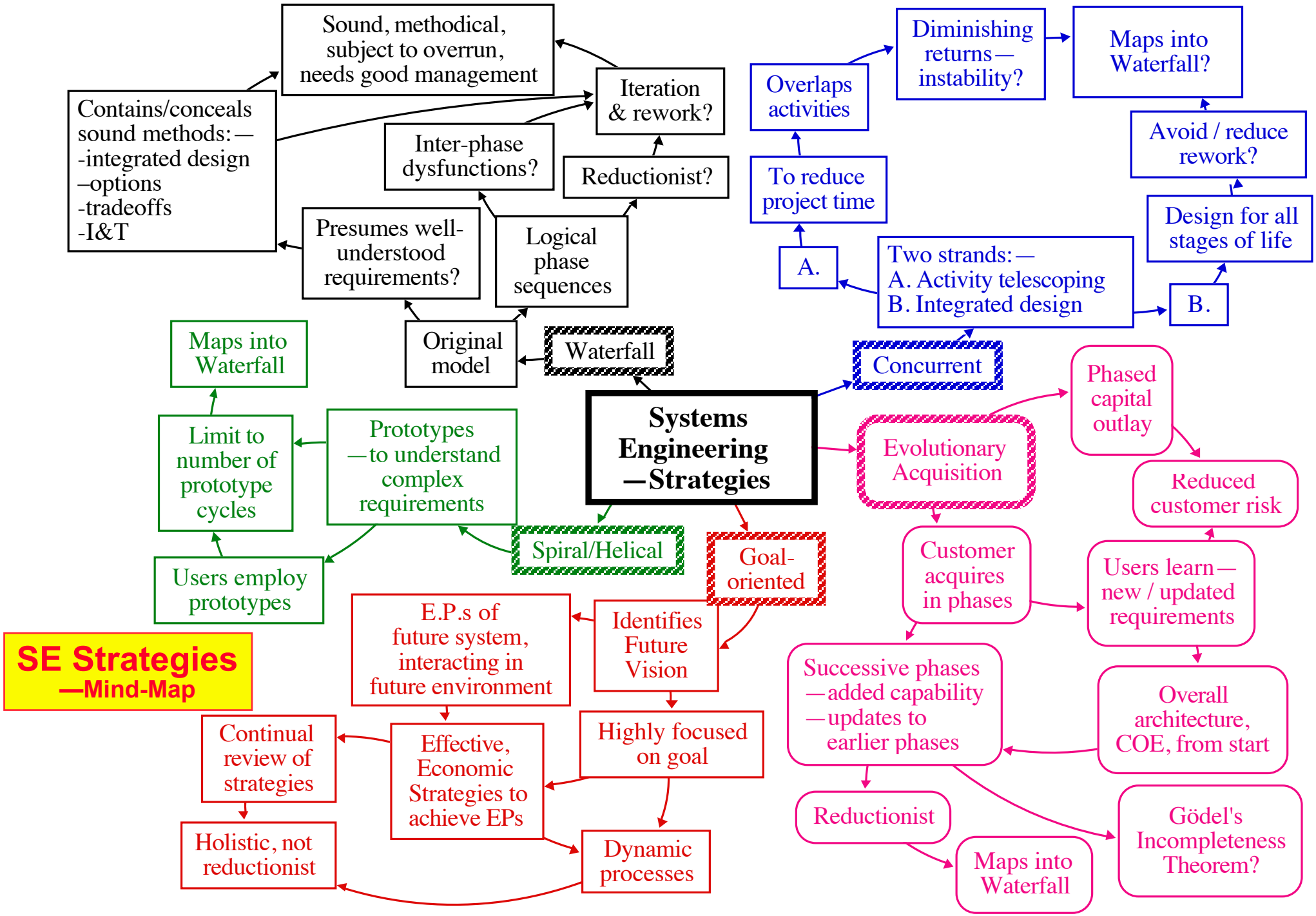
***Those properties of a system as a whole which cannot be ascribed completely to its individual component parts, e.g. self awareness from the brain, a picture emerging from a jg-saw puzzle.***

- *Physical* properties, such as coefficient of drag, emissions, recycle-ability
- *Functional* properties, such as carrying capacity, top speed, braking distance
- *Temporal* properties, those that change or remain constant, such as rust-resistance, times between servicing, lifetime
- *Aesthetic* properties, such as elegance, comfort, style
- *Behavioural* properties, such as responsiveness, handling, ride.
- *Value* properties, such as cost, litres per kilometre, resource utilization

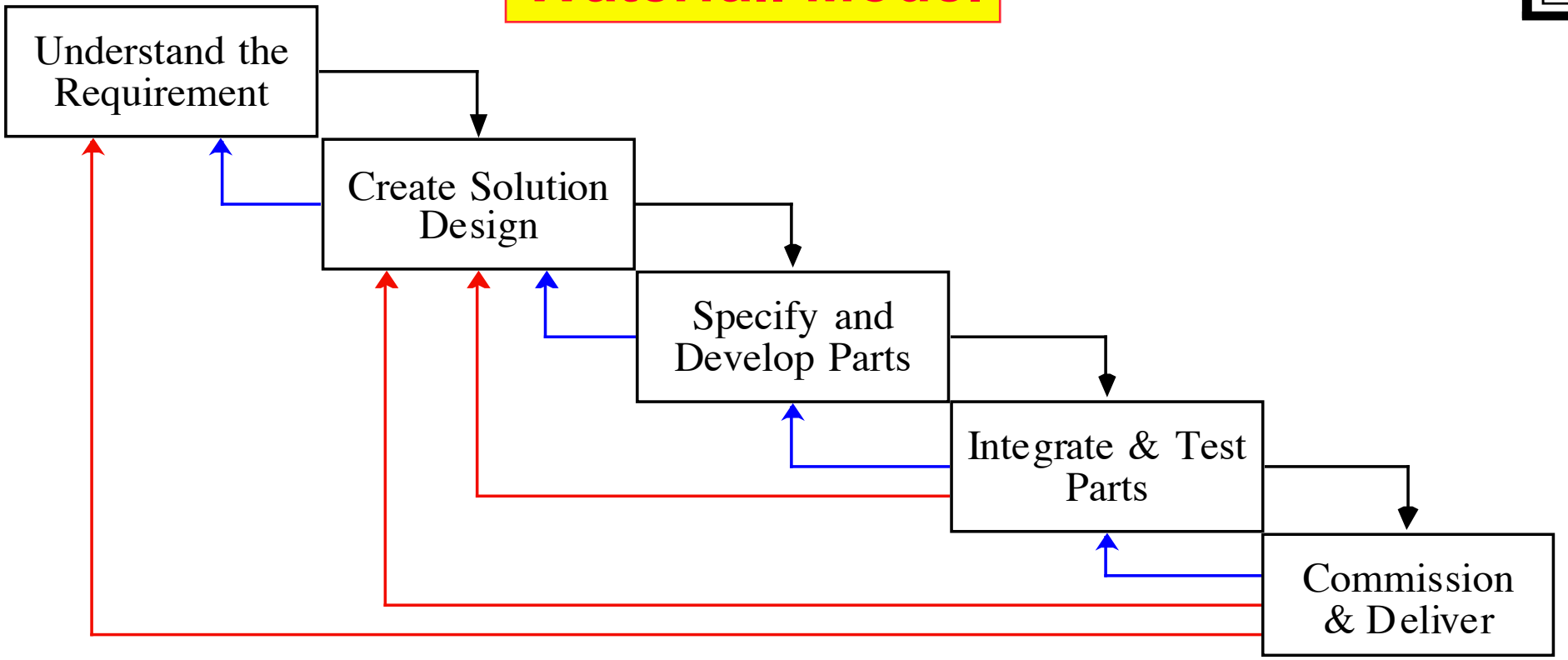
# Systems Engineering—Guardian of Emergent Properties

- **Identify** requisite emergent properties
- **Create** systems with those requisite **emergent properties**
- **Diminish** undesirable emergent properties
- Create **support** systems so that the customer and user may maintain/ evolve the requisite emergent properties **throughout** the system's operational **life**


Systems
Engineering
Models

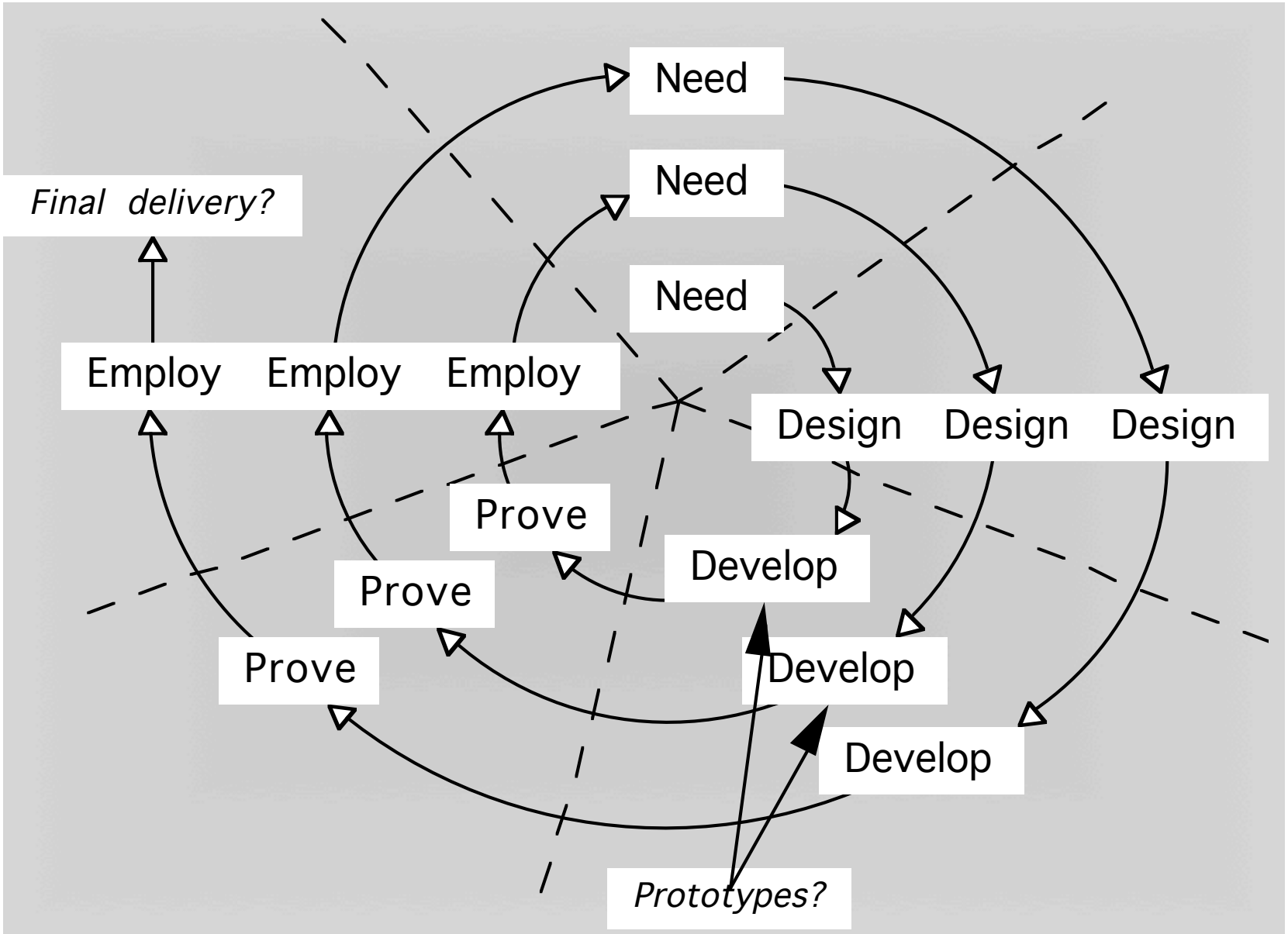

SE Strategies
—Mind-Map
Systems Engineering —Strategies
Waterfall
Original model
Sound, methodical, subject to overrun, needs good management
Iteration & rework?
Contains/conceals sound methods:— -integrated design –options -tradeoffs -I&T
Inter-phase dysfunctions?
Reductionist?
Presumes well-understood requirements?
Logical phase sequences
Maps into Waterfall
Limit to number of prototype cycles
Prototypes —to understand complex requirements
Users employ prototypes
Spiral/Helical
Goal-oriented
E.P.s of future system, interacting in future environment
Identifies Future Vision
Continual review of strategies
Effective, Economic Strategies to achieve EPs
Highly focused on goal
Holistic, not reductionist
Dynamic processes
Concurrent
Overlaps activities
Diminishing returns— instability?
Maps into Waterfall?
To reduce project time
Avoid / reduce rework?
A.
Two strands:— A. Activity telescoping B. Integrated design
Design for all stages of life
B.
Evolutionary Acquisition
Phased capital outlay
Reduced customer risk
Customer acquires in phases
Users learn— new / updated requirements
Successive phases —added capability —updates to earlier phases
Overall architecture, COE, from start
Reductionist
Maps into Waterfall
Gödel's Incompleteness Theorem?

# Waterfall Model

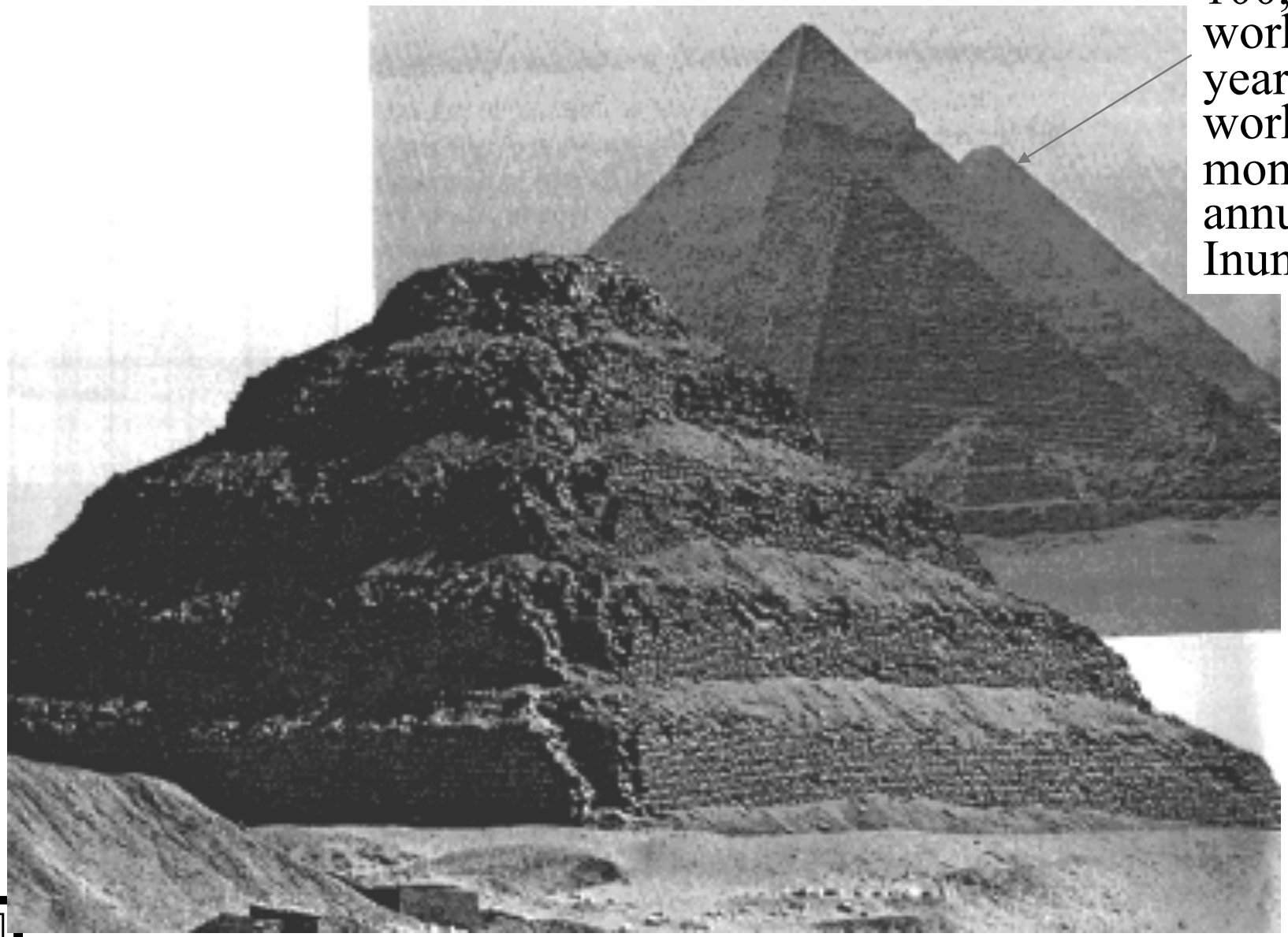



- Typical waterfall model—simplified
- Blue feedback lines—wasteful, time consuming, expensive
- Red feedback lines—disaster
- Any feedback creates "eddies" in workflow—*organizational entropy*

# Managing Complexity—Helical Model



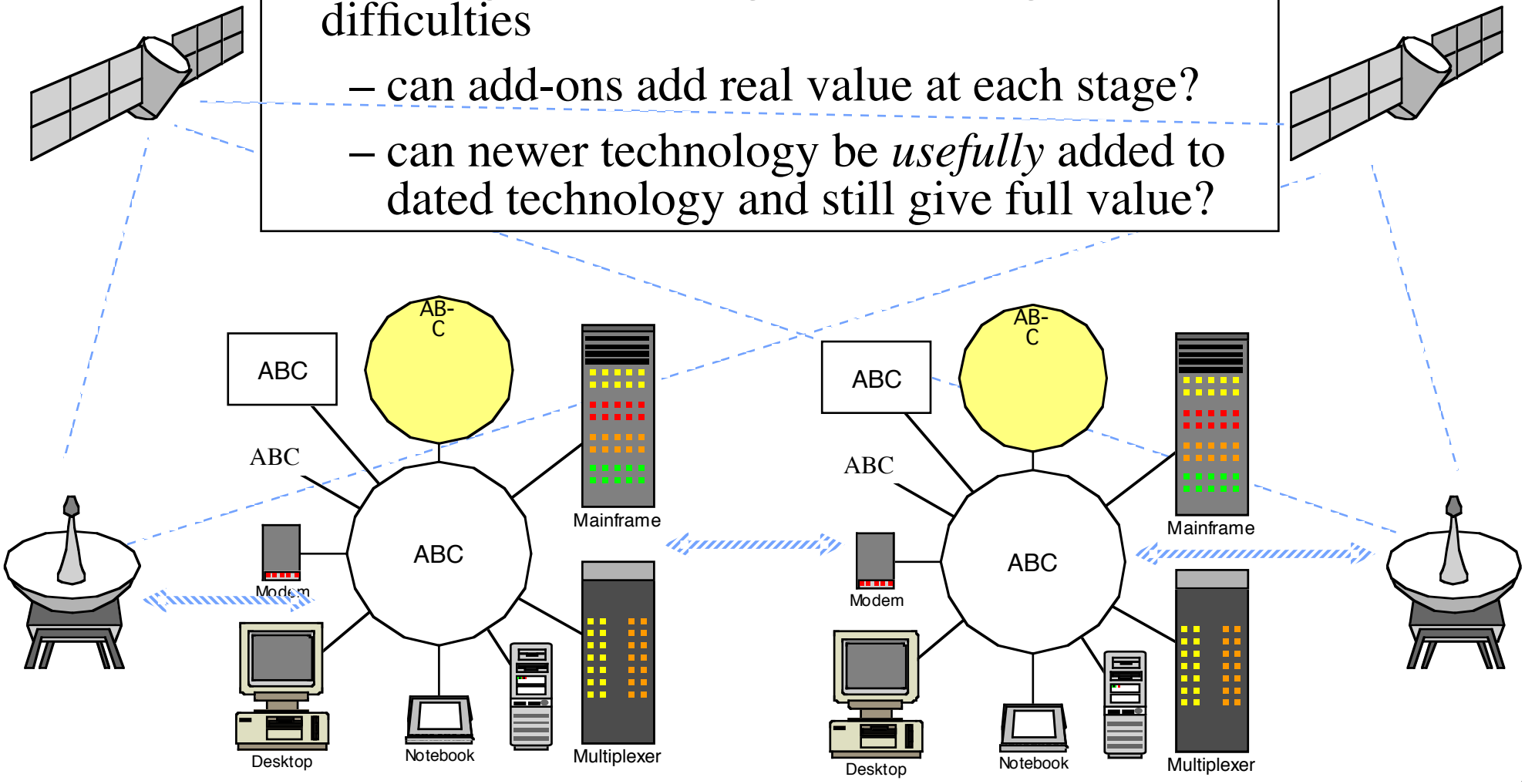
# Prototyping isn't new, either…



100,000 workers, 20 years, working 3 months per annum during Inundation

# Evolutionary Acquisition

- When a system becomes very complex / expensive, customers may seek evolutionary acquisition
- Sounds good, can be good, but fraught with difficulties
  - can add-ons add real value at each stage?
  - can newer technology be *usefully* added to dated technology and still give full value?

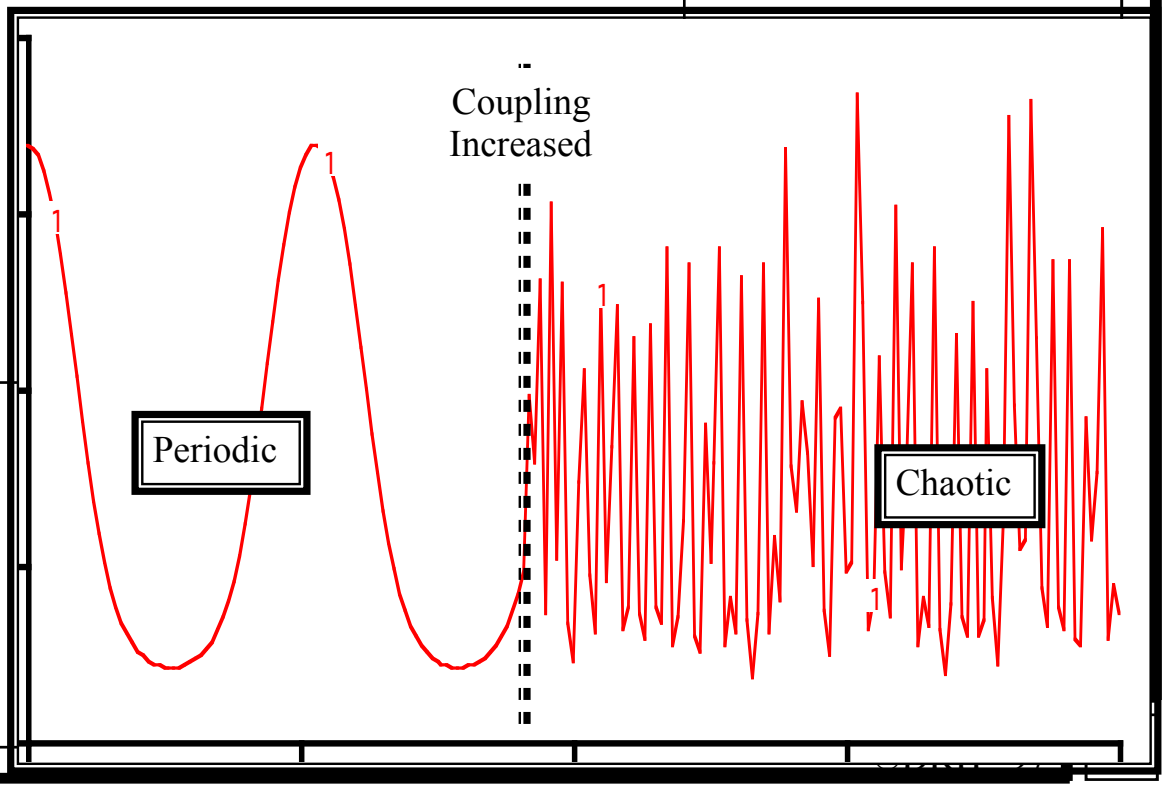


# Simultaneous/Concurrent

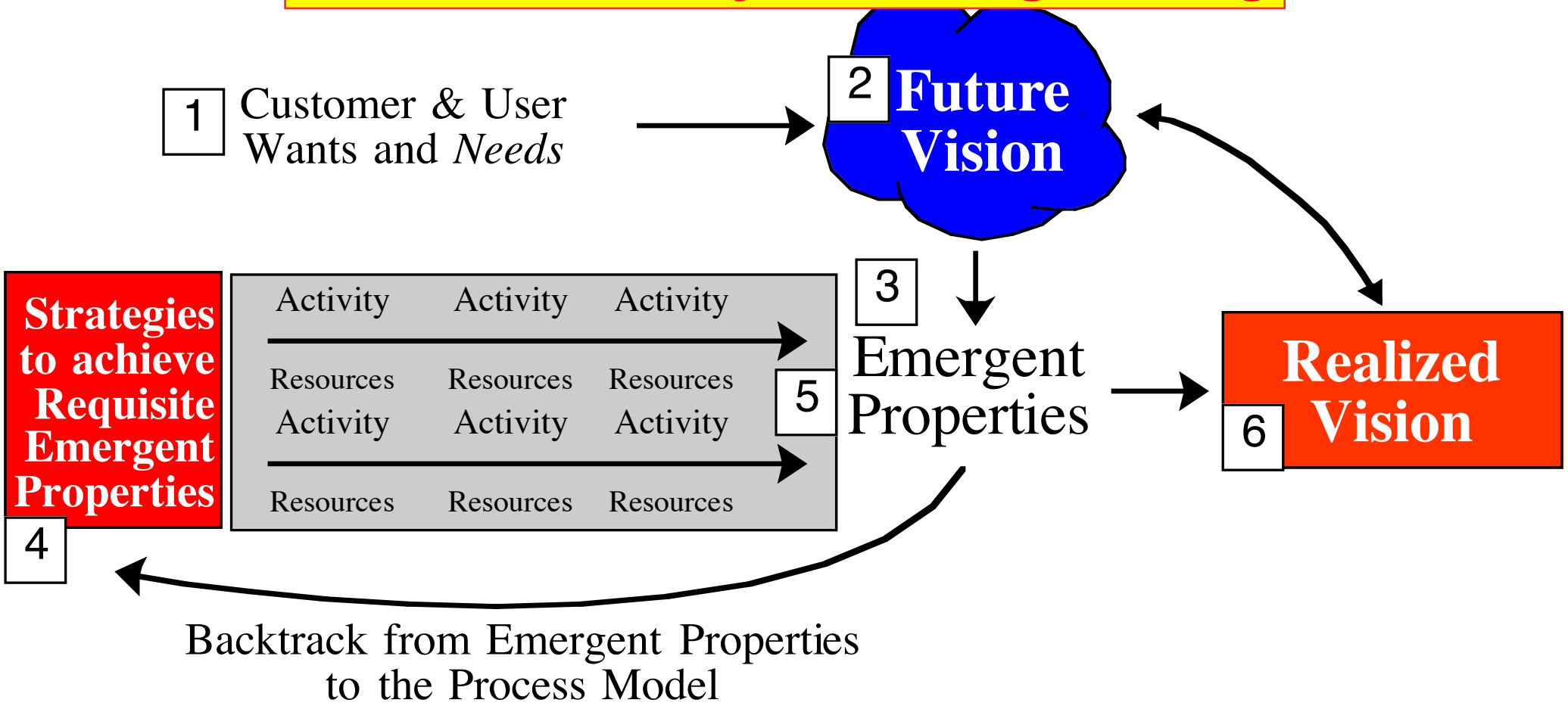
- Disguised waterfall, using an eclectic collection of techniques
- Two main themes, both seeking reduced time to market:
  —
  - 1. Telescope sequential activities
  - 2. Team design
    - » contributions from development, manufacture, integration, commissioning, installation, operation and maintenance
- Item 2 *de facto* approach to waterfall since the 50s. Basis of multi-disciplinary SE

# (UK) Concurrent / Simultaneous Engineering

- Contemporary bandwagon
- Design teams comprised of production, development, comissioning, etc., engineers to ensure robust design
  - reduce risk of rework
  - conventional part of waterfall
- Tasks compressed/overlapped
  - reduce time to market
  - diminishing returns
  - increased entropy
  - fragility & risk



# Goal-Oriented Systems Engineering

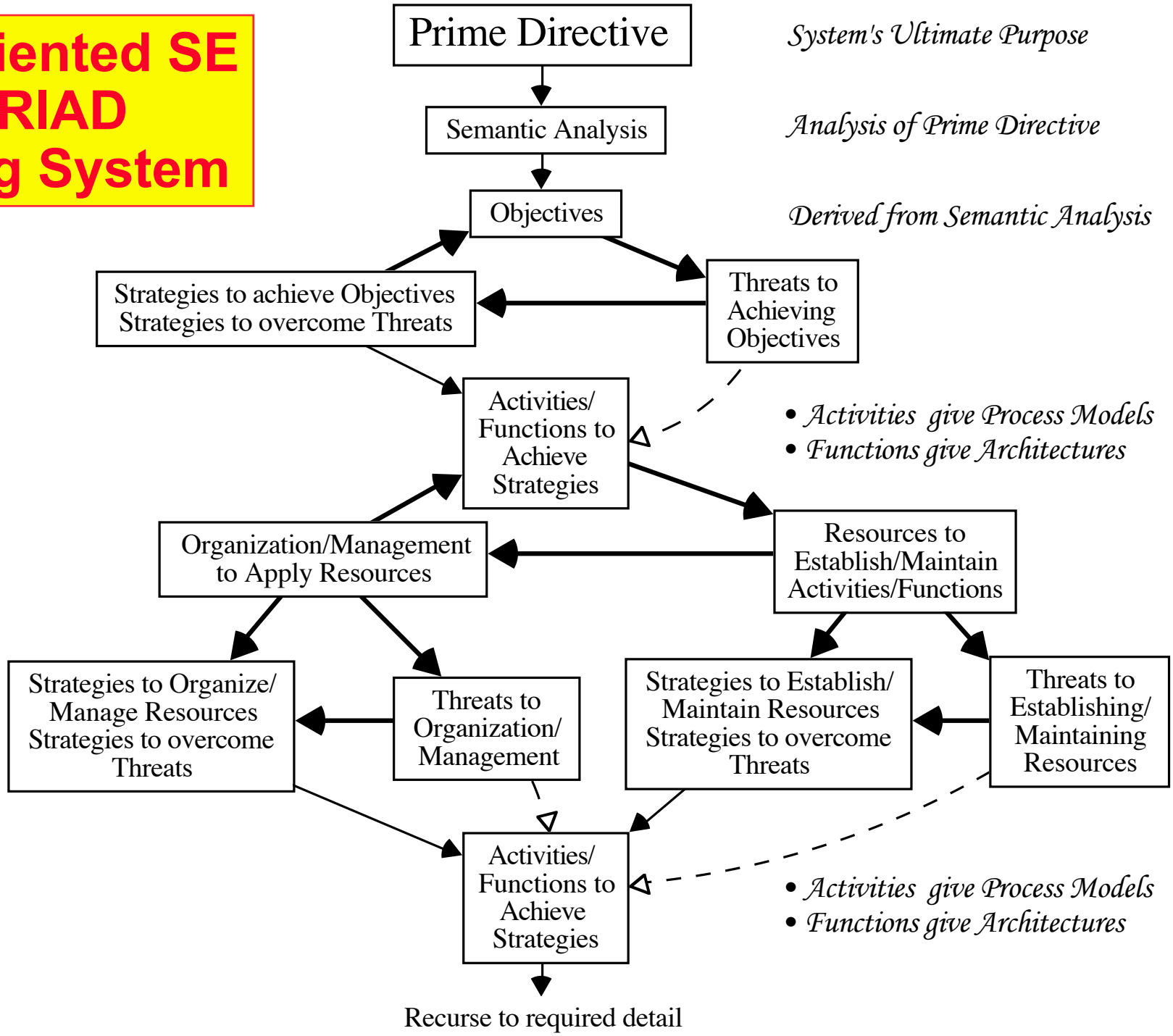


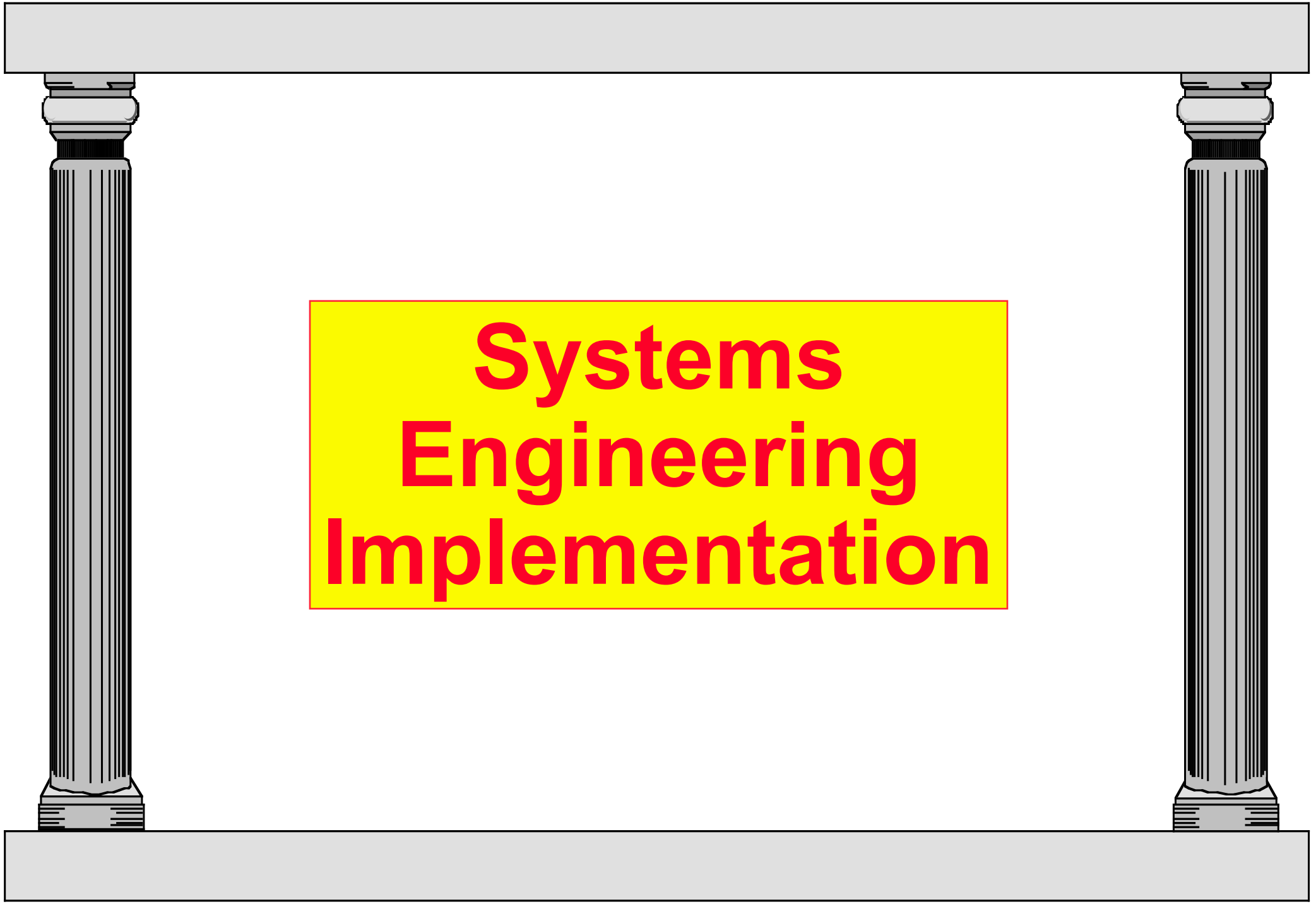

1. **Establish Customer and User Wants and Needs.**
2. **Conceive Future Vision with Customer.**
3. **Establish the Emergent Properties of the Future Vision**
4. **Conceive Strategies to Realize Requisite Emergent Properties**
5. **Select, Correlate, Resource and Pursue Effective Strategies to Realize Requisite Emergent Properties**
6. **Realize Future Vision**

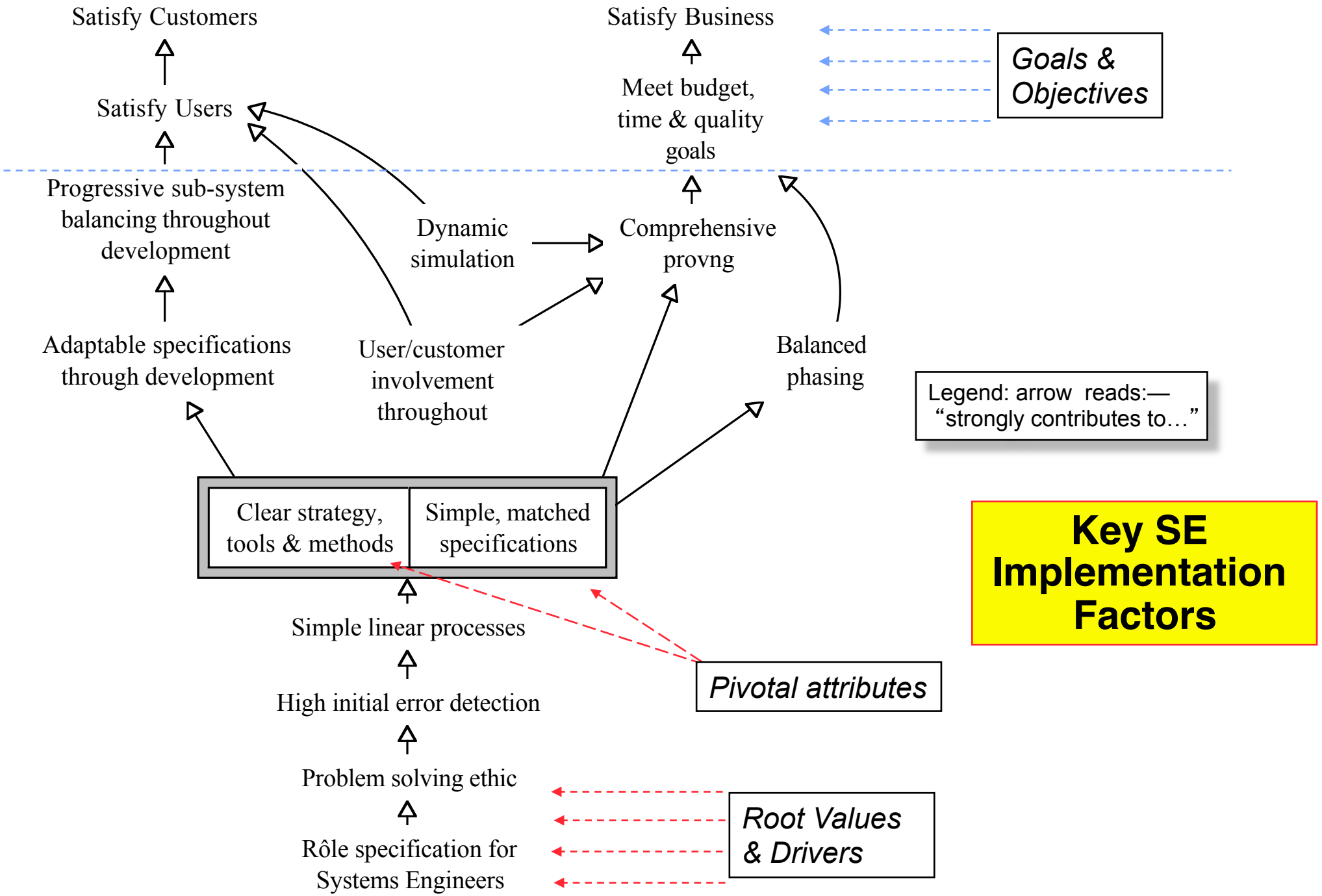
# Goal-oriented Systems Engineering

- Focused on Future Vision and its EPs
- Inspires, leads through shared vision, does not control
- Holistic, true SE
- Avoids premature cost limitation, which emasculates solution
- EP strategies—effective first, economic second
  - effective more important than economic
    - » "ineffective" does not work, wastes time *and* money
- Correlating, harmonizing various strategies for different EPs
  - crucial to success
  - may require reappraisal of selected strategies
- Ideal for Unprecedented Systems and for managing development as parallel, semi-autonomous projects

# Goal-oriented SE —TRIAD Building System




Systems
Engineering
Implementation

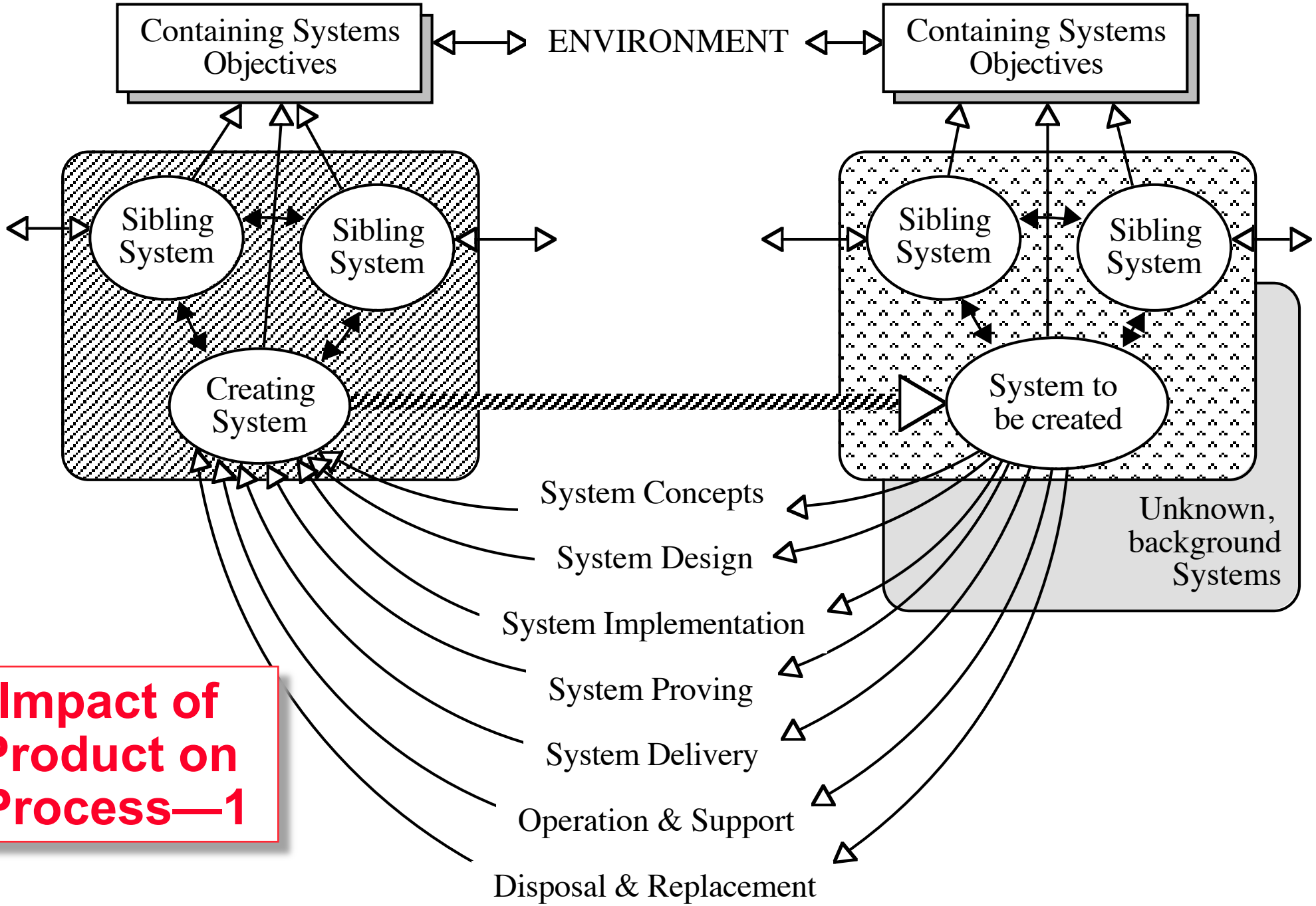

Satisfy Customers
Satisfy Users
Progressive sub-system balancing throughout development
Adaptable specifications through development
Satisfy Business
Meet budget, time & quality goals
Goals & Objectives
Dynamic simulation
Comprehensive provng
User/customer involvement throughout
Balanced phasing
Legend: arrow reads:— "strongly contributes to…"
Clear strategy, tools & methods
Simple, matched specifications
Simple linear processes
High initial error detection
Problem solving ethic
Rôle specification for Systems Engineers
Pivotal attributes
Root Values & Drivers
Key SE Implementation Factors

# Managing System Design/Engineering

- Managing the process is key
- Project management
  - Time, Budget, quality
- Systems Engineering Management
  - *How* to do the job in the time, to budget and to achieve quality
  - Planning and Management of :—
    - » Requirements capture
    - » Design—all phases from development to disposal
    - » Partitioning, interface control, specification
    - » Harmonisation of developing parts
    - » Progressive integration and test
    - » Trials, commissioning and installation
    - » In-service operation and support
    - » De-commissioning and recycling

# Systems Engineering Ethic

- ***Integrity*** — honesty, openness and thoroughness
- ***Quality*** — customer and user satisfaction, but a sense of excellence too.
- ***Manageability*** — methods and tools for planning, work decomposition and control to enhance effectiveness, improve efficiency and adapt
- ***Risk Control*** — identifying, categorizing, reducing, avoiding, anticipating, even accepting, but with knowledge of the degree of risk
- ***Holism*** — addressing the whole task, establishing balanced completeness in design, development, implementation, operation and disposal
- ***Intelligence*** — learning, perhaps from mistakes, applying the knowledge to new and changing circumstances so as to continually improve performance

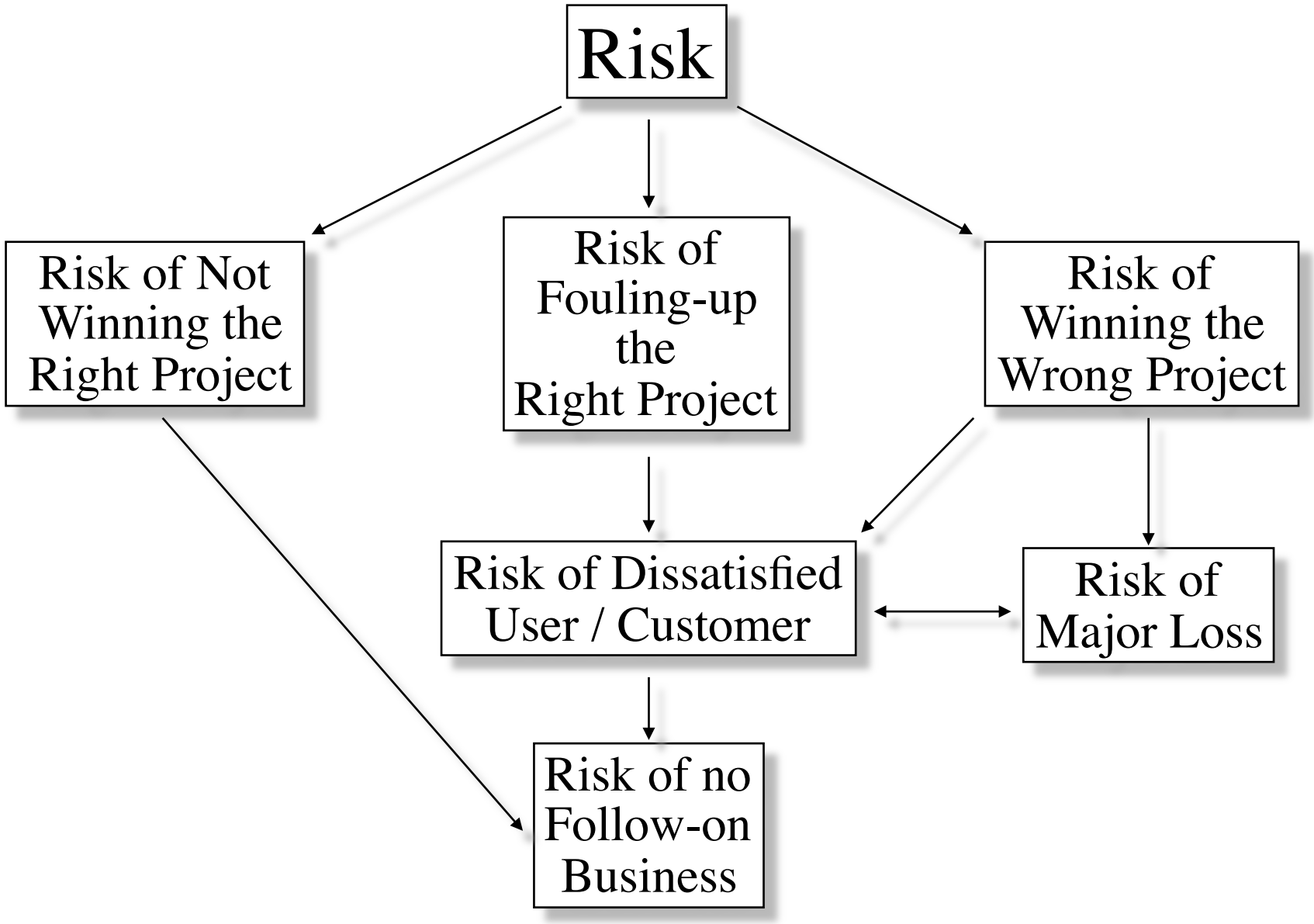



# Impact of Product on Process—1

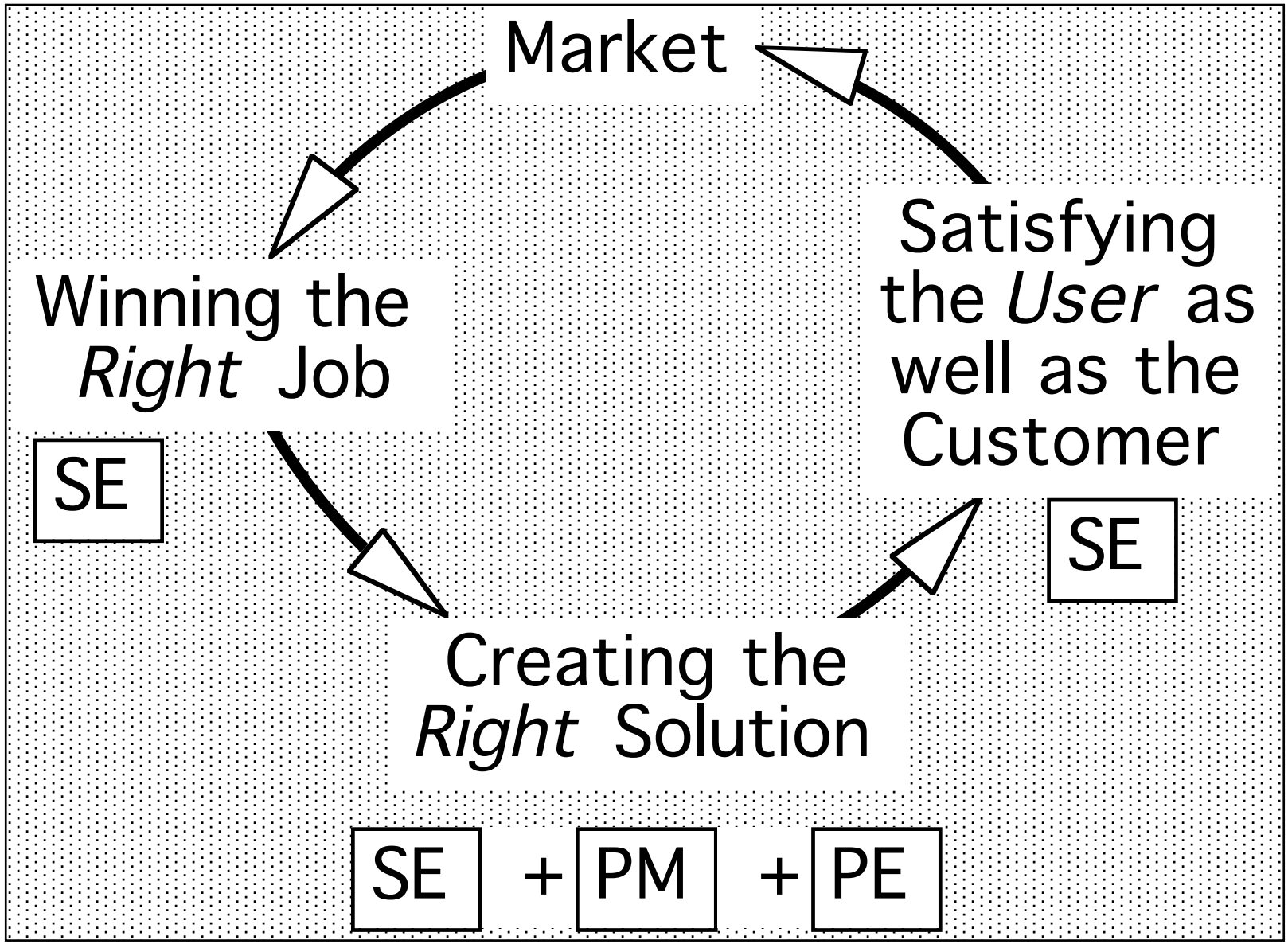
# Impact of Product on Process—2

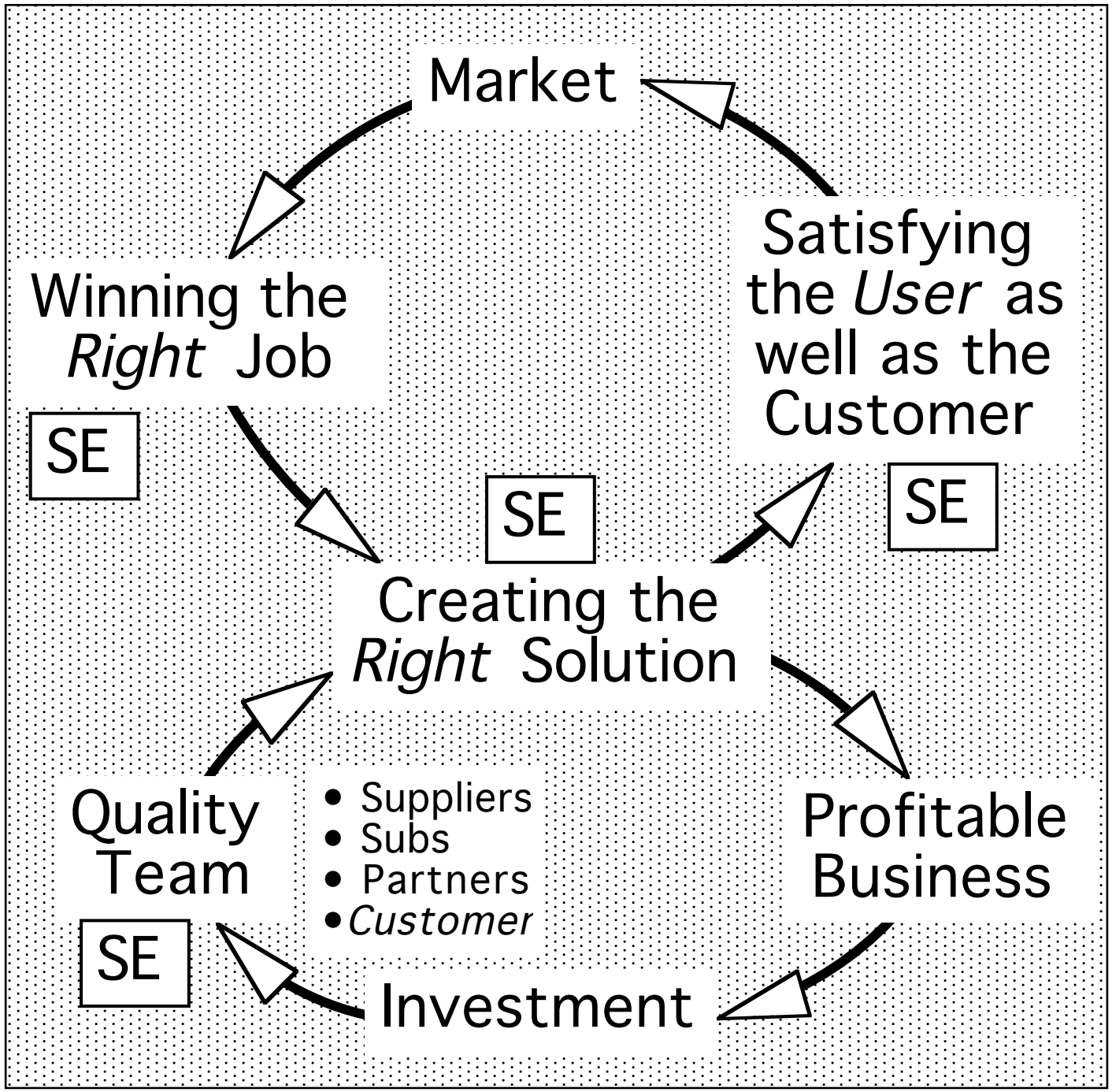
- It follows from the diagram that:—
  - product environment influences nature of product and its support
  - product determines (should determine?) process
  - for bespoke products:—
    - » notion of single, standard SE process with standard tools and procedures is untenable
  - for innovative and creative new products, or existing products in new markets:—
    - » notion is highly restrictive
- A feature of current UK SE thinking is the search for standard approaches—driven by reductionism

# Management of Risk—What Risk?



# Where should SE Fit in?





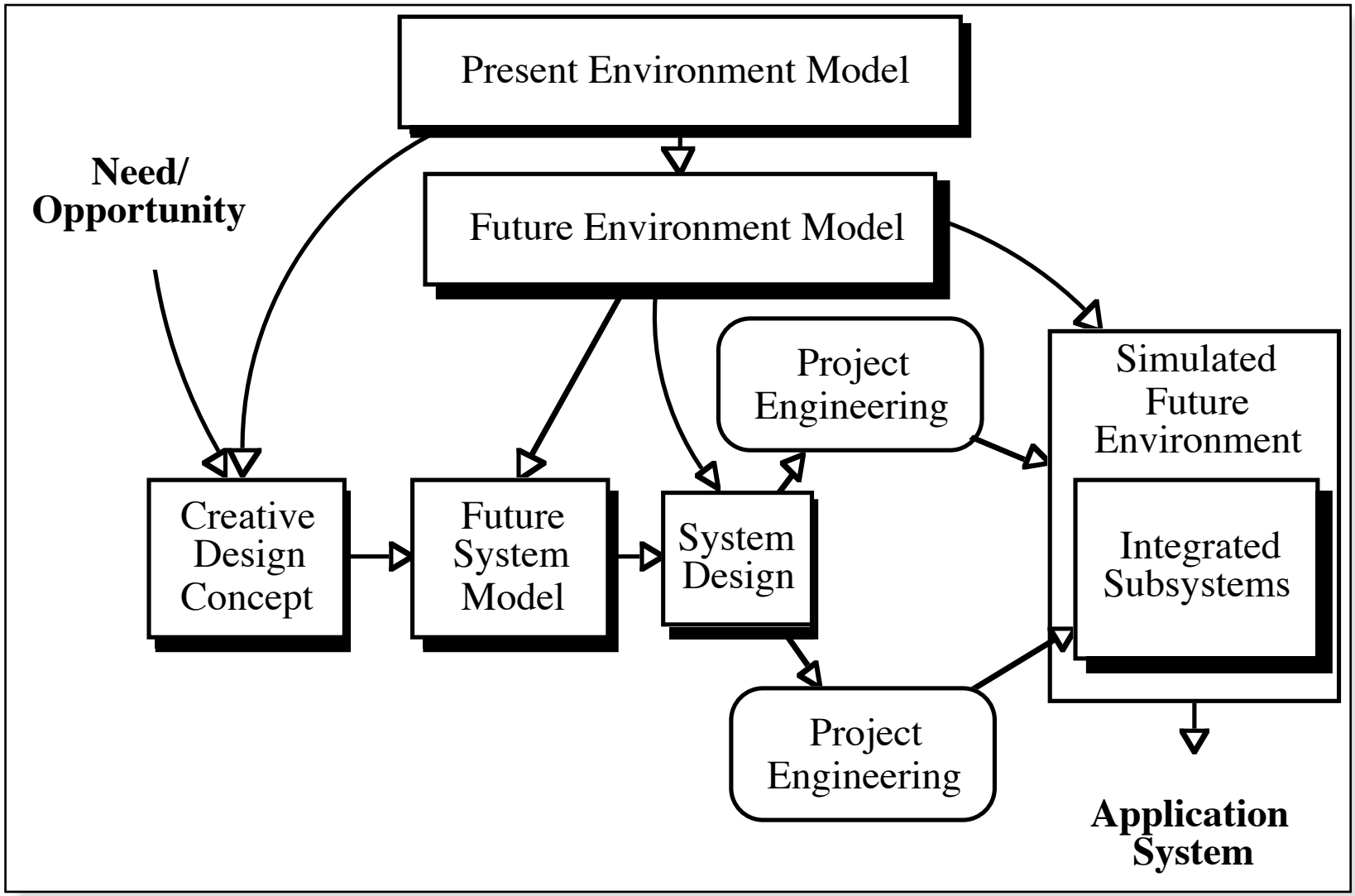

SE = Systems Engineering

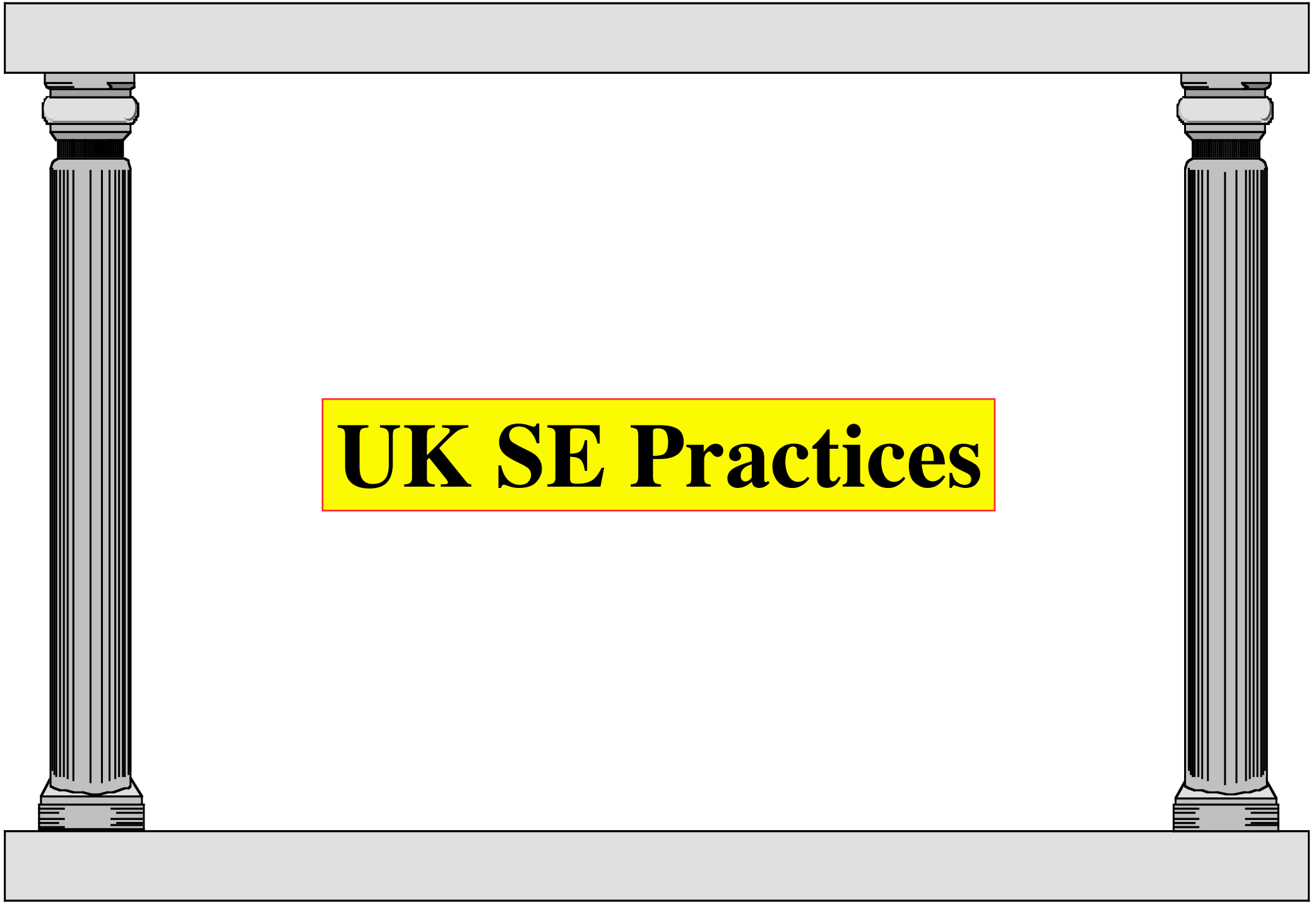
**And Good Systems Engineering includes the Subs and Customers, too**

# Proving

- Proving the system comprises:—
  - Progressive Test and Integration of Parts
    - » harnesses, jigs, man-in-the-loop, representations of missing parts
  - Test of the whole in a simulated environment
    - » static—test at a number of set points
    - » dynamic—test with many/all parameters varying simultaneously
    - » beyond design envelope
  - Trials
    - » live trials under prescribed, monitored conditions (e.g. live firings)
    - » operational trials

# Anticipating the Future




UK SE Practices

# Systems Engineering—the Principles of Creativity

Highest level of abstraction

Breadth before depth

Level at a time

Disciplined anarchy

Decomposition before integration

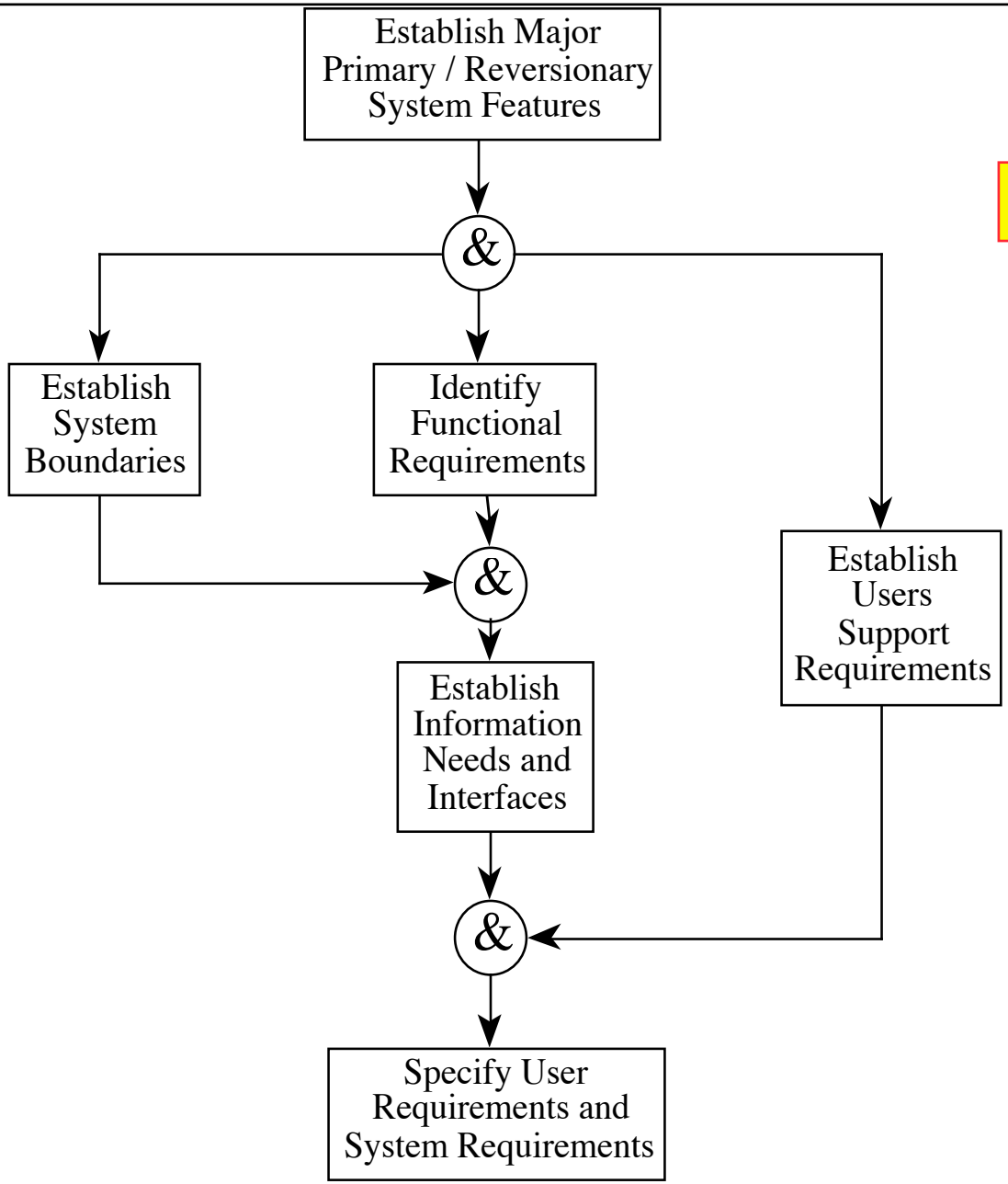
Functional before physical

Tight functional binding

Loose functional coupling

Functional migrates to physical

# Work Breakdown



# Partitioning Criteria

- Progressive integration of parts into the whole
- Testability
- Reliability
- Maintainability
- Safety
- Ease of manufacture
- Availability of existing sub-systems
- Phased delivery
- Packaging

# Configuration Management

**Controls fit, form and function**

- Interfaces
- Safety
- Records

**Configuration management involves:—**

- Identification
- Control of change to system, sub-system, interface, protocol and documentation
- Change accounting, to track changes
- Audit, to check real world against records

**Configuration change management ensures:—**

- Full analysis, design, impact and costing of changes is effected
- Only authorized changes are executed
- Interfaces are controlled

# Interface Control

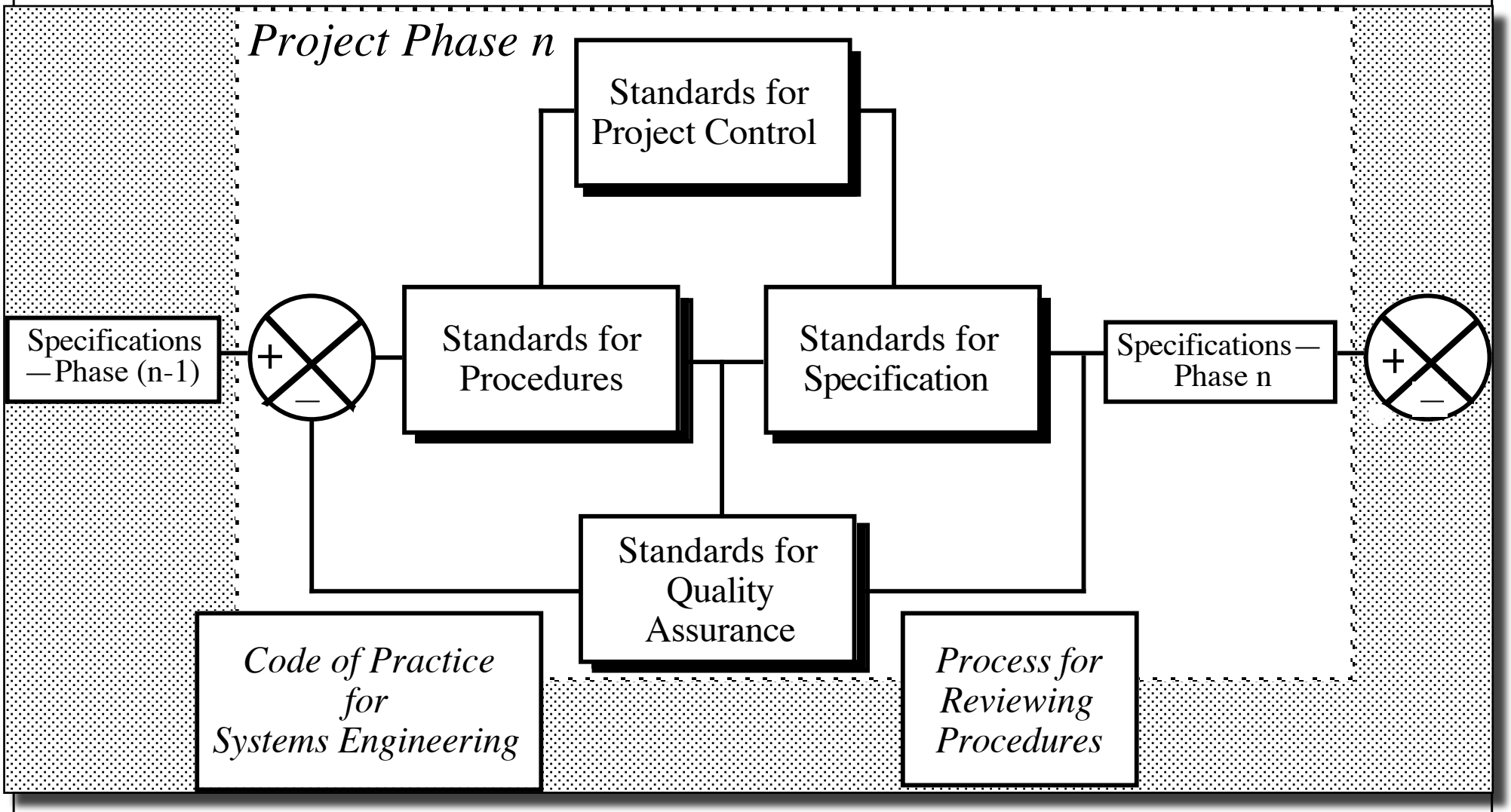
**Interface control identifies:—**

- The relevant systems and sub-systems
- The strategy for maintaining interface compatibility, e.g. use established interface & interchange protocols

**…and follows up with…**

- Co-ordination of the changes
- Documentation

# Standards and Procedures



# Specifications

- Definitions, standards and specifications applicable
- Interfaces
- Protocols
- Partitions
- Environment for use, storage and transportation
- Service life
- Shelf life
- Reliability
- Maintenance, tools, procedures, handbooks, spares, test facilities
- Size, weight/mass, power consumption, dissipation
- Appearance, finish, etc.
- Human factors in handling and operation

# Note on *ab initio* Specification Procedure

- Current vogue is to specify a boundary, interfaces crossing the boundary, internal parts and their interfaces—introspective
- Powerful alternative—***look outward***:—
  - ***avoid*** specifying a boundary *at the start*
  - identify what a new system is to *do*
  - identify how the new system will perturb other systems in their mutual environment *and* how *they* will react to *it*!
  - add additional systems to neutralize *un*wanted effects, augment *wanted* effects
  - then, and only then, new boundary establishes itself
  - outward looking, harmonized with future environment, *and* better business!
- Simple perspective shift—fundamental improvement!

| | Systems requirement | Systems design & development | Project control | Quality assurance |
|---|---|---|---|---|
| Standards for specification of: | Boundaries<br>Objectives<br>Performance<br>User's organization<br>Connectivities<br>Capacities<br>Information exchanges<br>Operational environments<br>Exercise and training<br>RAM/EMC/EMP<br>Functional description<br>Interoperability<br>Security | Architecture<br>Design<br>Support<br>Interfaces<br>ICDs<br>Operating systems<br>Information management<br>applications<br>Integration and test<br>Installation and commissioning<br>Acceptance | Life cycle plans<br>Procurement plans<br>Integration and test Plans<br>Demonstration and acceptance plans<br>Transition-to - use plans | Design quality plans<br>Development quality plans<br>System proving quality plans |
| Standard Procedures for: | Requirements analysis<br>Bounding systems<br>Functional description<br>Functional decomposition<br>Functional-to - physical mapping<br>RAM analysis | Design<br>Partitioning<br>Test and integration<br>Installation and commissioning<br>Customer support/PDS<br>Estimating<br>Optimization<br>tradeofffs | Planning system design<br>Planning system development<br>Monitoring progress<br>Data management<br>Configuration management<br>Reviewing standards<br>Reviewing procedures | Design audit<br>Development audit<br>Development procedure audit<br>Project control audit<br>Quality plans review<br>Quality audit reviews |

**Standards & specifications**

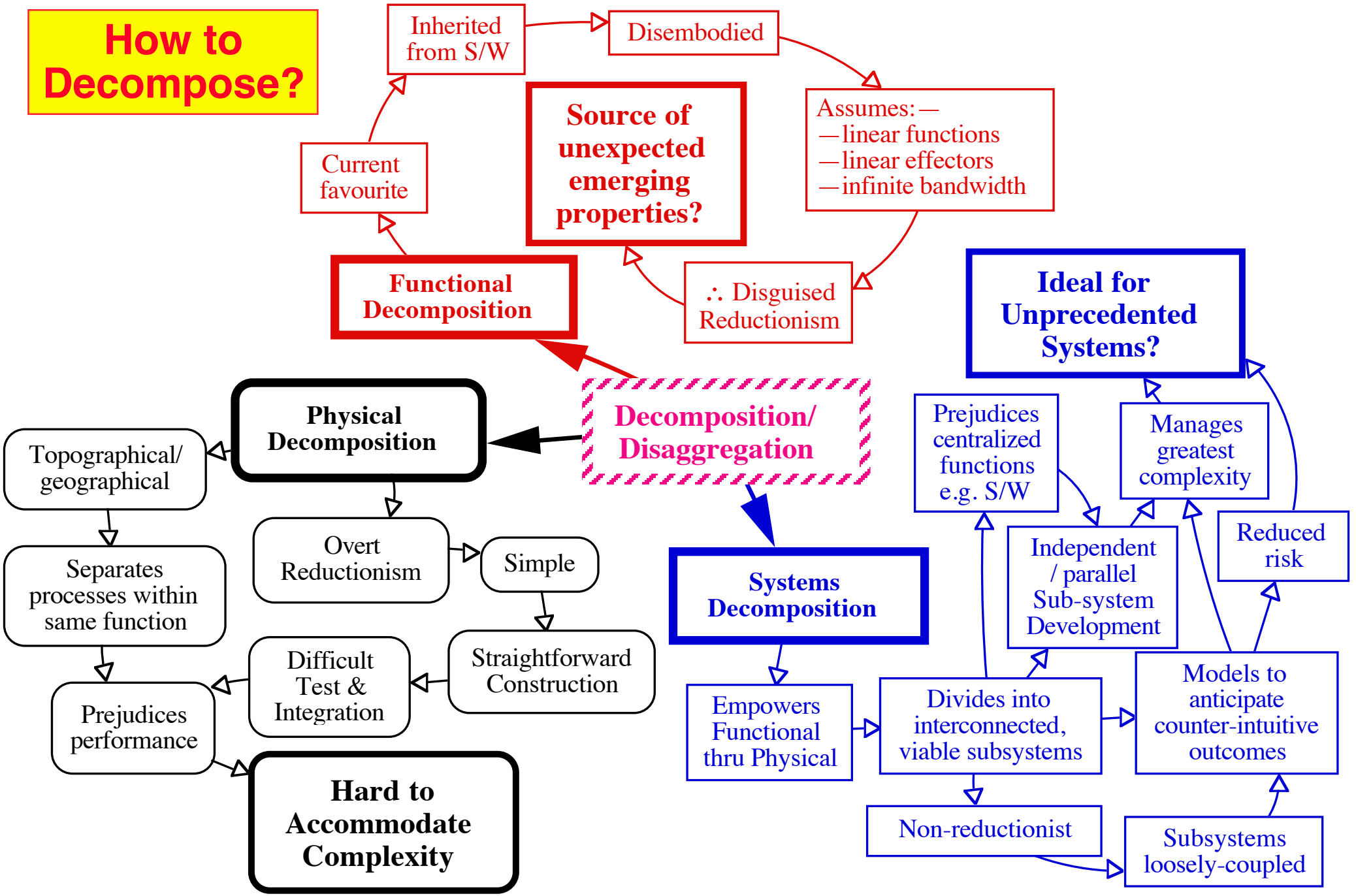
| | | | | |
|---|---|---|---|---|
| Mission analysis | Mission system design | System architecture | System requirements | Reliability, availability, maintainability |
| Man-machine interface | Human-computer interface | Human engineering | System specification | Trade-off analysis |
| Programming languages | Work breakdown structures | Statements of work | Master schedules | Risk management |
| Software development planning | Software requirements specification | Failure modes, effects and criticality analysis | Safety management<br>Design to cost | Design reviews and red teaming |
| Configuration management | Interface control documentation | Programme management | Data management | Quality management |
| Cost estimation | Electro magnetic compatibility | Integration and test | Test and trials, Trials analysis | Mock-ups and prototypes |
| Programme evaluation and review technique | Field support | Logistic support analysis | Integrated logistic support | Spares provisioning |
| Spares provisioning | Design change control | Test and support equipment | Automatic test equipment | Special-to-type support |
| Software support facilities | Operator training facilities | Configuration identification | Warranty | Modifications and mod. management |
| Software management | Survivability analysis | Damage tolerance | Maintenance manuals | Operations manuals |
| Technical publications | Training requirements | Training facilities | Maintenance training | On-the-job training |
| Life cycle costing | Repair and maintenance philosophies | Etc. | Etc. | Etc. |

**Some of the Skills**

# *Critique* of Systems Engineering Practices

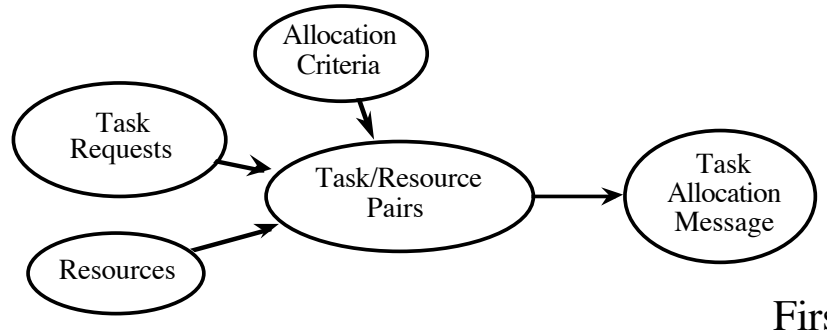
# Decomposition

- Any system made up from both **transformer** and **transporter** systems
- Software is not a system—transforms *only* in conjunction with hardware, does not transport
- Software functional decomposition flawed:—
  - implies undefined physical transformer and transporter systems, ignoring how their unstated characteristics would affect function & behaviour (e.g. limits, non-linearities…)
  - e.g. not possible to rigorously add back sub-functions to re-create original functions, without establishing:—
    - » sub-function behaviour and
    - » sub-function interconnection structure and limitations


How to Decompose?
Inherited from S/W
Disembodied
Current favourite
Source of unexpected emerging properties?
Assumes:—
—linear functions
—linear effectors
—infinite bandwidth
Functional Decomposition
∴ Disguised Reductionism
Ideal for Unprecedented Systems?
Decomposition/ Disaggregation
Physical Decomposition
Topographical/ geographical
Separates processes within same function
Prejudices performance
Overt Reductionism
Simple
Straightforward Construction
Difficult Test & Integration
Hard to Accommodate Complexity
Systems Decomposition
Empowers Functional thru Physical
Prejudices centralized functions e.g. S/W
Manages greatest complexity
Independent / parallel Sub-system Development
Reduced risk
Divides into interconnected, viable subsystems
Models to anticipate counter-intuitive outcomes
Non-reductionist
Subsystems loosely-coupled

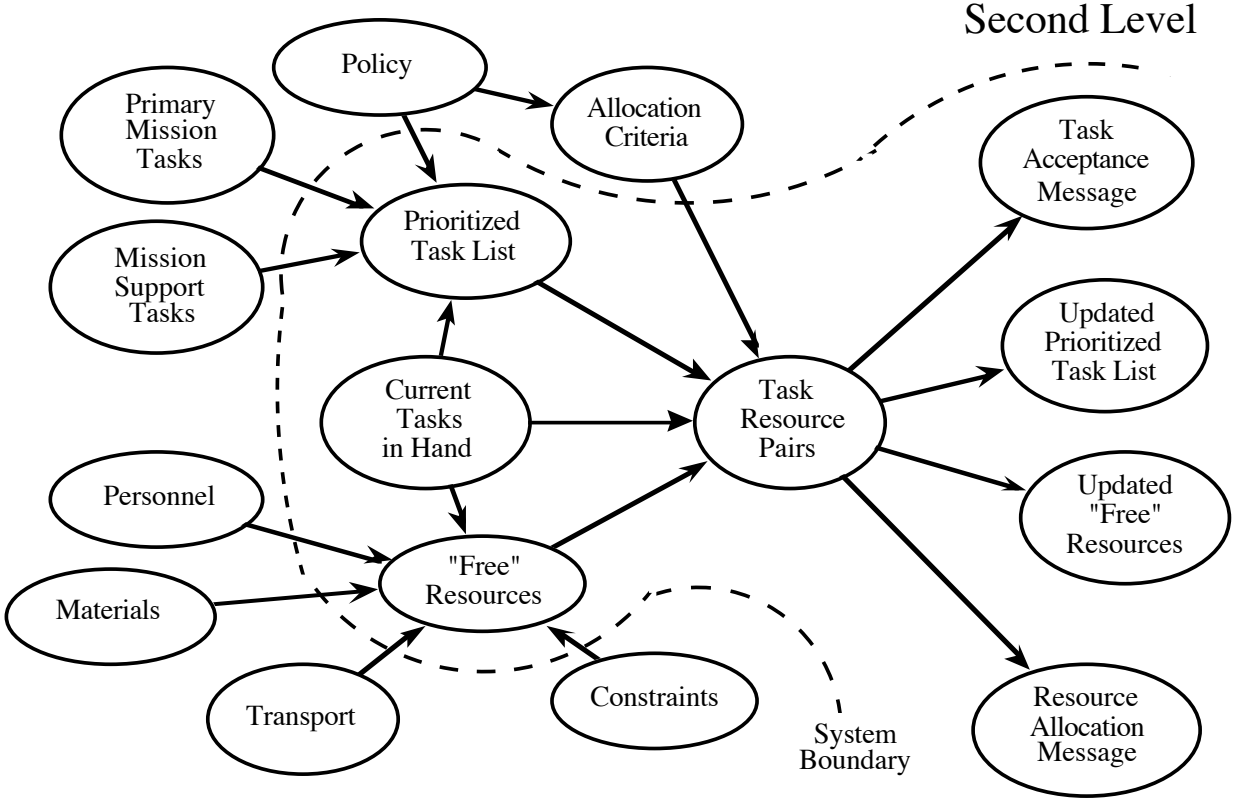
# Functional Decomposition

*Allocation of Resources to Tasks*




First Level

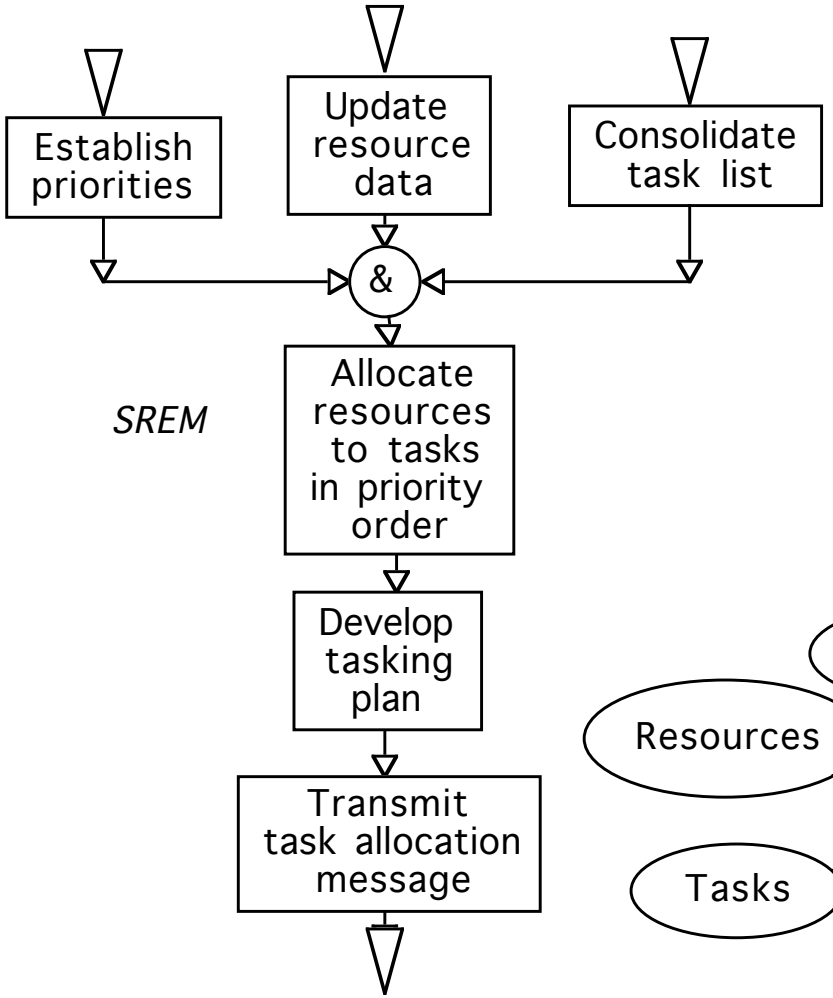
Second Level




**Functional decomposition—1**

Data State Design

# Functional Decomposition—2



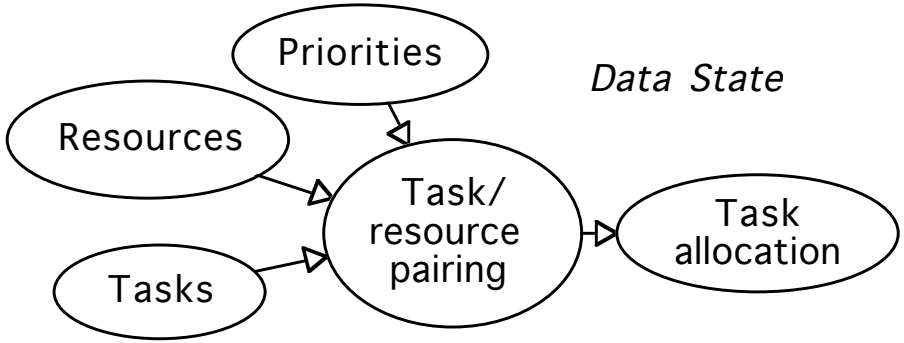

*SREM*

*States*

Unallocated Resources

Allocated Resources

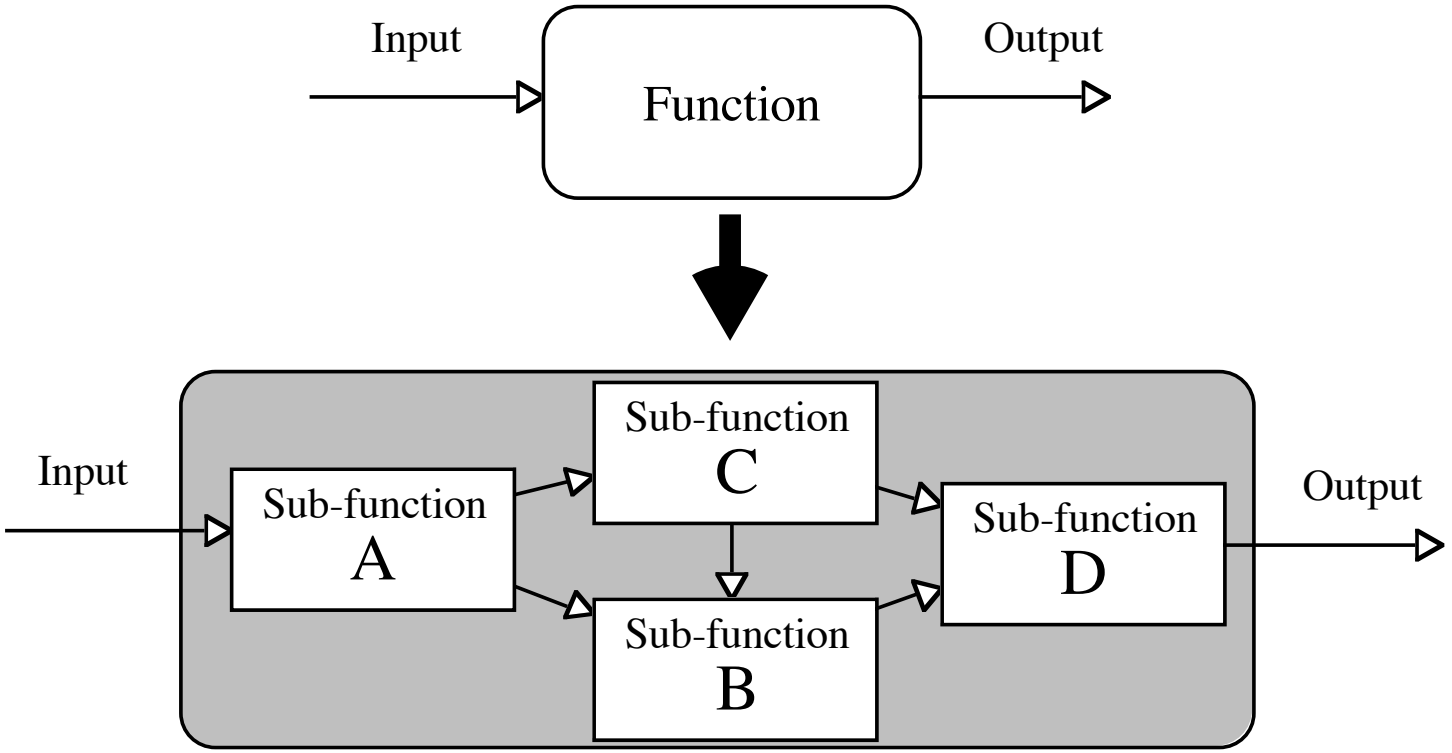
Assigned Resources



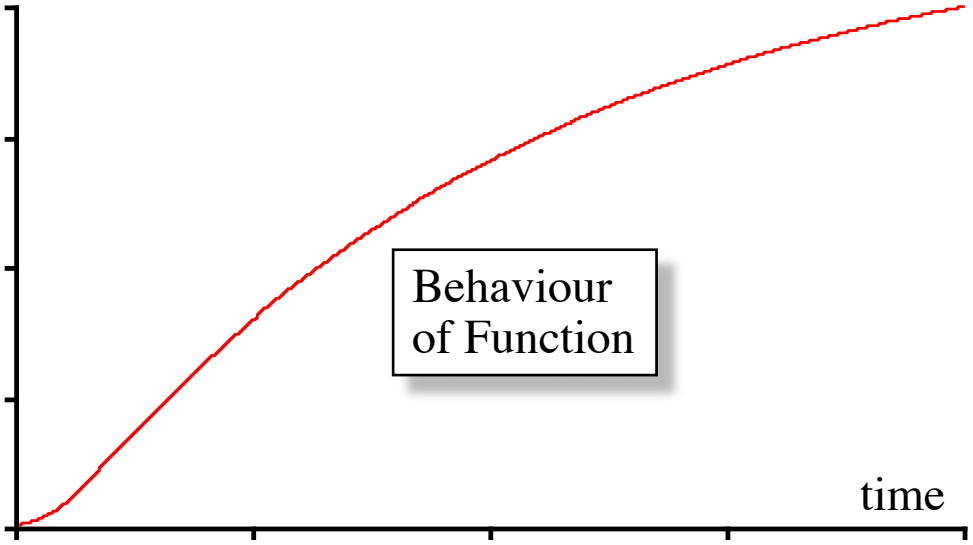

*Data State*

# Functional Decomposition

- The standard software paradigm—but, is it valid?
- Could you **prove** that the dynamic response of the decomposed figure will be identical with that of the original without *first* defining physical parameters and limitations?

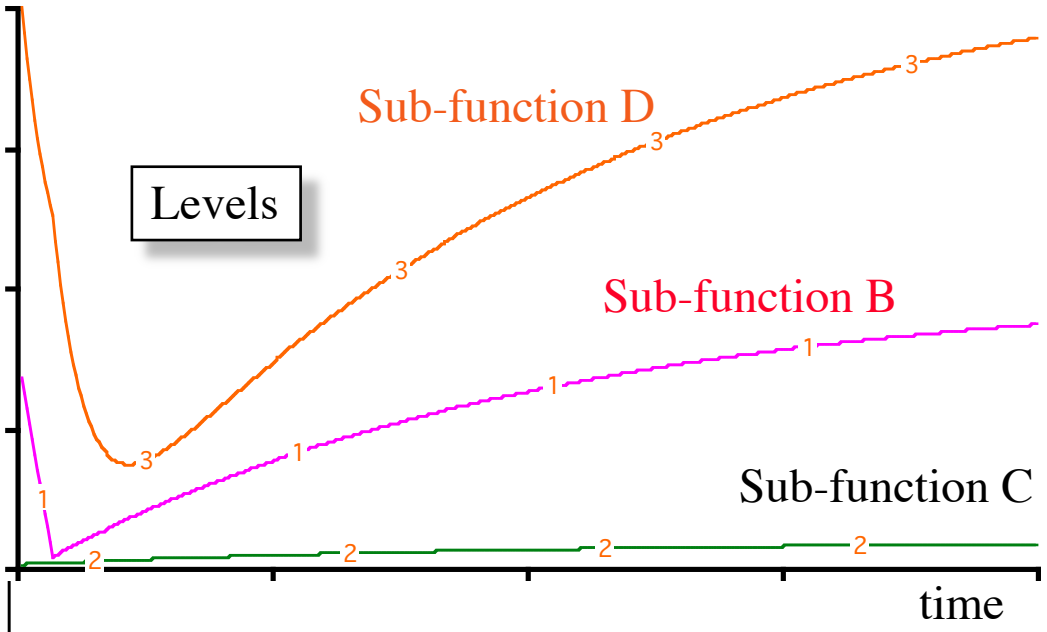


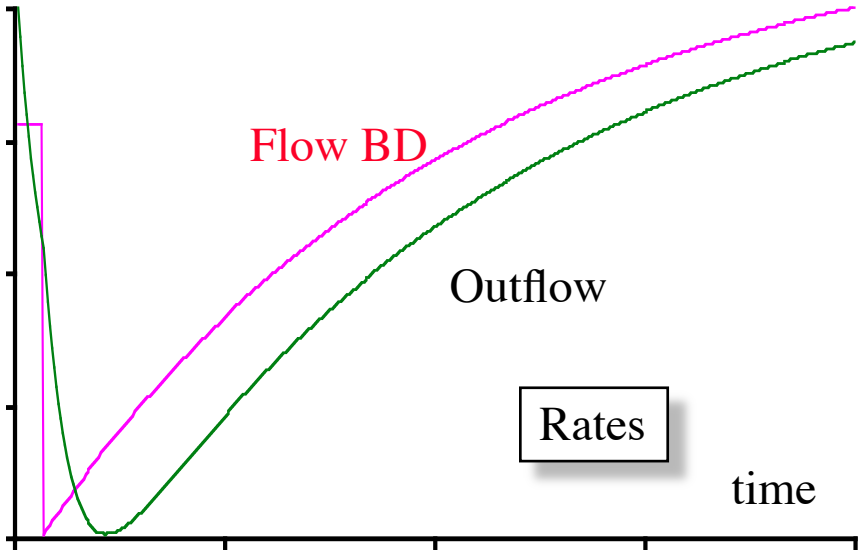
# Function Behaviour




- Overall function is to accumulate a reserve by holding back on some of the flow-through
  - working effectively
- Sub-functions channel different elements of flow, some being delayed more than others
- Sub-functions have capacity limits, inflow limits, starting conditions, etc.
- None of these is visible in the function behaviour—how about at sub-function level?

# Sub-function Behaviour




- Determined only by modelling:—
  - characteristics of each sub-function—capacity, inflow limits, delays, transit times and their variations…
  - nature of interchange between sub-functions—direction, driving functions, etc.
- Each sub-function affects the others—so how can decomposing into separate functions *work?*

# Systems Decomposition

# System Decomposition—1

- Any viable system is:—
  - open, connected to, and interacting with other viable systems
  - comprised of open, interacting complementary subsystems
- Partition overall system into viable subsystems:—
  - Each subsystem is a viable system in its own right—complete with sensors, organs, architecture, effectors…
  - Every subsystem is complementary and interconnected
  - Creates infrastructure
  - Retains interconnections to "other viable systems"

# System Decomposition—2

- Two philosophies:—
  - decompose "inwards—specify a system as comprising subsystems and infrastructure

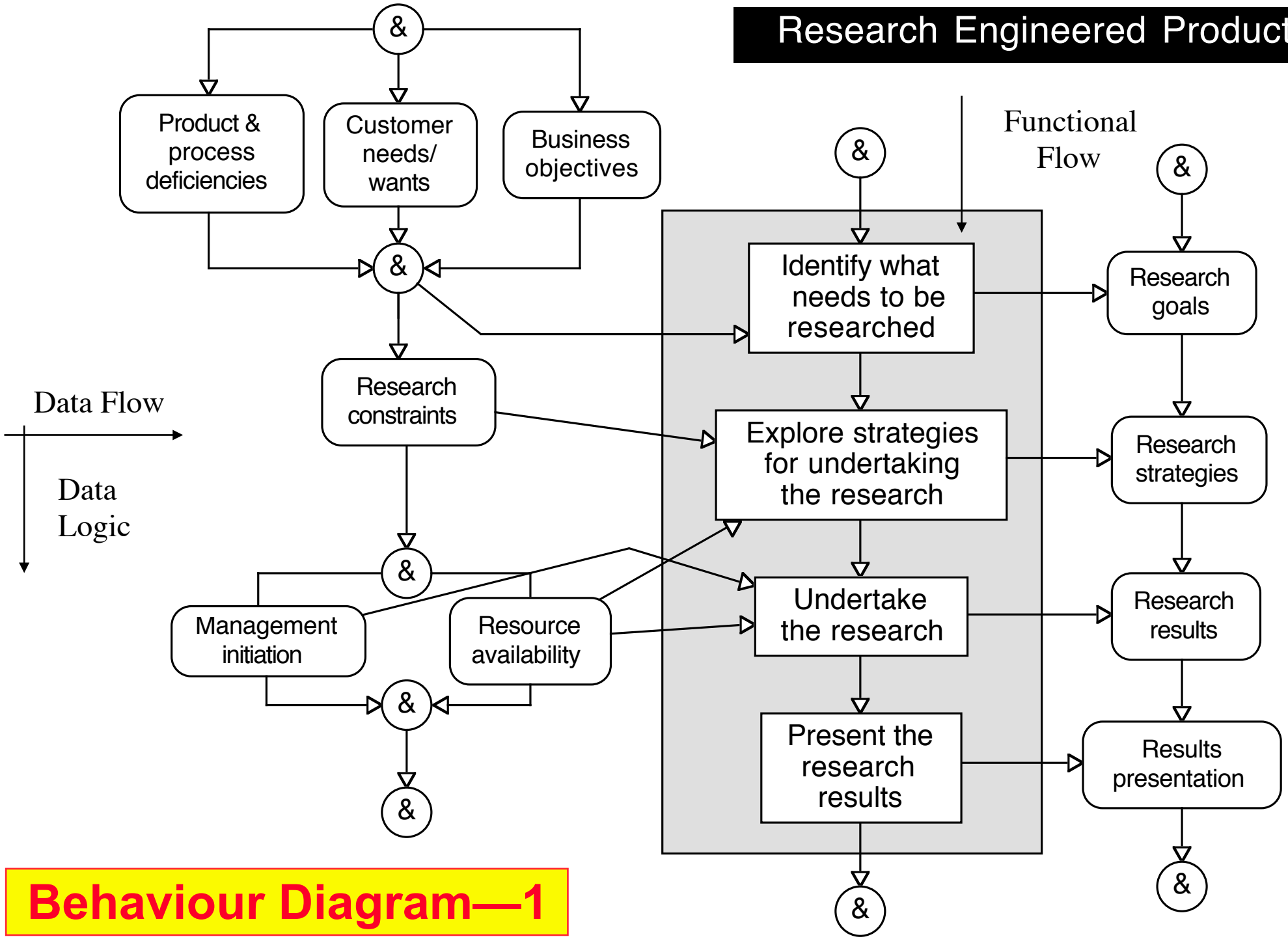
    ***OR***

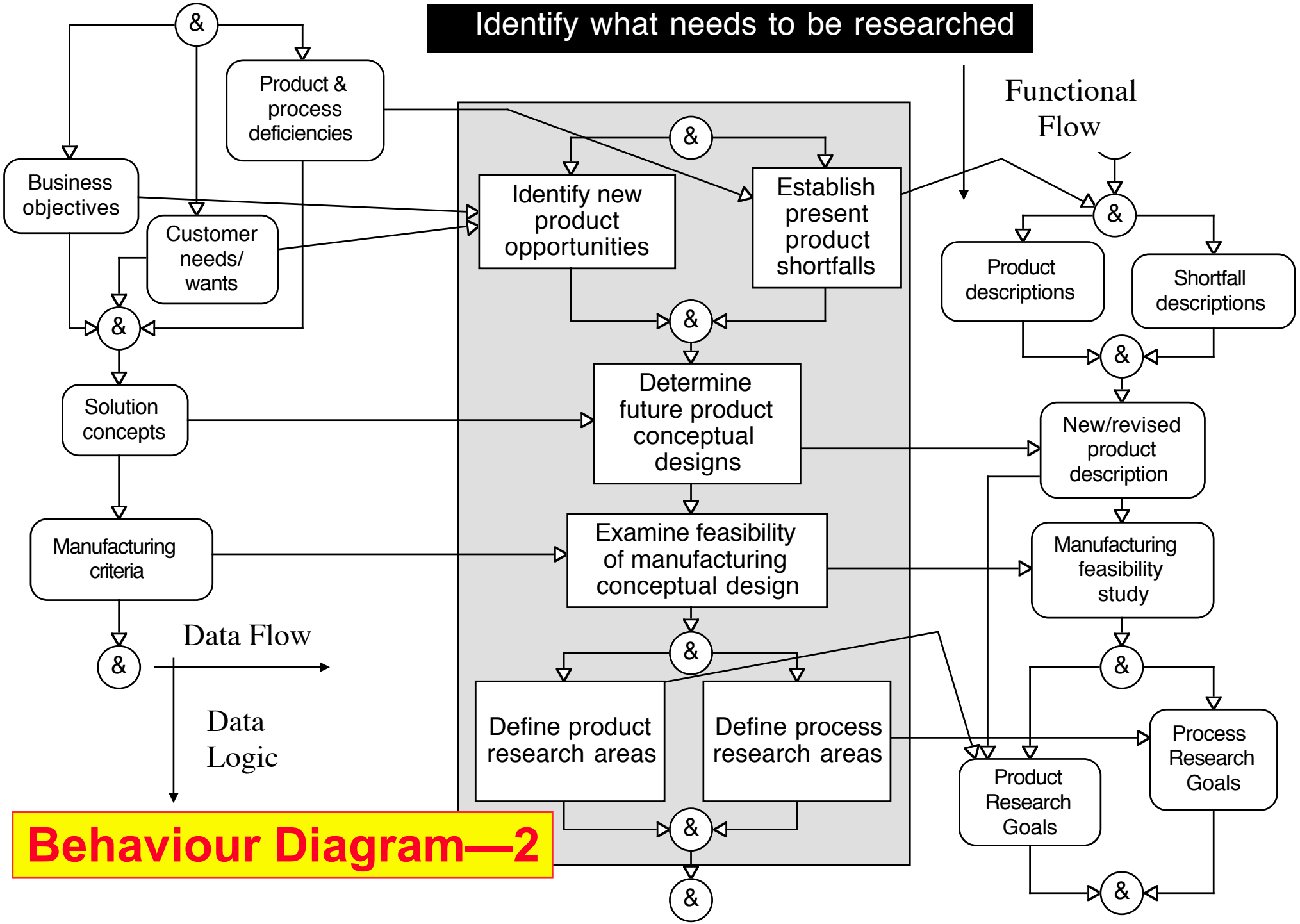
  - synthesize "outwards". Specify a system's:—
    - » properties, capabilities and behaviours,
    - » interactions with other systems, and
    - » contribution to a "containing system"
- First method is the convention—reductionist
- Second method is powerful—synthesist

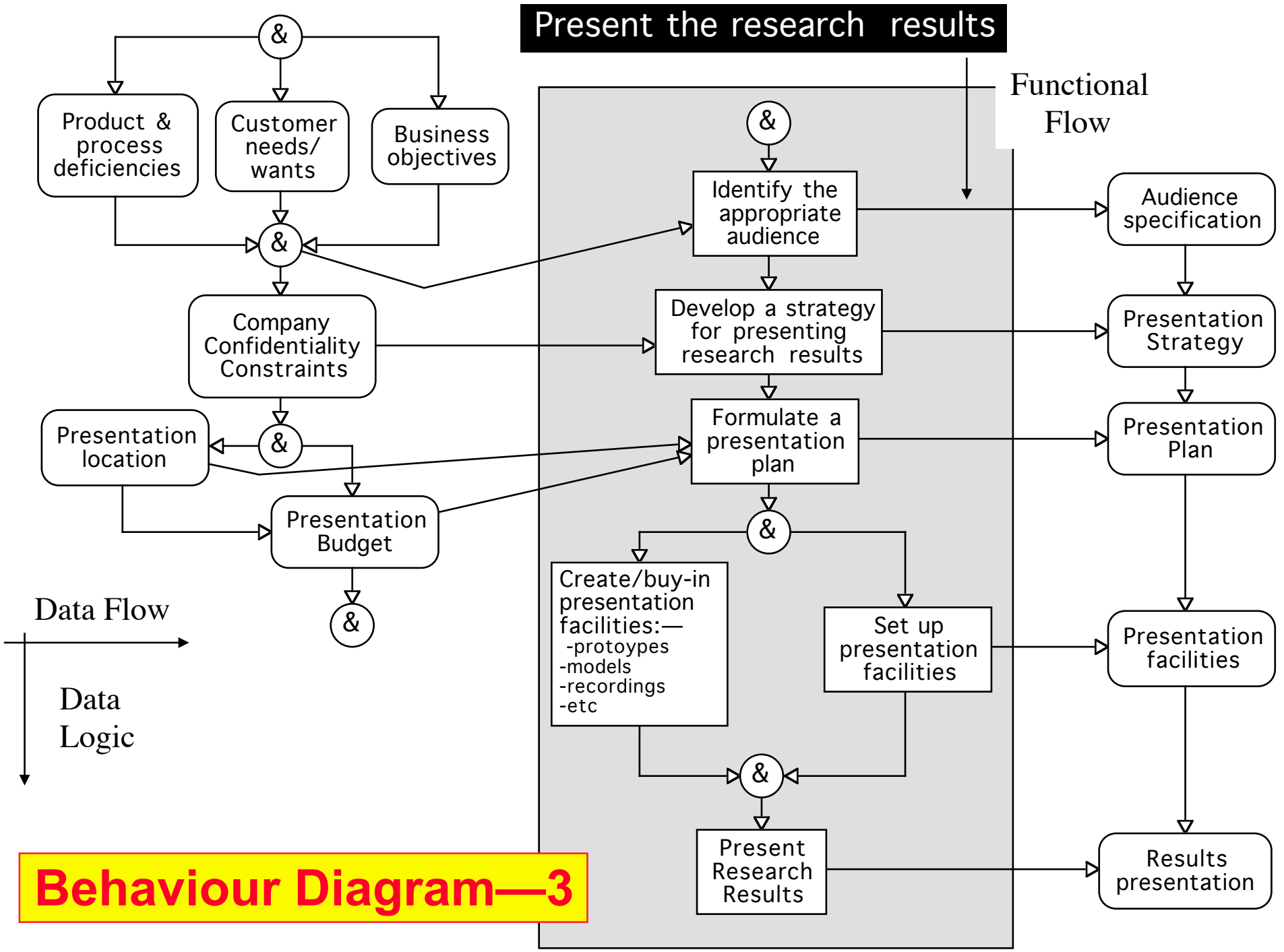
# Value of Systems Decomposition

- Each and every viable subsystem can be developed independently as parallel projects:—
  - requires clean interface and emergent property definition from start.
  - minimal organizational entropy
  - motivated, dedicated small project teams
- Problems arising in any one project do not hamper others
  - problem project— apply additional effort back-up, or, change strategy—go around problem
- Offers optimum route to innovative/unprecedented systems
  - e.g. Giza Pyramids & Neil Armstrong on the Moon
- Avoids "economy of scale" delusions and traps:—
  - e.g. no central software function

# Managing and Recording Decomposition


Research Engineered Products
Product & process deficiencies
Customer needs/ wants
Business objectives
Functional Flow
Identify what needs to be researched
Research goals
Research constraints
Explore strategies for undertaking the research
Research strategies
Data Flow
Data Logic
Management initiation
Resource availability
Undertake the research
Research results
Present the research results
Results presentation
Behaviour Diagram—1


Identify what needs to be researched
Product & process deficiencies
Business objectives
Customer needs/ wants
Identify new product opportunities
Establish present product shortfalls
Functional Flow
Product descriptions
Shortfall descriptions
Solution concepts
Determine future product conceptual designs
New/revised product description
Manufacturing criteria
Examine feasibility of manufacturing conceptual design
Manufacturing feasibility study
Data Flow
Data Logic
Define product research areas
Define process research areas
Process Research Goals
Product Research Goals
Behaviour Diagram—2


Present the research results
Functional Flow
Product & process deficiencies
Customer needs/ wants
Business objectives
Company Confidentiality Constraints
Presentation location
Presentation Budget
Identify the appropriate audience
Develop a strategy for presenting research results
Formulate a presentation plan
Create/buy-in presentation facilities:—
-protoypes
-models
-recordings
-etc
Set up presentation facilities
Present Research Results
Audience specification
Presentation Strategy
Presentation Plan
Presentation facilities
Results presentation
Data Flow
Data Logic
Behaviour Diagram—3

# Summary

# We haven't Scratched the Surface…

- Behaviour, individually and socially
- Identifying your good systems people
- Measuring systems
- Evolving better systems
- Training, tools, techniques, methods…
- Architectonics, emergence, organization, regulation and control…

# System Engineer's Shadowboard

| Operations Analysis | Requirements Analysis | System Design | Project Engineering | Integration and Test | Installation & Commissioning |
|---|---|---|---|---|---|
| Solution feasibility & Performance | Requirements consistency & completeness | Design options, interfaces, tradeoffs & specifications | Configuration, compatibility, interchange | Test environment | Customer acceptance |

| | | | | | |
|---|---|---|---|---|---|
| Scenario Models | System Models | | | Environment Simulation | |
| | Relationship Models | | | | |
| System boundary models | Requirements Tools | | | Threat Simulation | Acceptance Models |
| | Human engineering | | | | |
| | Logistics Models | | | Sub-system Simulation | |
| Risk Models | RAM/FMECA | | | | |
| Functional Decompo-sition | Networks & Architectures | Configuration management tools | | | |
| | System Prototyping | Interface control tools | | | |
| | Functional/physical mapping | Data management tools | | | |
| System design and engineering framework model | | | | | |
| Cost, planning and scheduling tools and models | | | | | |

# The Future of Systems Engineering

- Resurgent—taking over from the piecemeal, bits and pieces of Right First Time, Zero Defects
- TQMdead or dying, BPR following suit
- Requires special people—systems architects?—with ability to see the whole project
- Automated support becoming available—offers productivity for these few special people
- Successful companies will employ systems engineering principles
- Japanese—good systems engineers, but do not have discrete quality or systems engineering departments

# Future Developments?

- Systems engineering the Business:—
  - overcome wasteful, damaging internal divisions between commercial, marketing, finance, business, R&D, design, devalopment, assembly/production…
- Systems engineering the Supply Chain/Circle
  - wealth creation (= survival) in volume manufacture
  - spread of synergistic relationships with suppliers
    - » supply chain information and business systems
    - » *Kaizen* to continuously improve performance
      - move toward problem-solving ethic
    - » creation of stable environment within chain
      - reduce logistic waste
      - enhance creativity
- Join or fail?

# Why are systems engineers different?

- Multi-disciplinary
- …but a different discipline, too…
- Perceive whole as parts *plus interrelationships*
- Cannot be taught, but innate capability can be developed
- Ability to build mental models, projecting experience into future situations—imagination
- Not that uncommon—about 10% of staff have potential
- Physicists often make good systems engineers

Wherever, your company needs good systems engineering to compete

# Training Systems Engineers

- Not practicable at undergraduate level
  - Need a good classical discipline on which to build e.g. physics, biology, engineering
  - Concepts develop through making mistakes
  - No rôle for a 22-year old systems engineer?
- Youngest age for training c.28, having had a variety of jobs, and some failures

# Conclusions

- Systems and systems engineering definitions—feasible and important—basis for discipline
- Systems engineering looks at *whole systems* and at *all management levels*. Some current ideas too introspective
- Not some short-term, quick-fix "silver bullet" / t.l.a.
- A philosophy. A way of thinking. A way of Life, of Performance through Continual Improvement.
- Not about product *or* process—*not separable* in true systems engineering
- Major business advantage accrues from *strategic systems thinking*.

# Assignment

- Identify what you think are the pros and cons of systems engineering
- Develop a plan to extend systems engineering to your whole business/supply chain
- Present your views