Domain Analysis & Description

A Tutorial*

Dines Bjørner

Technical University of Denmark
Fredsvej 11, 2840 Holte, Denmark
bjorner@gmail.com

September 3, 2025

Abstract. We present a summary of a domain analysis & description method. Domains are the realm in which [large scale] software is embedded – in order to serve human actions in predominantly man-made"surroundings". The method, with its *principles, procedures, techniques* and *tools*, are outlined. A main principle is that of delineating observable **phenomena** into describable **entities**; these into **endurants** and **perdurants**, i.e., roughly speaking "statically" and "dynamically" observable **entities**; entities into **endurants** and **perdurants**; endurants into **solids** and **fluids**; solids into **parts** and **living species**; parts into **atomic** and **compound parts**; and compound parts into **Cartesians** and **part sets**. Endurants are then "endowed" with **unique identities, mereologies, attributes** and **intentional "pull"**. Endurants are then, by transcendental deduction, "morphed" into **perdurants: behaviours** that *communicate*, and where unique identities, mereologies and attributes serve as possible updateable behaviour arguments.

The Triptych Dogma

In order to *specify* $\mathbb{S}$*oftware*, we must understand its $\mathbb{R}$*equirements*.
In order to *prescribe* $\mathbb{R}$*equirements* we must understand the $\mathbb{D}$*omain*.
So we must **study, analyze** and **describe** $\mathbb{D}$*omains*.
$\mathbb{D}$,$\mathbb{S} \models \mathbb{R}$:
In **proofs** of $\mathbb{S}$*oftware* correctness,
with respect to $\mathbb{R}$*equirements*,
assumptions are made with respect to the $\mathbb{D}$*omain*

1 Introduction

We encourage the reader to carefully study the above triptych[1] – and the abstract with its *slanted text* and **bold face** highlighted terms. We are not concerned with computing. We are concerned with software nor with requirements to software. Computability and correctness of software is not our concern.

We are concerned with understanding domains such as *railways, insurance, banking, retail and wholesale trading, health care, container terminal ports*, etcetera. How can we analyze and describe domains ? That is our concern. So we propose a rigorous method for analyzing and describing domains.

• • •

We structure this paper in a perhaps unusual form.Instead of compact paragraphs interspersed with definitions cum characterizations, examples, etc., You shall mostly find itemized and enumerated statements. For a more conventional presentation form we refer to the [much] longer 37 page [8].

2 Domains

Characterization 1 *Domain*:

By a *domain* we shall understand a *rationally describable* segment of a *discrete dynamics* fragment of a *human directed & assisted* reality:

* This paper is the basis for an invited tutorial for ICTAC 2025, Marrackesh, Morocco, 24–29 Noember 2025. It is 'derived' from Domain Analysis & Description – `https://www.imm.dtu.dk/ dibj/2025/ictac/main.pdf`.

[1] The domain modeling approach of this paper has been extensively covered in books and lectures notes[2, 4, 6]. The present paper is derived from [8].

- the world that we daily observe
- in which we work and act –
- a reality made significant by human-created entities ▪

Characterization versus Definition

- It is important to observe that we use the term 'characterization' and not the term 'definition'.
- The reason is the following:
  - The describable concepts of the domains that we wish to delineate / encircle are not formal[2].
  - Were they formal, then we could use the term 'definition'.
  - The aim of a 'domain description' is to formalize an instance of a domain.
  - But the formal instances do not mean that the underlying concepts are formal.

An Aside: From Algorithmics to Domains

- "In the beginning" there were **algorithms**
- About 1948 came the von Neumann **computers**
- 1960s: Focus was on **software** implementing algorithms on data
- Late 1970s" **requirements**
- 2010s: **domain engineering**

Domain Engineers face the $\mathbb{D}$*omain*.

– end of an aside

Informal Example 1 *Some Domain Examples*: [3]

- **Rivers:** sources, deltas, tributaries, waterfalls, etc., and their man-made dams, harbours, locks, etc. – and their conveyage of materials (ships, barges, etc.) [7, *Chapter B*].
- **Road nets:** street segments and intersections, traffic lights and automobiles – and the flow of these, etc [7, *Chapter E*].
- **Pipelines:** liquids (oil, gas, or water), wells, pipes, valves, pumps, forks, joins and wells and the flow of fluids, etc. [7, *Chapter I*].
- **Container terminals:** – container vessels, containers, cranes, trucks, etc. – and the movement of these [7, *Chapter K*]
- **Retailing:** customers, shops, distributors, manufacturers, ... ▪

Characterization 1Domainsmycharacterization.1 relies on the understanding of the terms

- *'rationally'*
- *'discrete'*
- *'human'*

By **rationally describable** we mean that what is described can be understood, including reasoned about, in a rational, that is, logical manner – in other words **logically tractable**. By **discrete dynamics** we imply that we shall basically rule out such domain phenomena which have properties which are continuous with respect to their time-wise, i.e., dynamic, behaviour. By **human-directed & assisted** we mean that the domains – that we are interested in modeling – have, as an important property, that they possess man-made and utilized entities.

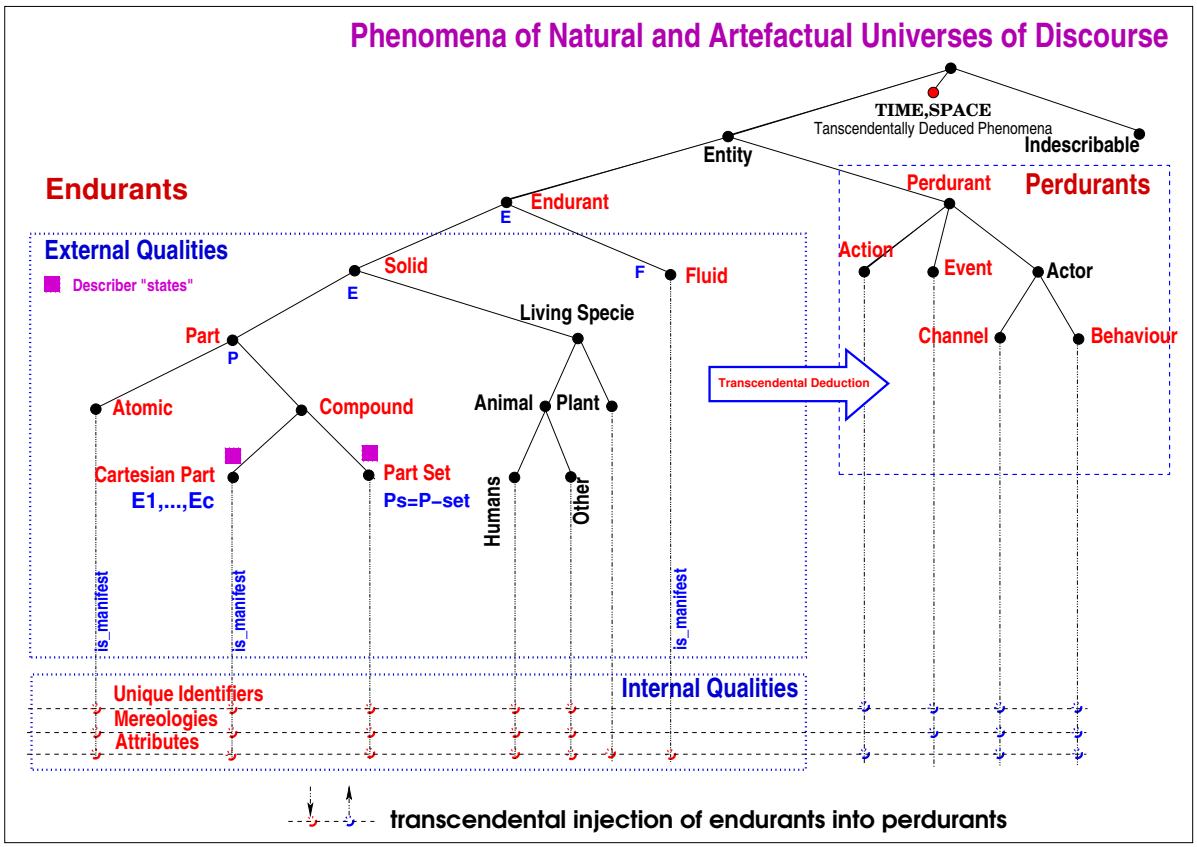
[2] The describable/underlying concepts are those of *entities, endurants, perdurants, solids, fluids, parts, living species, atomic parts, compound parts, Cartesians, part sets*, etc. These concepts will all be 'characterized'.

[3] Some examples are informal, as is this, some are "formal". We shall alert You to the formal ones!

3 A Domain Modeling Analysis & Description Ontology

So how do we approach analyzing and describing the kind of domains that we attempted to outline above ? We propose an altogether new approach. It is partly motivated by the philosophy of Kai Sørlander, a Danish philosopher [10]. The approach, as already revealed in the **abstract**, consists of inquiring, when You, as a domain analyzer cum describer, physically observe a domain and mentally reflect on what You observe in that domain: which are the phenomena; which of these are rationally describable, i.e., are entities, and, of the entities, which are endurants, i.e., somehow "statically" observable, and which are perdurants, i.e., somehow "dynamically" observable, etc.

The terms: phenomena, entities, endurants, perdurants, etc., will be explained now in detail. Their ontological relationship is captured in Fig. 1*figure.1.1.



. *

A Domain Modeling Analysis & Description Ontology

4 Phenomena and Entities. Endurants and Perdurants

The are "things" in domains we can rationally describe, and there are "things" we cannot, at present, rationally describe.

4.1 Phenomena

Characterization 2 *Phenomena*:

- By a *phenomenon*

we shall understand a fact
that is observed to exist or happen ▪

Some phenomena are rationally describable – to some degree – others are not.

Informal Example 2 *Phenomena*: For a transport domain we identify the following phenomena: *trains, unpleasant smell of automobile exhaust, the flight of an aircraft* ▪

4.2 Entities

Characterization 3 *Entities*:

- By an entity an *entity*
- we shall understand a [more-or-less]
- rationally describable phenomenon ▪

Informal Example 3 *Entities*: For a transport domain we identify the following entities: *the way bill* and *bill of lading* for a transport, the *inquiry* as to a transport of specific goods, the *departure* of a train ▪

- **Prompt 1** *is_entity(φ)*:
  - `is_entity`(ϕ) holds
  - for phenomenon ϕ
  - if ϕ is describable ▪

By a prompt (*cue*[4], *schlüsselwörter, mots-clés, spunto, ...)* we shall here understand: a mental note – something for the domain analyze & describer to do – according to the domain analysis & description ontology.

4.3 Endurants

Characterization 4 *Endurants*:
Endurants are those quantities of domains that we can observe (see and touch), in *space*,as "complete" entities at no matter which point in *time* – "material" entities that persists, endures – capable of enduring adversity, severity, or hardship [Merriam Webster]▪

Endurants are either *natural* ["God-given"] or *artefactual* ["man-made"]; and either **solid** or **fluid**; and either *manifest*, or *conceptual*; and either *mobile*, or *immobile* – or are *immobile* but can be moved!

Informal Example 4 *Endurants*: In a transport domain we can identify the following endurants:*streets, street intersections, automobiles, trucks, buses, rails, trains, sea, container vessels, air and aircraft* ▪

Endurants are:

- **"God-given" vs. Man-made:**
  - Lakes, rivers, mountains, fish, and roses – are "God-given".
  - Roads, automobiles and aircraft – are man-made.
- **Solid vs. Fluid:**
  - An automobile and a mountain is solid.
  - The milk in a carton, and the water in a lake is fluid.
- **Manifest vs. Conceptual:**
  - An automobile is manifest.
  - The "assembly" of automobiles and roads is seen as conceptual.
- **Mobile vs. Immobile:**
  - A ship is mobile.
  - A road is immobile.
  - Most cargo on a ship, or on-shore, is immobile – but can be moved!

- **Prompt 2** *is_endurant(e)*:
  - `is_endurant` holds
  - for entity `e`
  - if `e` is an endurant ▪
  - **pre:** `is_entity(e)`

[4] cue: thing said or done that serves as a signal to an actor or other performer to enter or to begin their speech or performance.

4.4 Perdurants

Characterization 5 *Perdurants*: Perdurants are those quantities of domains for which only a fragment exists, in *space*, if we look at or touch them at any given snapshot in *time* ▪

Perdurants are here considered to be *actions*, *events* and *behaviours*.

Informal Example 5 *Perdurants*: In a transport domain we can identify the following perdurants: *moving automobiles, moving trucks, moving trains, moving ships, moving aircraft.* ▪

- **Prompt 3** `is_perdurant(e)`:
  - `is_perdurant(e)` holds
  - for entity `e`
  - if `e` is a perdurant ▪
  - **pre:** `is_entity(e)`

5 External and Internal Endurant Qualities

Characterization 6 *External Qualities*: External qualities of endurants of a manifest domain are, in a simplifying sense, those we can see, touch and have spatial extent. They, so to speak, "take form".

Informal Example 6 *External Qualities*: the Cartesian of sets of solid atomic street intersections, and of sets of solid atomic street segments, and of sets of solid automobiles of a road transport system reflect external qualities ▪

Characterization 7 *Internal Qualities*: Internal qualities are those properties [of endurants] that do not occupy *space* but can be measured or spoken about or have occurred ▪

Informal Example 7 *Internal Qualities*: the distinct identity of each automobile; the [mereological] relations between street segments [links] and intersections [hubs]; the position of an automobile on a street segment; the state of a hub: green–red ▪

6 External Qualities

External qualities of endurants are, simplifying, those that we can see and touch and which have spatial extent.

6.1 The Universe of Discourse

The "outermost" quality of a domain is the "entire" domain – "itself"! Any domain analysis starts by identifying that "entire" domain! We it a name, say UoD, for *universe of discourse*, We describe it, in *narrative* form,
that is, in natural language
containing terms of professional/technical nature, the domain. Finally, *formalizing* just the name:
giving the name "status" of being a type name,
that is, of the type of a class of domains
whose further properties will be described subsequently.

Schema 1 *The Universe of Discourse*

Narration:
The name, and hence the type, of the domain is UoD
The UoD domain can be briefly characterized by ▪▪▪
Formalization:
type UoD ▪

Formal Example 1 *Multi-modal Transport*: [5]

Narration:
The domain is that of multi-modal transport T: land, sea and air,
of goods, G: passengers and merchandise,
by conveyors, C: bus, truck, train, ship and aircraft.
"K"ustomers, K, inquire, order, deliver and receive goods.
Firms, F, offer, confirm order and convey goods.
Conveyors load and unload merchandise at nodes, N,
travel along links, L of a transport net, N, and
keep firms and customers informed by messages, M.
Etcetera, etcetera.

Formalization:
type
T, M, C, K, G, F, ..., N, L, N, M, ...
value
inq, ordr, deliv, recv, offr, conf_ordr, convey, load, unload, travel, inf, ...
axiom
... ∎

6.2 Solid Endurants

Given then that there are endurants we now postulate that they are either [mutually exclusive] *solid* (i.e., discrete) or *fluid*.

Characterization 8 *Solid Endurants*:

- By a *solid* endurant
  - we shall understand an endurant
  - which is separate, individual or distinct in form or concept,

  or, rephrasing,
  - have body (or magnitude) of three-dimensions:
    - length/height,
    - breadth/width and
    - depth ∎

Informal Example 8 *Solid Endurants of a Pipeline System*: Some are: wells, pipes, pigs, valves, pumps, forks, joins and sinks ∎

Prompt 4 `is_solid`:

- `is_solid(e)` holds
- for endurant `e`
- if `e` is solid ∎
- **pre:** `is_endurant(e)`

6.3 Fluids

Characterization 9 *Fluid Endurants*: By a *fluid endurant* we shall understand an endurant which is prolonged, without interruption, in an unbroken series or pattern; [] or, rephrasing: a substance (liquid, gas or plasma) having the property of flowing, consisting of particles that move among themselves ∎

Informal Example 9 *Fluid Endurants*: Examples of fluid endurants are: *water, oil, gas, compressed air, smoke*∎

Fluids are otherwise liquid, gaseous, plasmatic, granular, or plant products, et cetera.

Prompt 5 `is_fluid`: `is_fluid(e)` holds for endurant `e` if `e` is fluid ∎
pre: `is_endurant(e)`

[5] This example is listed as 'formal' – although it is mostly "sketchy informal" !

6.4 Parts and Living Species Endurants

Given then that there are solid endurants we now postulate that [mutually exclusive] they are either *parts* or *living species*.

6.4.1 Parts

Characterization 10 *Parts*: The non-living-species solids are what we shall call parts ▪

Parts are the "work-horses" of man-made domains.

Informal Example 10 *Parts*: *Pipeline Units*: wells, pumps, pipes, pigs, valves, forks, sinks ▪

Prompt 6 *is_part*:

- `is_part(e)` holds
- for solid endurants `e`
- if `e` is a part ▪
- **pre:** `is_solid(e)`

6.4.2 Atomic and Compound Parts

We distinguish between atomic and compound parts.

- It is an empirical fact that
- parts can be composed from parts.
- That possibility exists.
- Hence we can [philosophy-wise] reason likewise.

6.4.2.1 Atomic Parts

Characterization 11 *Atomic Part*:

- By an *atomic part*
- we shall understand a part
- which the domain analyzer considers to be indivisible
- in the sense of not meaningfully consist of sub-parts ▪

Informal Example 11 *Atomic Parts*: hubs, H, i.e., street intersections links, L, i.e., the roads between two neighbouring hubs automobiles, A ▪

Prompt 7 *is_atomic*:

- `is_atomic(p)` to hold
- for parts `p` if
- `p` is atomic ▪
- **pre:** `is_part(e)`

6.4.2.2 Compound Parts

Characterization 12 *Compound Part*: Compound parts are those which are observed to consist of several parts ▪

Informal Example 12 *Compound Parts*:

- A **road net** consists of a **Cartesian** of [○] a **set of hubs**, i.e., street intersections or "end-of-streets", and [○] a **set of links**, i.e., street segments (with no contained hubs) ▪

Prompt 8 *is_compound*:

- `is_compound(p)` holds
- for parts `p`

- if p is a compound ▪
- **pre:** is_part(e)

— Cartesians

Characterization 13 *Cartesians*: Cartesian parts are those compound parts which are observed to consist of two or more distinctly sort-named endurants (solids or fluids) ▪

Formal Example 2 *Road Transport*: [6]

Narrative:

1. A road transport, rt:RT, is abstracted as a Cartesian of
2. a road net, RN and
3. an aggregate of automobiles, SA –
4. where the road net is a Cartesian of a set of hubs, AH,
5. and a set of links, AL.
6. An aggregate of automobiles is a set of automolbiles.
7. Automobiles are here considered atomic.

Formalization:

type
1Compound PartsItem.1. RT
2Compound PartsItem.2. RN
3Compound PartsItem.3. SA
4Compound PartsItem.4. AH = H-**set**
5Compound PartsItem.5. AL = L-**set**
6Compound PartsItem.6. AS = A-**set**
7Compound PartsItem.7. A
value
2Compound PartsItem.2. **obs**_RN: RT → RN
3Compound PartsItem.3. **obs**_SA: RT → SA
4Compound PartsItem.4. **obs**_AH: RN → AH
5Compound PartsItem.5. **obs**_AL: RN → AL
6Compound PartsItem.6. **obs**_AS: SA → AS ▪

Prompt 9 *is_Cartesian*: is_Cartesian(p) holds for compound parts p if p is Cartesian ▪
pre: is_compound(e)

A Cartesian part, say p:P, consists of two or more endurants. Which are the type names of the endurants of which it consists? The inquiry: record_Cartesian_part_type_names(p:P), yields the *type names* of the constituent endurants.

Prompt 10 *record-Cartesian-part-type-names*:

```
value
  record_Cartesian_part_type_names: P → 𝕋-set
  record_Cartesian_part_type_names(p) as {ηE1,ηE2,...,ηEn} ▪
```

Here 𝕋 is the **name** of the type of all type names, and ηEi is the **name** of type Ei.

Informal Example 13 *Cartesian Parts*:

- The Cartesian parts of a road transport, rt:RT, consists of
  - an aggregate of a road net, rn:RN, and
  - an aggregate set of automobiles, sa:SA:
- That is:
  - record_Cartesian_part_type_names(rt:RT) = {ηRN,ηSA}
  - record_Cartesian_part_type_names(rn:RN) = {ηAH,ηAL} ▪

— Part Sets

Characterization 14 *Part Sets*:

- Part sets are those compound parts

[6] This example is 'formal' in the sense that it adheres to the *narrative/RSL formalization* dogma.

- which are observed to consist of
- an indefinite number of zero, one or more parts ▪

Prompt 11 *is_part_set* :

- `is_part_set(p)` to holds
- for compound parts `e`
- if `e` is a part set ▪
- **pre:** `is_compound(e)`

The inquiry: `record_part_set_part_type_names`, yields the (single) type of the constituent parts.

Prompt 12 *record-part-set-part-type-names*:

```
value
  record_part_set_part_type_names: E → TPs×TP
  record_part_set_part_type_names(e:E) as (ηPs,ηP) ▪
```

Example 1. *Part Sets: Road Transport*: The road transport contains a set of automobiles. The part set type name has been chosen to be SA. It is then determined (i.e., analyzed) that SA is a set of Automobile of type A

- `record_part_set_part_type_names(sa:SA)` = (ηAs,ηA) ▪

6.4.2.3 Compound Observers

Prompt 13 *describe_compound(p): P → RSL*-**Text**:

```
value
  let {ηP1,ηP2,...,ηPn}=record_Cartesian_part_type_names(e:E) in
  " type
        P1, P2, ..., Pn;
    value
        obs_P1: E→P1, obs_P2: E→P2,...n obs_Pn: E→Pn "

  let (ηPs,ηP) = record_part_set_part_type_names(e:E) in
  " type
        P, Ps = P-set,
    value
        obs_Ps: E→Ps "
  end end ▪
```

6.5 States

Characterization 15 *States*:

- By a *state*
- we shall mean any subset of the parts of a domain ▪

Formal Example 3 *Road Transport State*:

8. There is the set of all hubs,
9. and the set of all links,
10. and the set of all automobiles.
11. The union of these form a state.

```
variable
??.  hs:AH := obs_AH(obs_RN(rt))
??.  ls:AL := obs_AL(obs_RN(rt))
??.  as:SA := obs_AS(obs_SA(rt))
??.  σ:(H|L|A)-set := hs∪ls∪as ▪
```

6.6 Summary of Endurant Prompts

- is_entity
- is_endurant
- is_perdurant
- is_solid
- is_fluid
- is_part
- is_atomic
- is_compound
- is_Cartesian
- is_part_set

6.6.2 Description Prompts

- record_Cartesian_part_type_names
- record_part_set_part_type_names
- describe_compound: Cartesians, Part Sets

7 Internal Qualities – Intangibles

Characterization 16 *Internal Qualities*: Internal qualities are those properties [of endurants] that do not occupy *space* but can be measured or spoken about ▪

Example 2. *Internal qualities*: Examples of internal qualities are **uid_**: the *unique identity* of a part, **mereo_**: the *mereological relation* of parts to other parts, and **attr_**: the attribute query of endurants ▪

7.1 Unique Identity

Characterization 17 *Unique Identity*: An immaterial property that distinguishes any two *spatially* distinct solids. The unique identity of a part p of type P is obtained by the postulated observer **uid_**P:

Schema 2 *Describe-Unique-Identity-Part-Observer*:

" **type**
P,PI
value
uid_P: P → PI " ▪

Here PI is the type of the unique identifiers of parts of type P.

Formal Example 4 *Unique Road Transport Identifiers*:
The unique identifierss of a road transport, rt:RT, is here limited:

12. each hub has a unique identifier,
13. each link has a unique identifier, and
14. each automobile has a unique identifier.

type
??. HI
??. LI
??. AI

value
??. **uid_**H: H → HI
??. **uid_**H: L → LI
??. **uid_**H: A → AI ▪

Schema 3 *Describe-Unique-Identifiers*:

let {η P1,η P2,...,η Pn} = **record_domain_part_type_names**(p:P) **in**
" **type**
P1I, P2I, ..., PnI;
value
uid_P1: P1→P1I, **uid_**P2: P2→P2I,..., **uid_**Pn: Pn→PnI "
end ▪

We have thus introduced a core domain modeling tool the **uid_**... observer function, one to be "applied" mentally by the domain describer. The **uid_**... observer function is "applied" by the domain describer. It is not a computable function.

No two parts have the same unique identifier.

Formal Example 5 *Road Transport Uniqueness*:
The unique identifiers of a road transport, rt:RT, consists of the unique identifiers of

15. the set of all hub identifiers,
16. the set of all link identifiers,
17. the set of all automobile identifiers.
18. Together they form a unique identifier state.
19. There are as many hubs, links and automobiles as there are hub, link and automobile identifiers.

variable
?? hs_{uids}:HI-**set** := { **uid_**H(h) | h:H • h ∈ σ}
?? ls_{uids}:LI-**set** := { **uid_**L(l) | l:L • l ∈ σ}
?? as_{uids}:AI-**set** := { **uid_**A(a) | a:A • a ∈ σ}
?? σ_{uids}:(HI|LI|AI)-**set** := $hs_{uids} \cup hs_{uids} \cup hs_{uids}$
?? **card**σ = **card**σ_{uids}

7.2 Mereology

The concept of mereology is due to the Polish mathematician Stanisław Leśniewski (1886–1939)

Characterization 18 *Mereology*: Mereology is a theory of the relations of an [endurant] parts to a whole and the relations of [endurant] parts to [endurant] parts within that whole ▪

From Mereology to Communication Channels

We shall analyze and describe: narrate and formalize the mereology of manifest parts. This form of description serves to explain how parts relate to one another. These relationships "reappear" in the part-perdurant behaviours in the form of CSP-like communications over channels between mereologically prescribed sub-channels.
Mereologies can be expressed in terms of unique identifiers.

Formal Example 6 *Road Traffic Mereology*: We shall be concerned onlt with the mereology of some manifest parts.

20. The mereology of links is a 2 element set of hub identifiers of the road net[7].
21. The mereology of a hub is a possibly empty set of hub identifiers of the road net.
22. The mereology of an automobile is [some subset of] a set of hub and link identifiers[8]

type
?? ML = LI-**set axiom** ∀ ml:ML • **card** ml = 2 ∧ ml ⊆ ls_{uis}
?? MH = HI-**set axiom** ∀ mh:MH • mh ⊆ hs_{uis}
?? MA = (HI|LI)-**set axiom** ∀ ma:MA • ma ⊆ as_{uis}
value
?? **mereo_**L: L → ML
?? **mereo_**H: H → MH
?? **mereo_**A: A → MA ▪

In general:

Schema 4 *Describe-Mereology*:

" **type**
PMer = $\mathcal{M}$(PI1,PI2,...,PIm)
value
mereo_P: P → PMer
axiom
$\mathcal{A}$(pm:PMer) " ▪

where $\mathcal{M}$(PI1,PI2,...,PIm) is a type expression over unique identifier types of the domain; **mereo_**P is the mereology observer function for parts p:P; and $\mathcal{A}$(pm:PMer) is an axiom that secures that the unique identifiers of any part are indeed of parts of the domain ▪

[7] This is a simplified version: it allows for automoblie traffic in both directions of the link. We leave it to the reader to "cook" up othe such traffic possibilities.
[8] – a full set means that the specific automobile is allowed to travel all over the net.

7.3 Attributes

Attributes are what finally gives "life" to endurants: The external qualities "only" named [i.e., typed] and gave structure to their atomic or compound types. The internal qualities of uniqueness and mereology are intangible quantities. The internal quality of attributes gives "flesh & blood" to endurants: they let us express endurant properties that we can more easily, i.e., concretely, relate to.

Characterization 19 *Attributes*: are properties of endurants that can be measured either physically or can be objectively spoken about ▪

Attributes are of types and, accordingly have values.

An informal domain analysis function, `record_attribute_type_names`: analyzes parts, p:P, into the set of attribute names of parts p:P

Schema 5 *record-attribute-type-names*:

```
value
  record_attribute_type_names: P → ηT-set
  record_attribute_type_names(p:P) as ηT-set ▪
```

Formal Example 7 *Road Net Attributes, I*:

Example attributes are:

23. Hubs have states, hσ:HΣ: the set of pairs of link identifiers, ($f$li,$t$li), of the links $f$rom and to which automobiles may enter, respectively leave the hub.
24. Hubs have state spaces, hω:HΩ: the set of hub states "signaling" which states are open/closed, i.e., green/red.
25. Links that have lengths, LEN; and
26. Automobiles have road net positions, APos,
27. either *at a hub*, atH,
28. or *on a link*, onL, some fraction, f:**Real**, down a link, identified by li, from a hub, identified by fhi, towards a hub, identified by thi.
29. Links have states, lσ:LΣ: the set of pairs of link identifiers, ($f$li,$t$li), of the links $f$rom and to which automobiles may enter, respectively leave the hub.
30. Links have state spaces, lω:LΩ: the set of link states "signaling" which states are open/closed, i.e., green/red.
31. Hubs, links and automobiles have *histories:* time-stamped, chronologically ordered sequences of automobiles entering and leaving links and hubs, with automobile histories similarly recording hubs and links entered and left.
32. Link positions have well-defined identifiers and fractions.

```
type
??. HΣ = (LI×LI)-set
??. HΩ = HΣ-set
??. LEN = Nat
??. APos = atH | onL
??.    atH :: HI
??.    onL :: LI × (fhi:HI × f:Real × thi:HI)
??. LΣ = (HI×HI)-set
??. LΩ = LΣ-set
??. HHis,LHis = (TIME×AI)*
??. AHis =  (TIME×(HI|LI))*
```

```
value
??. attr_HΣ: H → HΣ
??. attr_HΩ: H → HΩ
??. attr_LEN: L → LEN
??. attr_APos: A → APos
??. attr_LΣ: L → LΣ
??. attr_LΩ: L → LΩ
??. attr_HHis: H → HHis
??. attr_LHis: L → LHis
??. attr_AHis: A → AHis
```

```
axiom
??. ∀ mk_onL(li,(fhi,f,thi)):onL • 0<f<1  ∧ li∈$ls_{uids}$ ∧ {fhi,thi}⊆$hs_{uids}$ ∧ ...▪
```

Schema 6 *Describe-endurant-attributes(e:E)*:

let {η A1,ηA2,...,ηAn} = `record_attribute_type_names`(e:E) **in**
" **type**
A1, A2, ..., An
value
attr_A1: E → A1, **attr_**A2: E → A2, ..., **attr_**An: E → An
axiom
∀ a1:A1, a2:A2, ..., an:An: $\mathcal{A}$(a1,a2,...,an) "
end ▪

7.4 Intentional Pull

Two or more parts of different sorts, but with overlapping sets of intents[9] may excert an intentional "pull" on one another. This *intentional "pull"* may take many forms. Let $p_x : X$ and $p_y : Y$ be two parts of *different sorts* (X, Y), and with *common intent*, ι. *Manifestations* of these, their common intent must somehow be *subject to constraints*, and these must be *expressed predicatively*.

Example 3. *Road Transport Intentionality*: *Automobiles* include the *intent:* `transport`, as do *hubs* and *links*. *Manifestations* of `transport` are reflected in *hubs, links* and *automobiles* having the *history* attribute. The *intentional "pull"* of these manifestations is this: For every automobile, if it records being in some hub or on some link at time τ, then the designated hub, respectively link, records exactly that automobile; and vice versa: for all hubs [links], if it records the visit of some automobile at time τ, then the designated automobile records exactly that hub [link] ▪

Example 4. *Double-entry Bookkeeping*: Another example of intentional "pull" is that of double-entry bookkeeping. The *incomes/expenses ledger* must *balance* the *actives/passives ledger* ▪

Example 5. *The Henry George Theorem*: states that under certain conditions, *spending* by government on *public goods* will increase *rent* based on *land value* more than that amount, with the benefit of the last marginal investment equaling its cost ▪

For example: *Increase in land value around a new bridge "equals" the cost of the bridge.*

8 Transcendental Deduction

8.1 Some Characterizations

Characterization 20 *Transcendental*:

- By transcendental we shall understand
  - the philosophical notion:
  - the a priori or intuitive basis of knowledge,
  - independent of experience ▪

Characterization 21 *Transcendental Deduction*:

- By a transcendental deduction we shall understand
  - the philosophical notion:
  - a transcendental "conversion"
  - of one kind of knowledge
  - into a seemingly different kind of knowledge ▪

[9] Intent: purpose; God-given or human-imposed!

8.2 On Manifest Deductions

Definition 1 *Manifest Parts*: By a manifest part we shall understand a part which we have endowed with internal qualities: unique identification, mereology and attributes ▪

That is: You, the domain analyzer cum describer decides which are the manifest parts and which are not the manifest parts

Informal Example 14 *Manifest Road Traffic Parts*: We decide, for our "running" road traffic formal example that the manifest parts are those of hubs, links, and automobiles ▪

Comments:
We could have chosen otherwise. We could, for example, have chosen the aggregate of automobiles to be manifest and represent, for example, either the department of vehicles, or a nation-wide automobile club!

9 Perdurants

- We shall deploy the notion of transcendental deduction when
  - * "moving" from **endurant** parts
  - * to **perdurant** behaviours!
- And we shall apply transcendental deduction only to manifest parts.

9.1 Actions

- Actions [instantaneously] change state.
- Actions are prescribed.

9.2 Events

- Events [instantaneously] change state.
- Events are not planned.
- They "do so" surreptitiously.

9.3 Behaviours

Characterization 22 *Behaviours*: Behaviours are sets of sequences of actions, events and behaviours – and take place "over time"! ▪

Concurrency is modeled by the *sets* of behaviours.Synchronization and communication of behavioursare effected by CSP *output/inputs*:

- ch[{i,j}]!value – and
- ch[{i,j}]?.

Informal Example 15 *Road Net Traffic*: Road net traffic actions: of **automobiles:** start, stop,turn right, turn left, etc.; of **links:** automobiles entering, leaving, and move on the link, etc; of **hubs:** automobiles entering, leaving, and move, etc. within the hub; etc.

9.4 Channel

Characterization 23 *Channel*: A channel is anything that allows synchronization and communication of values between behaviours▪

Schema 7 *Channel*:

We suggest the following schema for describing channels:

" **channel** { ch[{ui,uj}] | ui,ij:UI • ... } M "

where ch is the describer-chosen name for an array of channels; ui,uj are channel array indices of the unique identifiers; UI, of the chosen message domain ▪

Formal Example 8 *Road Transport Interaction Channel*:

33. There is a set of channels between hubs, links and automobiles.
34. These channels communicate messages, `M`. `M` will "transpire" frm the behaviour definitions.

```
channel
??. { ch[ {ui,uj} ] | {ui,ij}:(HI|LI|AI)-set • ui≠uj∧{ui,uj}⊆σ_uids } M
type
??. M ▪
```

9.5 Behaviour Signatures

Schema 8 *Behaviour Signature*:

Behaviour signatures[10]reflect the internal qualities of the part endurantsfrom which they emerge by transcendental deduction:

```
value B_p:                                                  name of behaviour
    → Uid_p                                                 its unique identifier
        → Mereo_p                                           mereology
            → Sta_Vals_p                                    static attributes
                → Inert_Vals_p                              inert attributes
                    → Mon_Refs_p                            monitorable attributes
                        → Prgr_Vals_p                       programmable attributes
                            → { ch[{i,j}] | ... }           communication channels
                                → Unit                      "ad infinitum"
```

We do not cover static, inert, monitorable and programmable attributes in this paper!

Formal Example 9 *Road Transport Behaviour Signatures*:

35. The signature of hub behaviours follow the "Schönfinkel'ed pattern" of *unique identifier* → *mereo* → *static attributes* → *programmable attributes* → *channel arrays* and **Unit**.
36. The signature of link behaviours likewise.
37. The signature of automobile behaviours likewise.

We hint at these signatures.

```
value
??. hub: HI
      → MereoH
        → (HΩ×...)
          → (HΣ × HHist × ...)
            → {ch[ {uid_H(p),ai} ]|ai:AI•ai∈as_uid} Unit
??. link: LI
      → MereoL→
        → (LΩ×LEN × ...)→
          → (LΣ × LHist × ...)
            → {ch[ {uid_L(p),ai} ]|ai:AI•ai∈as_uid} Unit
??. automobile: AI
          → MereoA
            → (...)
              → (AVel × HAcc × ... × APos × AHist)
                → {ch[ {uid_H(p),ri} ]|ri:(HI|LI)•ri∈hs_uid∪ls_uid} Unit
```

Here we have suggested additional and omitted some part attributes ▪

[10] We 'Schónfinkel'/'Curry' function sigatures.

9.6 Behaviour Invocation

Schema 9 *Behaviour Invocation*:

Behaviours are invoked as follows:

" B(**uid_**B(p))
(**mereo_**P(p))
(**attr_**staA$_1$(p),...,**attr_**staA$_s$(p))
(**attr_**inertA$_1$(p),...,**attr_**inertA$_i$(p))
(**attr_**monA$_1$(p),...,**attr_**monA$_m$(p))
(**attr_**prgA$_1$(p),...,**attr_**prgA$_p$(p)) "

- All arguments are passed *by value*.
- The *uid* value is never changed.
- The *mereology* value is usually not changed.
- The *static attribute* values are fixed, never changed.
- The *inert attribute* values are fixed, but can be updated by receiving explicit input communications.
- The *monitorable attribute* values are functions, i.e., it is as if the "actual" monitorable values are passed *by name*!
- The *programmable attribute* values are usually changed, "updated", by actions described in the behaviour definition.

9.7 Behaviour Description – An Example

Formal Example 10 *Automobile Behaviour at Hub*:

38. We abstract automobile behaviour at a Hub (hi).
    (a) Either the automobile remains at the hub,
    (b) or, internally non-deterministically,
    (c) leaves the hub entering a link,
    (d) or, internally non-deterministically,
    (e) stops.

?? automobile(ai)(ris)(...)(atH(hi),ahis,_) ≡
?? automobile_remain_at_hub(ai)(ris)(...)(atH(hi),ahis,_)
?? ⌈⌉
?? automobile_leaving_hub(ai)(ris)(...)(atH(hi),ahis,_)
?? ⌈⌉
?? automobile_stop(ai)(ris)(...)(atH(hi),ahis,_)

39. [??] The automobile remains at a hub:
    (a) time is recorded,
    (b) informing the hub behaviour, whereupon
    (c) the automobile remains at that hub, "idling",

?? automobile_remain_at_hub(ai)(ris)(...)(atH(hi),ahis,_) ≡
?? **let** τ = **record_**$\mathbb{TIME}$ **in**
?? ch[{ai,hi}] ! τ ;
?? automobile(ai)(ris)(...)(atH(hi),⟨(τ,hi)⟩^ahis,_) **end**

40. [??] The automobile leaves the hub entering link li:
    (a) time is recorded;
    (b) hub is informed of automobile leaving and link that it is entering;
    (c) "whereupon" the vehicle resumes (i.e., "while at the same time" resuming) the vehicle behaviour positioned at the very beginning (0) of that link.

```
?? automobile_leaving_b(ai)({li}∪ris)(...)(atH(hi),ahis,_) ≡
??    let τ = record_𝕋𝕀𝕄𝔼 in
??    (ch[ {ai,hi} ] ! τ ‖ ch[ {ai,li} ] ! τ) ;
??    automobile(ai)(ris)(...)(onL(li,(hi,0,_)),⟨(τ,li)⟩^ahis,_) end
??     pre: [hub is not isolated]
```

41. [**??**] Or the automobile stops, "disappears — off the radar" !

```
?? automobile_stop(ai)(ris),(...)(atH(hi),ahis,_) ≡ stop ▪
```

9.8 Behaviour Initialization.

Formal Example 11 *Road Transport Initialization*:
We "wrap up" the main example of this tutorial:

42. Let us refer to the system initialization as an action;
43. all hubs are initialized,
44. and
45. all links are initialized,
46. and
47. all automobiles are initialized.

```
value
??. rts_initialisation: Unit → Unit
??. rts_initialisation() ≡
??.    ‖ { hub(uid_H(h))(mereo_H(h))(attr_HΩ(h),...)(attr_HΣ(h),...)| h:H • h ∈ hs }
??.    ‖
??.    ‖ { link(uid_L(l))(mereo_L(l))(attr_LEN(l),...)(attr_LΣ(l),...)| l:L • l ∈ ls }
??.    ‖
??.    ‖ { automobile(uid_A(a))(mereo_A(a))(attr_APos(a)attr_AHis(a),...) | a:A • a ∈ as }
```

10 Conclusion

- This talk was **not** about computers, computing or 𝕊*oftware*.
- This talk was about 𝔻omain *descriptions*. [2, Chapters 3–8]
- From these we develop ℝequirements *prescriptions*. [2, Chapter 9]
- And from requirements we develop 𝕊oftware *designs*. [1]

References

1. Dines Bjørner. *Software Engineering, Vol. 1: Abstraction and Modelling; Vol. 2: Specification of Systems and Languages; Vol. 3: Domains, Requirements and Software Design*. Texts in Theoretical Computer Science, the EATCS Series. Springer, Heidelberg, Germany, 2006.
2. Dines Bjørner. *Domain Science & Engineering – A Foundation for Software Development*. EATCS Monographs in Theoretical Computer Science. Springer, Heidelberg, Germany, 2021. A revised version of this book is [5].
3. Dines Bjørner. *Domain Science & Engineering – A Foundation for Software Development*. EATCS Monographs in Theoretical Computer Science. Springer, Heidelberg, Germany, 2021. A revised version of this book is [6].
4. Dines Bjørner. Domain Modelling – A Primer. A short and significantly revised version of [5]. xii+202 pages[11], May 2023.

[11] This book is currently being translated into Chinese by Dr. Yang ShaoFa, IoS/CAS (Institute of Software, Chinese Academy of Sciences), Beijing and into Russian by Dr. Mikhail Chupilko and his colleagues, ISP/RAS (Institute of Systems Programming, Russian Academy of Sciences), Moscow

5. Dines Bjørner. Domain Science & Engineering – A Foundation for Software Development. Revised edition of [2]. xii+346 pages[12], January 2023.
6. Dines Bjørner. Domain Modelling – A Primer. A significantly revised version of [3]. xii+202 pages[13], Summer 2024.
7. Dines Bjørner. Domain Models – A Compendium. Internet: `http://www.imm.dtu.dk/~dibj/2024/models/domain-models.pdf`, March 2024. This is a very early draft. 19 domain models are presented.
8. Dines Bjørner. Domain Analysis & Description. In *ICTAC 2025 Conference Proceddings*, Lecture Notes in Computer Science. Springer, November 2025.
9. Kai Sørlander. *Den rene fornufts struktur [The Structure of Pure Reason]*. Ellekær, Slagelse, Denmark, 2022. See [10].
10. Kai Sørlander. *The Structure of Pure Reason*. Springer, February 2025. This is an English translation of [9] – done by Dines Bjørner in collaboration with the author.

[12] Due to copyright reasons no URL is given to this document's possible Internet location. A primer version, omitting certain chapters, is [4]

[13] This book is currently being translated into Chinese by Dr. Yang ShaoFa, IoS/CAS (Institute of Software, Chinese Academy of Sciences), Beijing and into Russian by Dr. Mikhail Chupilko and colleagues, ISP/RAS (Institute of Systems Programming, Russian Academy of Sciences), Moscow