30th Annual INCOSE international symposium

Virtual Event
July 20 - 22, 2020

Swaminathan Natarajan, Anand Kumar, Subhrojyoti C, Rahul Sinha
swami.n@tcs.com, {anand.ar, subhrojyoti.c, sinha.rahul}@tcs.com
Tata Consultancy Services Research, India
also INCOSE Systems Science WG

Linking Behaviour Data to Knowledge: Contextualization and De-Contextualization

Motivation

As engineers, we understand systems at multiple levels

- System Models
- Knowledge
- Observable systems behaviour and data

How do these relate to each other? How can we link up

- Systems Models?
- Knowledge Models?
- Data Science?

This paper presents a conceptual approach based on systems science

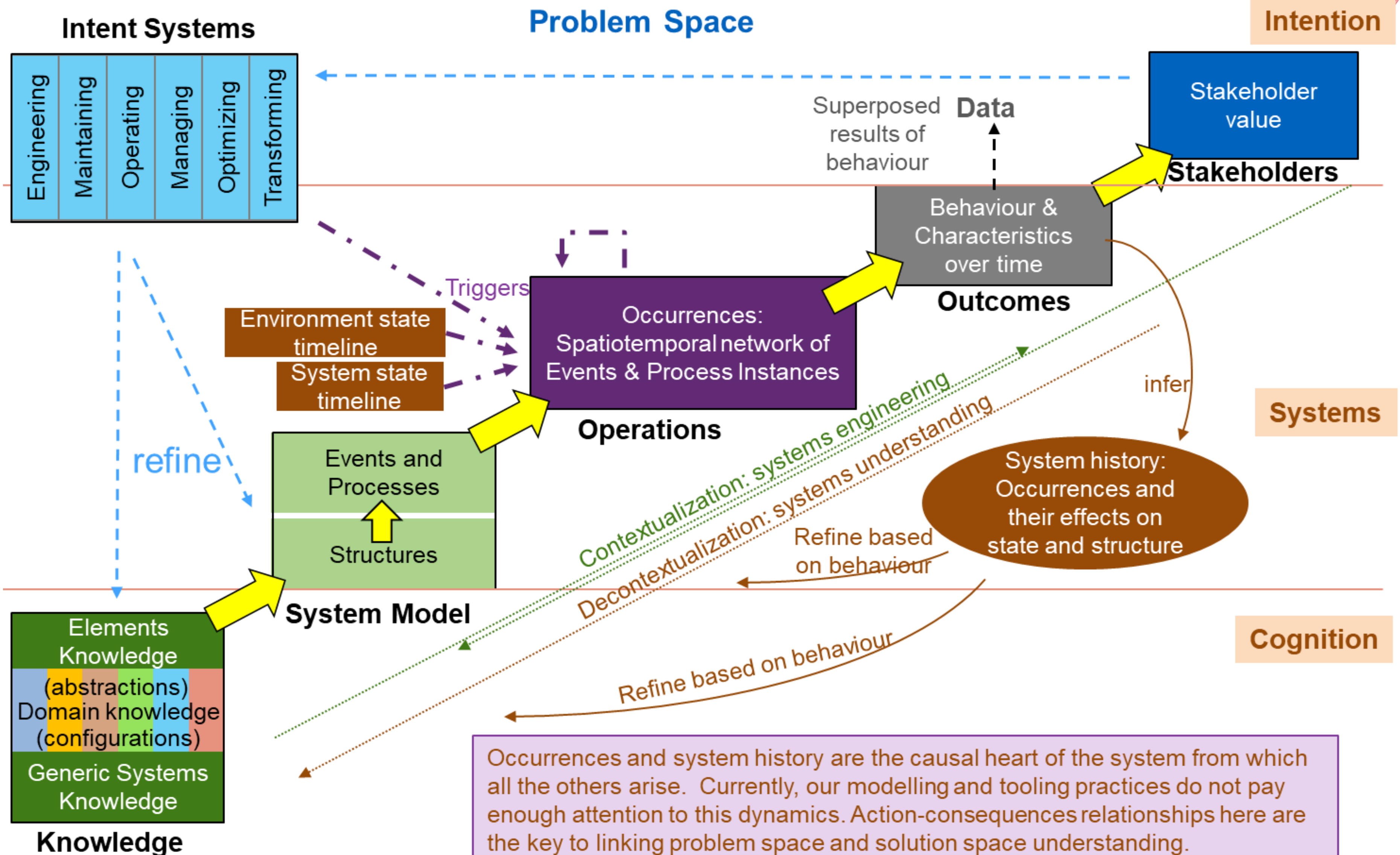
Frames these as contextualization / de-contextualization levels

Disclaimer: Not new ideas, but synthesis of existing understanding

Outline

- Generative Levels in Systems Understanding
- Modelling Systems Knowledge
  - Systems Phenomenon
  - Nature of Knowledge Formation: De-contextualization
  - Type DAG: Levels of Knowledge about an entity or interaction
  - Knowledge Frames
- Systems Modelling
  - Context Roles: Capturing Context Assumptions
  - Compositionality: Contents of Systems Models
  - Generic Block Model
- Modelling Behaviour
  - Modelling Behaviour Data
  - Observable Behaviour Modelling
- Framework: Levels of Contextualization / De-contextualization
- Summary and Next Steps

Knowledge to Data: Generative Levels

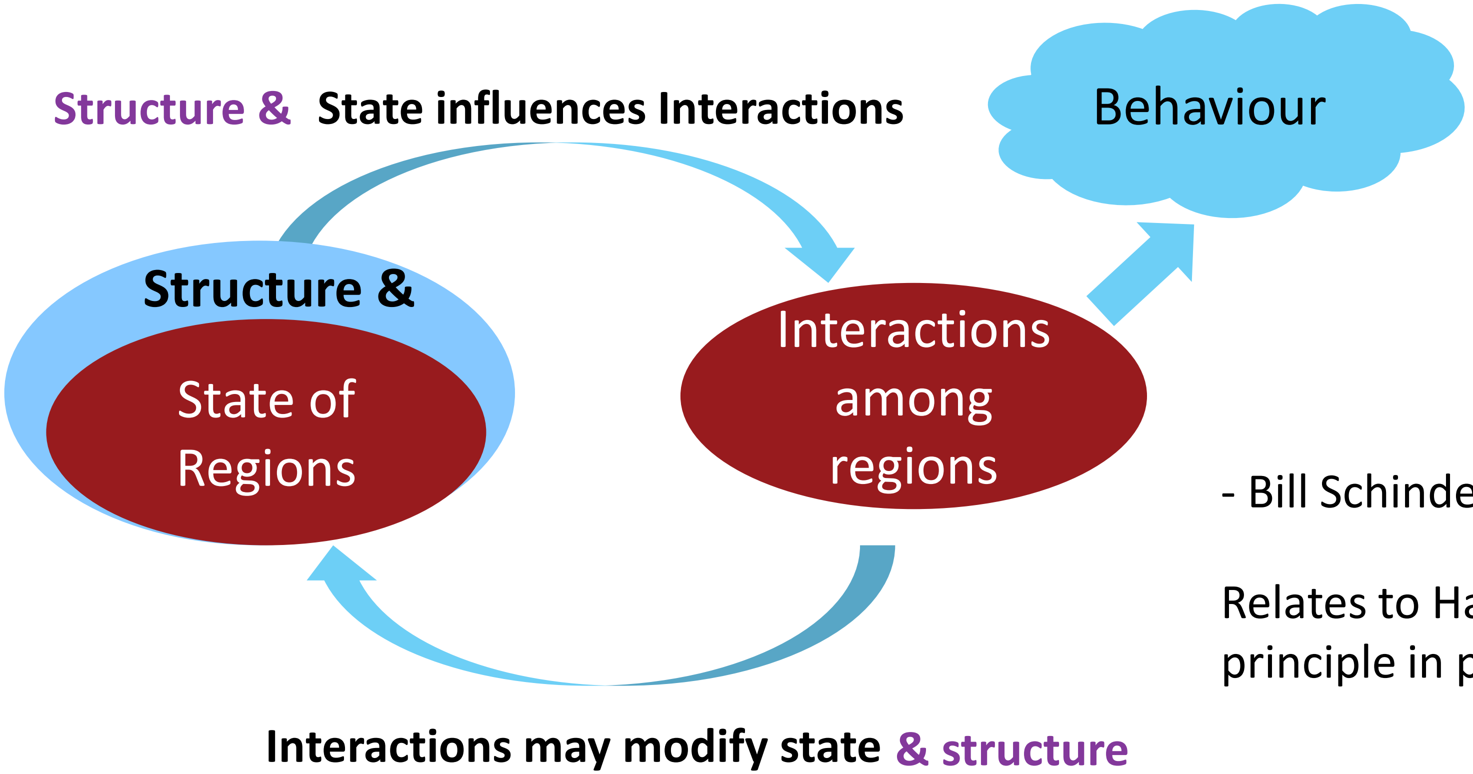


Occurrences and system history are the causal heart of the system from which all the others arise. Currently, our modelling and tooling practices do not pay enough attention to this dynamics. Action-consequences relationships here are the key to linking problem space and solution space understanding.

Solution Space

Knowledge Modelling

Systems Phenomenon



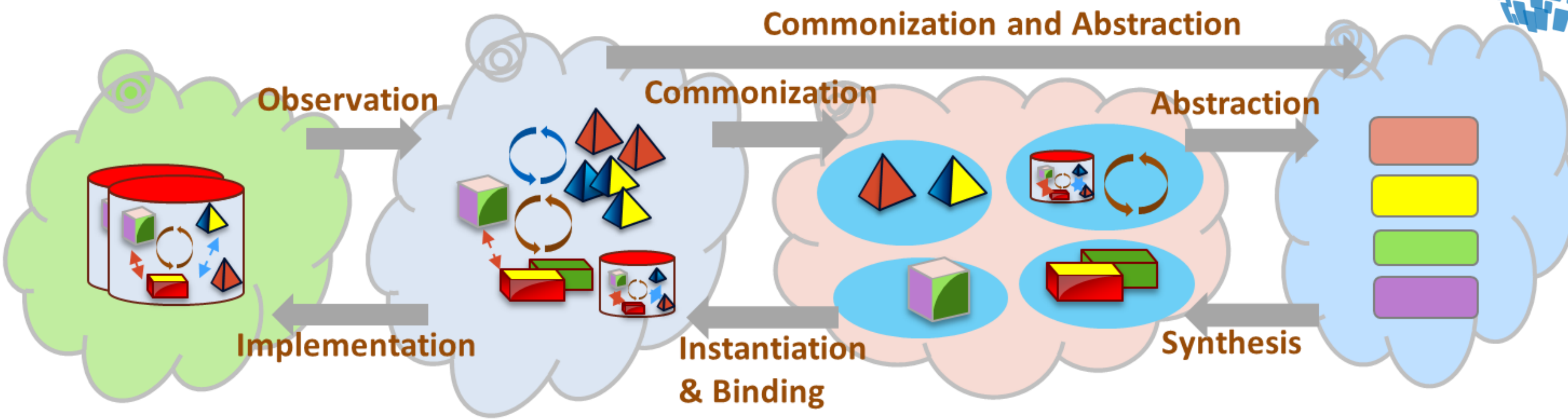
- Bill Schindel

Relates to Hamilton's principle in physics

Systems axiom: **Structures + context → Interactions → Outcomes**

Observable behaviour
Change in system
Change in environment

Formation of Knowledge



World of interest		Models World	Commonizations world	Abstractions World
Real World Experiential	Gaps	World of instance descriptions	Vocabularies of "wholes" "Application" domains	Viewpoints "Theory" domains

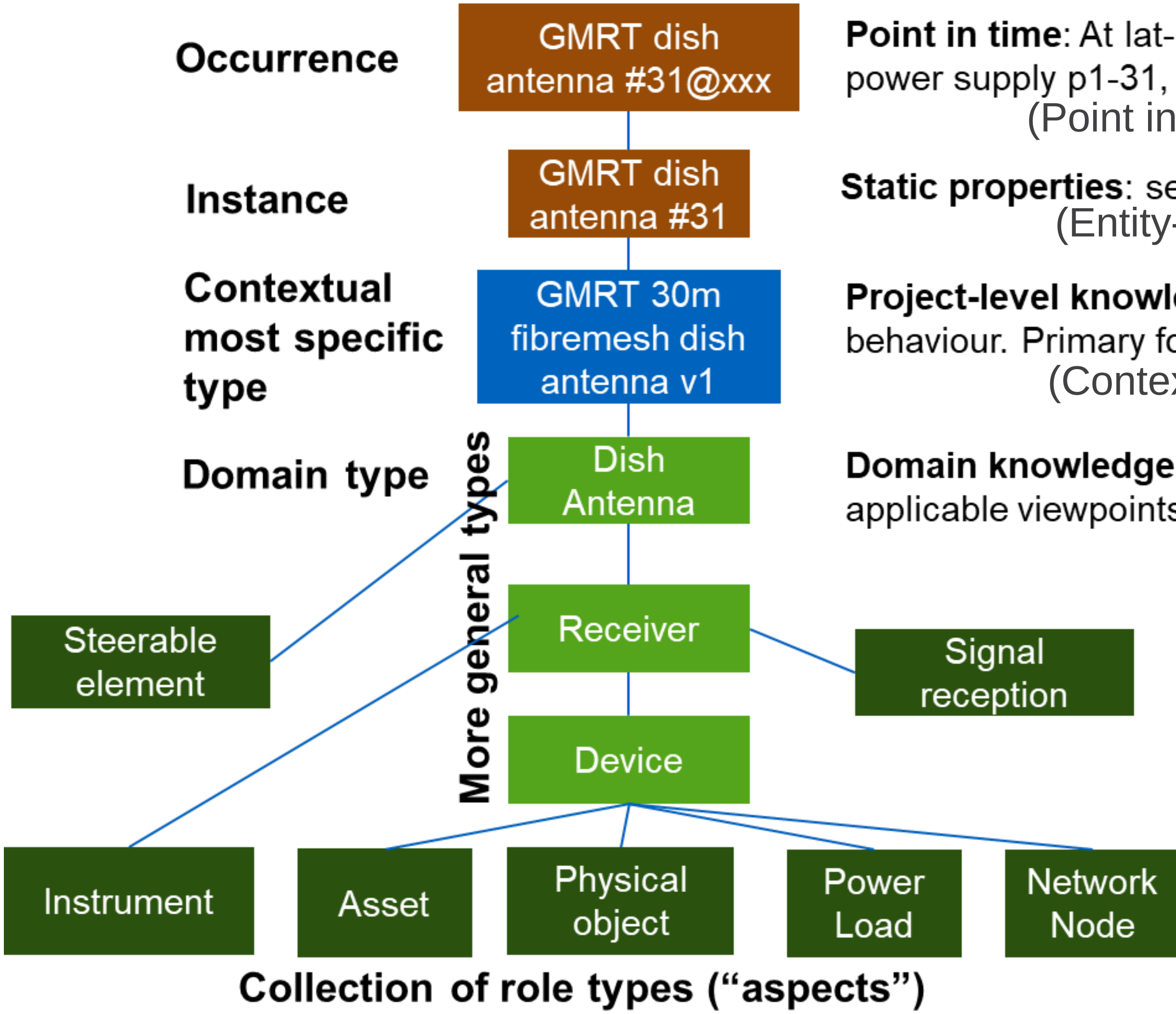
Systems | **Knowledge**

- Whenever we describe a world of interest formally or informally, we are in models world (tacit, informal, formal)
- Vocabulary and knowledge is generated by identifying commonalities across sets of observations. We can use this vocabulary only when describing instances in models world
- Focusing only on particular viewpoints / aspects allows to create deeper "theoretical" knowledge

- Network routers
- Candle, dog, tree
- Walking, raining
- Full, stationary (states)
- Above, inside (relationships)
- Knowledge that applies to similar configurations of elements

- Network routing
- Light source, power supply, parent
- Update operation, performance measure
- Knowledge that applies when we limit focus to a particular aspect or viewpoint

Type DAG: The types of a Whole Entity or Activity



Point in time: At lat-long (xxx, yyy), connected to power supply p1-31, on dd-mm-yy at hh:mm:ss
(Point in spacetime)

Static properties: serial number, history, current state
(Entity-specific)

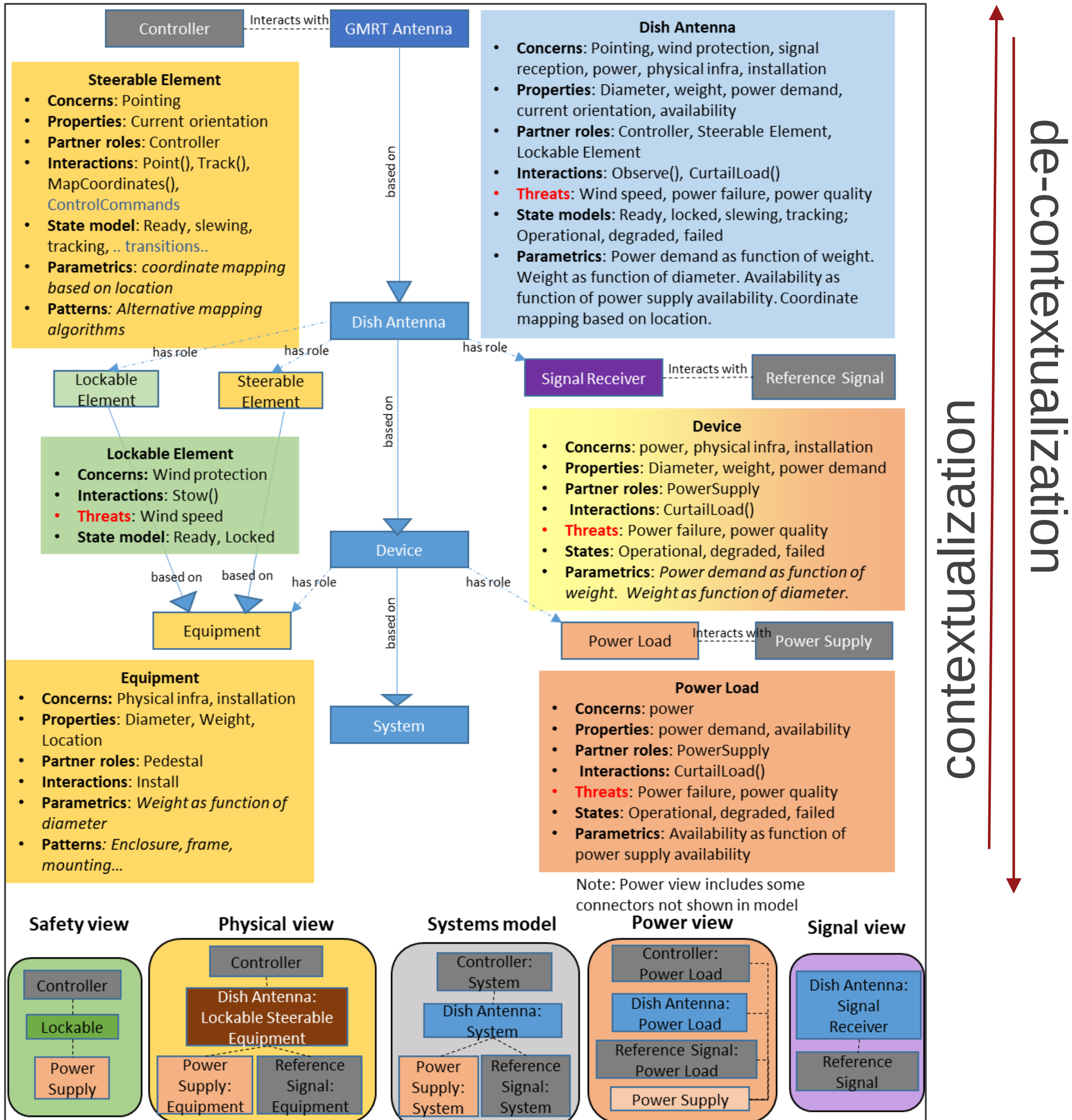
Project-level knowledge: Desired properties, processes, behaviour. Primary focus of models and proof structures
(Context-specific)

Domain knowledge: Typical properties, concerns, applicable viewpoints, processes etc.

All the green nodes are at the level of domain knowledge, not customer or project-specific.
We can build knowledge at each node that includes concerns, properties, states, failure modes, processes, state machines, property relationships etc. – basically a complete systems model of the abstract entity.

Key point: The actual **type** of the occurrence is the entire DAG, not just “dish antenna” or the most specific type. The green types are abstract types and roles that entity plays in viewpoints

Knowledge Synthesis and View Generation



Knowledge can be captured at each level of the Type DAG. Higher levels synthesize knowledge from lower levels.

We can build up a library of aspects and objects that will make it much easier to create models and system designs. We can also build tooling to support the design synthesis (contextualization) and knowledge formation (de-contextualization) processes.

The idea of designing using such libraries is called PBSE (Patterns-based systems engineering)

- Bill Schindel

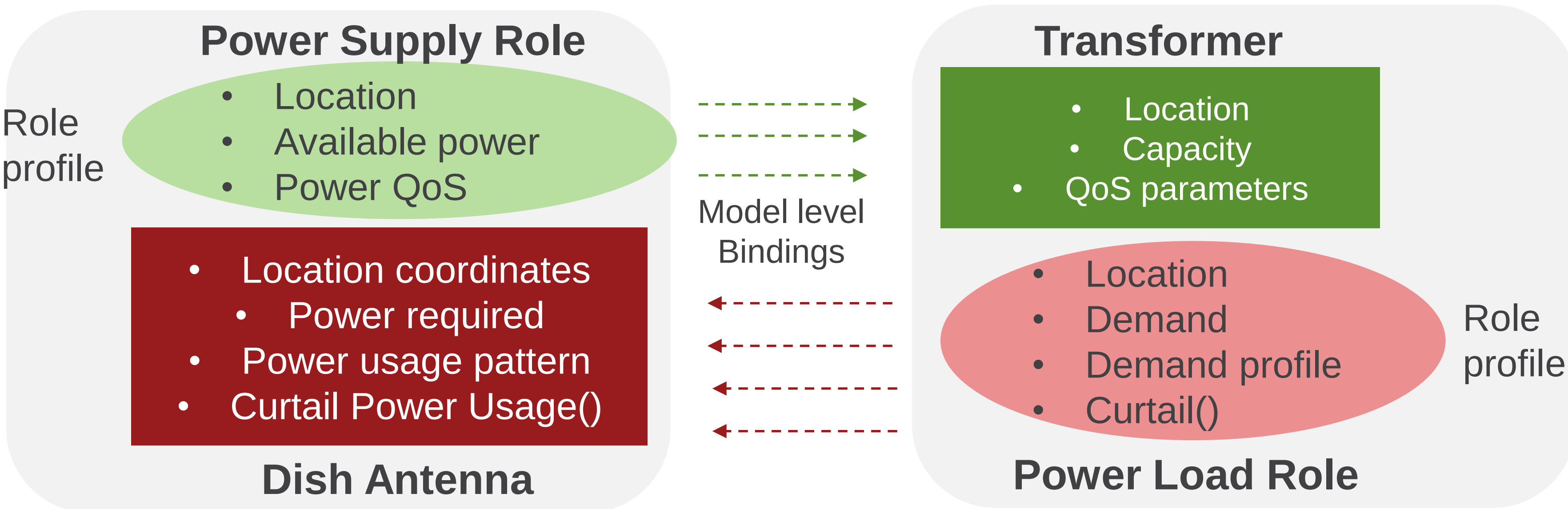
Knowledge Frames

	Entities	Interactions
Characterization	Properties	Associated flows, flow properties
Dependencies	Own roles, bindings, relationships with other roles	Participant roles and their states, including agents and resources
Behaviour determinants	States, [state machine, thresholds for stocks]	Triggering events (if any)
Variety and pathologies	Undesired inputs, pathologies, effects on behaviour	Undesired inputs, pathologies, effects on behaviour
Behaviour mechanism	Interfaces & interactions	Effect on state and structure of participants
Parametrics	Parametric relationships (internal & external)	Parametric relationships describing behaviour
Context impacts	Events, concerns, influences, outcomes	(captured in role impacts)
Internal pattern of organization	Parts and internal structure	Sequence of steps

Key idea: Align systems knowledge modelling with systems modelling

Systems Models

Context Roles & Role Profiles



Role profiles capture assumptions each entity makes about other: Assume-Guarantee

Binding

System integration involves binding wholes to each other.
Each entity must meet the role profile assumptions of the other

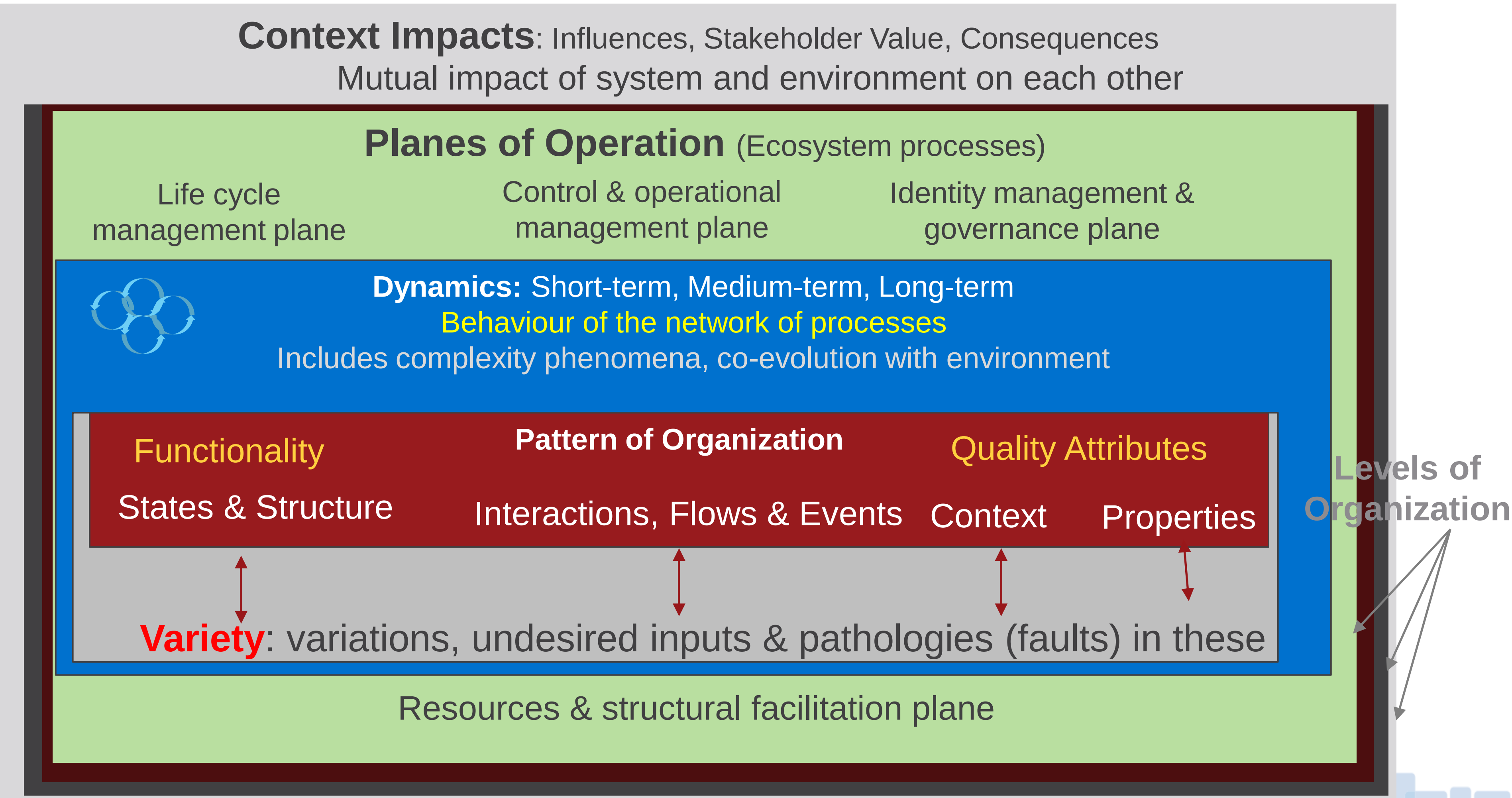
Knowledge domains include context roles for entities from other domains
Type DAG identifies the various roles (types) that an entity can play

Key Idea: Entity & Interaction Models should be self-contained

Conceptual Model of Block Compositionality

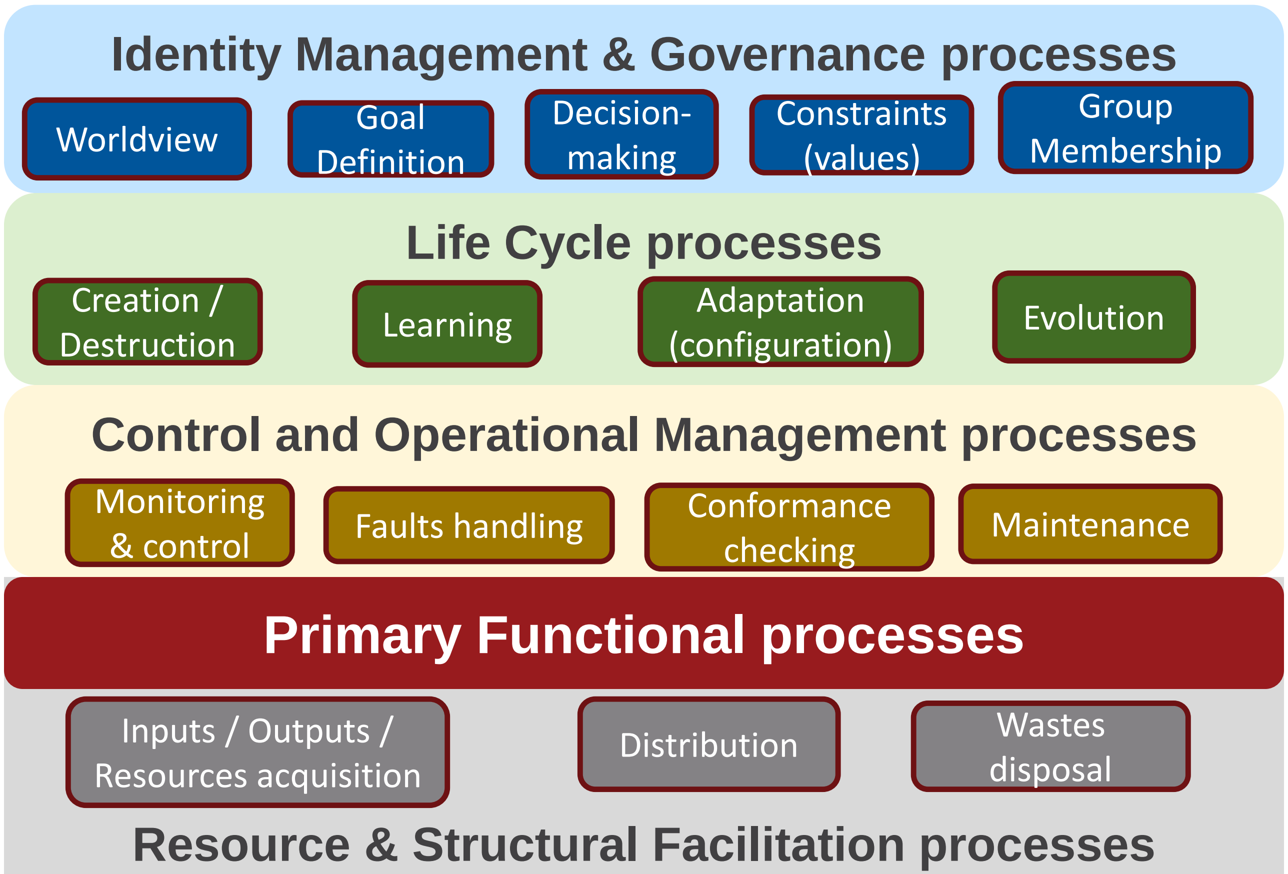
What contents should we include in a system model?

What all do we need to take into account in order to assert that a configuration of parts with particular characteristics will produce particular behaviours and characteristics?



Planes of Operation

- A system is acted upon by its ecosystem in multiple ways
  - For mechanistic systems, we think of these as intent systems
- Partition the network of processes associated with a system into planes of operation



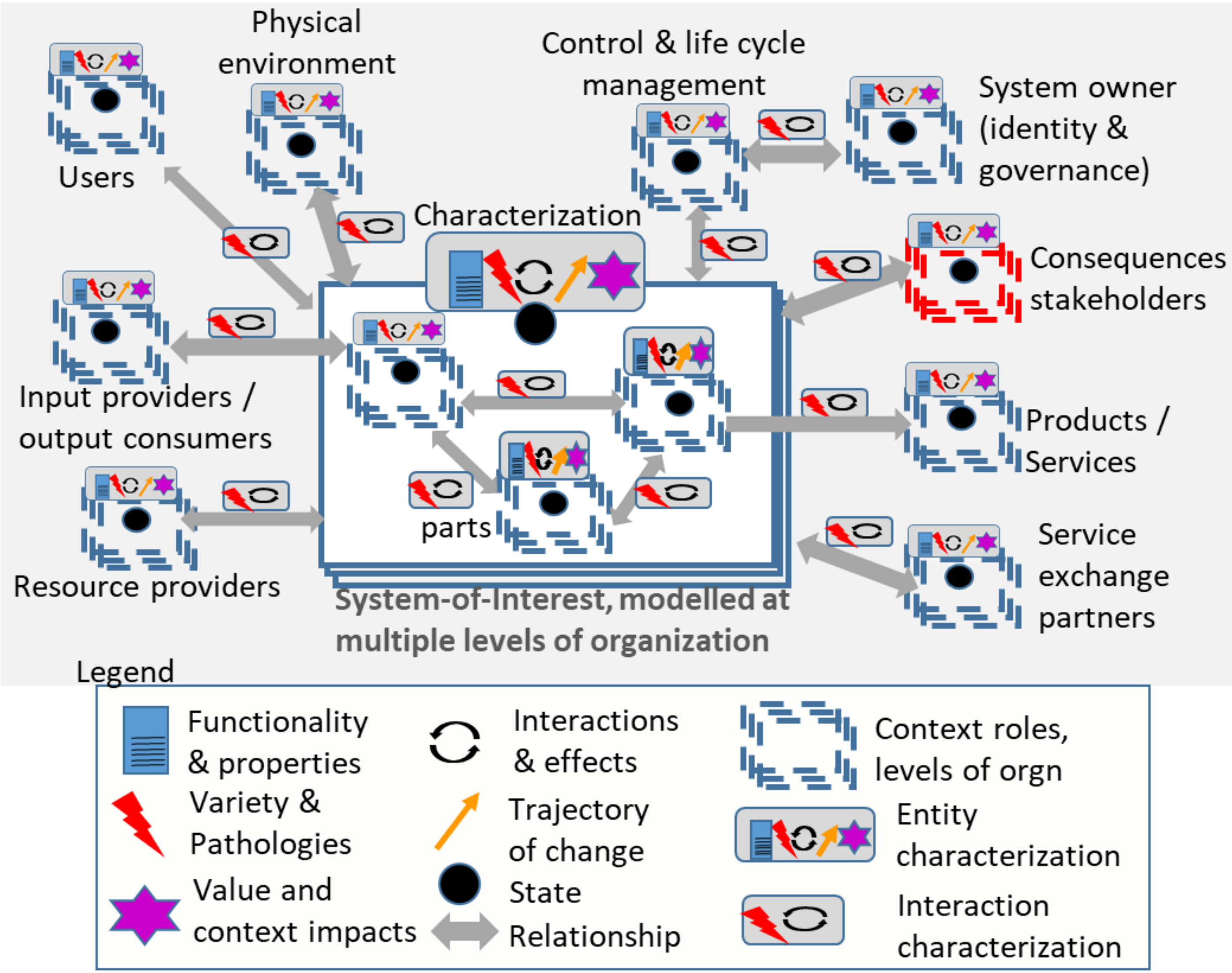
Only a few of the processes shown in each plane

In mechanistic systems, most higher-order functions performed by ecosystem.

Living and purposeful systems increasingly incorporate higher-order functions within them – this is the nature of purposeful behaviour

- VSM (Beer), Mobus, Swami

Block Modelling: Generic Model of a Block



Description of particular entity in system hierarchy, at a particular level of abstraction

Levels of organization include functional, technical, technological – relates to ontogenetic levels (levels of system description)

Block model contents dictated by compositionality model

Ideally create as reusable block models at domain knowledge level

Modelling Behaviour

Linking Actions to Consequences

- Consider

  CloseValve() → *FlowRate* = 0

  TurnOnLight() → Room becomes brighter

  InitiateTransaction() → *ResponseReceived* or *ErrorMessage* or *Timeout*

- Each time we initiate actions, as engineers we have clear expectations of action outcomes
  - Typically modelling languages do not provide ability to express expected outcomes
  - Only in requirements, test cases, simulations (consequences of actions)
- Reason
  - Behaviour is contingent on context! Bulb may burnout, valve may leak…
  - Consequences can only be expectations (assertions)
  - Or subject to context assumptions
- Without action-consequence, miss linkage between system model actions and observed behaviour
- Our systems modelling approach includes context assumptions, paves the way for assertions of expected consequences
  - Which can be validated against actual observed behaviour

Modelling Behaviour Data

- Observations data can be modelled as *obs*(t)[*seq*]
  - Where t is a timestamp
  - *Seq* is a thread sequence identifier for discrete actions
    - E.g. transaction initation, transaction received, transaction performed, response sent, response received
    - Each observable action annotated with a transaction id (sequence identifier)
  - Continuous system observations can carry sequence identifier of initiating event or process instance
- Sequence identifiers enable us to correlate related information
  - Critical for linking behaviour to systems models and knowledge
  - Engineering practice includes sequence identifiers, but implementation is uneven. Often challenging to work out sequencing.
  - Needs to become standard theory and practice with tooling support
- What we are really doing, we are focusing on occurrences information (runtime event sequence) as the missing link between system models and observable behaviour

Observable Behaviour & Experience Modelling

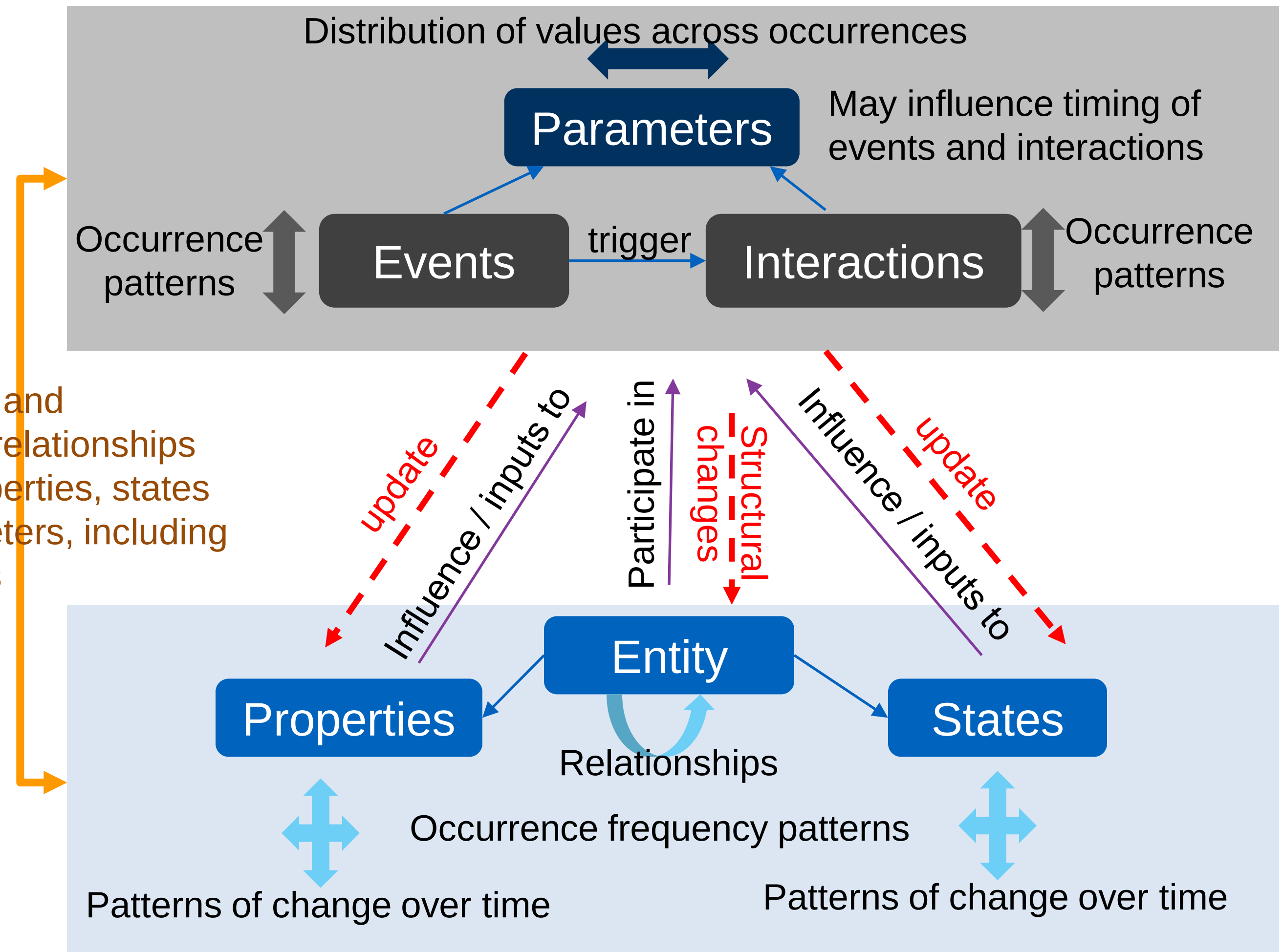
- "When I change channels on the TV, screen goes blank, then sound cuts over, then picture appears"
  - ChangeChannels(t)[*seq*] → *screen*(t+t_1, *seq*) = *blank_screen*, *sound*(t+t_2,*seq*) = *target_channel.sound(t+t_2)*, picture(t+t_3, *seq*) = *target_channel.picture*(t+t_3)
  - Where $t_1 < t_2 < t_3$ (even though t1, t2, t3 may be unspecified)

- A structured vocabulary for expressing experience
  - As machine-checkable assertions
  - Linkable to engineering models
- Parametric equations are of this form
  - *Force*(t) = *mass* * *acceleration*(t)
- An observable behaviour model is a network of relationships among observable quantities *obs*(t)[*seq*]
  - A vocabulary to express desired / expected / actual behaviour in structured form

Some of this work is the subject of TCS patenting

Modelling Occurrence Patterns: Concept

Constraints and parametric relationships among properties, states and parameters, including conditionals



Model system behaviour outcomes

Change over time arising from network of outcomes

Model system structure and patterns over time

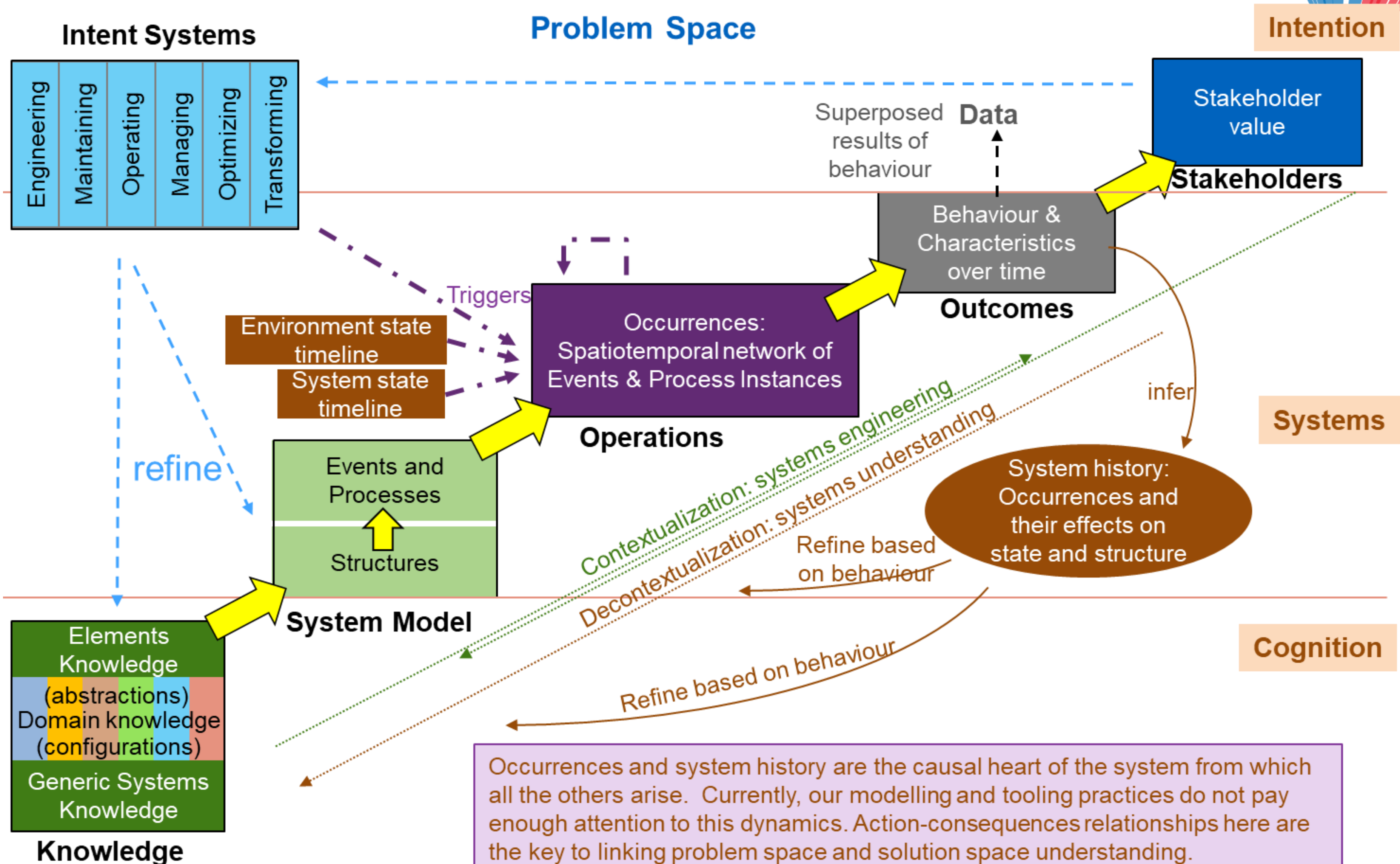
Indicate uncertainties and incompleteness of information

Given observations data sets, we can check these assertions

Some of this work is the subject of TCS patenting

Contextualization & De-Contextualization

Knowledge to Data: Generative Levels



Occurrences and system history are the causal heart of the system from which all the others arise. Currently, our modelling and tooling practices do not pay enough attention to this dynamics. Action-consequences relationships here are the key to linking problem space and solution space understanding.

Solution Space

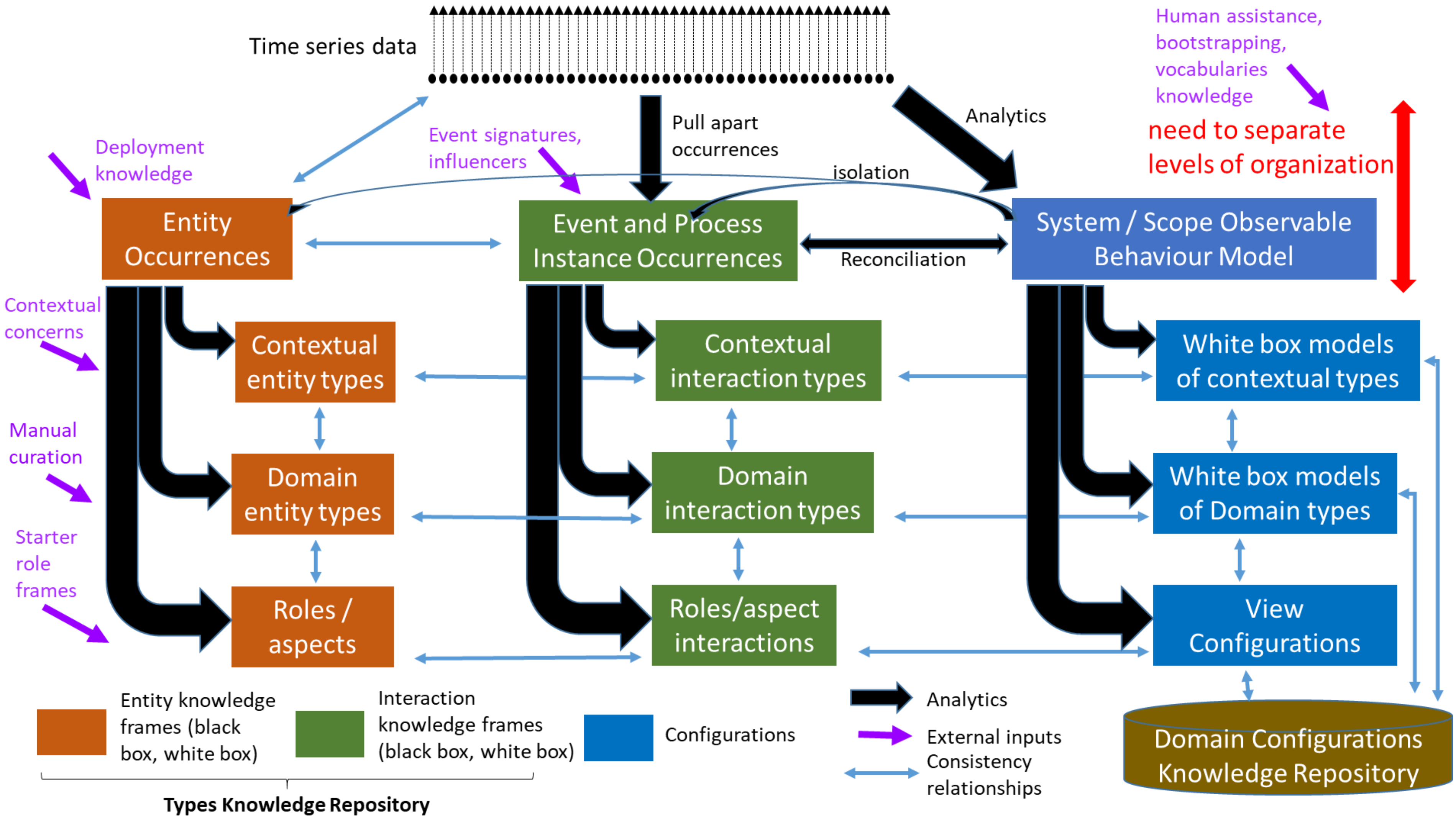
Contextualization & De-Contextualization Processes

Black box behaviours	Time series data about observable behaviour	Time series data about entities, properties, states, interactions, flows, events, relationships, structures		
	Pull apart overlaid behaviours Separate levels of organization	**Observations of behaviour** ↑	**Combines sources, viewpoint & perception filters**	
Deployed system	Occurrences	Entity instance with bindings to its deployment context	Process occurrence that produces particular behaviours	Model of deployed system and history as network of occurrences
	Generalize across deployment contexts	**Configure to deployment context** ↑	**Bind to operational environment(s)**	
System models	Instances and contextual types	Entity instances of project / organization-specific types	Contextual process definitions and behaviours: effects on state	White box decomposition model of each entity, each interaction
	Commonize across instances	**Filter to contextual concerns, select patterns** ↑	**Synthesize patterns, bind into network of elements, bind and harmonize across levels of organization**	
Vocabulary, associated knowledge	Commonizations (domain types)	Hierarchy of domain knowledge type variants with associated concerns, properties, behaviours, addressing range of concerns	Domain vocabulary of actions, with associated roles. Effects of each action on state and structure of participant roles.	White box domain model of each entity and interaction, each with range of concerns, and synthesized combination of solution patterns
	Map to particular viewpoint	**Synthesize pattern choices for each concern** **Alternative pattern choices → type variants** ↑	**Binding of white box parts to roles in each pattern**	
"Theory" Knowledge domains	Abstractions (viewpoint roles, interactions)	e.g. signal receptor, physical object, power load, query operation Same entity / operation described from different viewpoints.		Knowledge about configurations: patterns, parametrics, concepts etc

Engineering involves contextualization & Synthesis

Data → Knowledge involves de-contextualization, commonization, abstraction

Framework for De-Contextualization



Summary and Next Steps

- We have presented a framework for linking knowledge, systems models and data
  - Type DAG concept for organizing knowledge as de-contextualization levels
  - Knowledge frames concept for capturing knowledge about an entity
    - In a form consistent with systems knowledge
  - Generic block model for systems models
  - With context role profiles to capture assumptions, enabling modular models
  - *Obs(t)[seq]* approach to labelling behaviour data
  - Enabling models of observable behaviour, relatable to systems models
- The schema fits together as a series of generative levels
  - Linked by contextualization / de-contextualization relationships
  - Proposed a framework for de-contextualization (and contextualization)
- Translating the framework to reality requires extensive tooling
  - Including working out how to provide automation support for de-contextualization
  - PBSE (Pattern-based systems engineering) points the way to tool support for contextualization

30th Annual INCOSE
international symposium

Virtual Event
July 20 - 22, 2020

www.incose.org/symp2020