An Essence of Domain Engineering

A Basis for Trustworthy Aeronautics and Space Software[⋆]

Dines Bjørner

DTU Compute, Technical University of Denmark, DK–2800 Kgs.Lyngby, Denmark
Fredsvej 11, DK–2840 Holte, Denmark
bjorner@gmail.com, www.imm.dtu.dk/~dibj

Abstract. Before *software* can be *designed* one must have a reasonable grasp of its requirements. Before *requirements* can be *prescribed* one must have a reasonable grasp of the domain in which the software is to serve. So we must *study, analyse* and *describe* the application *domain*. We shall argue that *domain science & engineering* is a necessary prerequisite for requirements engineering, and hence software design. We survey elements of domain science & engineering – and exemplify some elements of domain descriptions. We finally speculate on the relevance of domain engineering in the context of and aeronautics and space.

Keywords: Formal Methods · Philosophy · Software · Domain Engineering · Requirements Engineering.

1 Introduction

A monograph has been published: [11, *Domain Science and Engineering*]. We immodestly claim that the contents of that monograph "heralds" a *new*, an *initial*, phase of software development — a *new area of study* within the exact sciences.

An aim of the present paper is to propagate awareness of the aim & objectives of that book and hence of this new field, also, of computer science – as labeled by the book title.

Another side-aim is to also introduce the possibility of a *Philosophy of Informatics*[1]. This, we think, is *a first* for computer & computing science, to be "endowed" with a philosophy, as is mathematics [41], physics [14], life sciences

⋆ Invited paper for the *The 14th NASA Formal Methods Symposium*, https://nfm2022.-caltech.edu, May 24-27, 2022, Pasadena, California, USA

[1] We take *informatics* to be an amalgam of mostly mathematical nature: computer & computing science and mathematics. Another such amalgam is *IT* which we consider as mostly of technological nature: electronics, plasma and quantum physics, etc. *Informatics*, to us, is a *universe of intellectual quality:* meeting customers expectations, correct wrt. specifications, etc.. *IT* is then a *universe of material quantity:* smaller, bigger, faster, less costly, etc. The products of *informatics* [must] satisfy laws of mathematics, in particular of mathematical logic. The products of *IT* [must] satisfy the laws of physics.

[75], etc. Yes, we are aware of previous attempts[2] to include considerations of specific, detailed, technical issues of theoretical computer science as being of philosophical nature. But what we are suggesting, is, perhaps immodestly expressed, of a more foundational kind. In our treatment of a possible philosophy of informatics we shall *"dig deeper"*, as directed by [66–69].

The first four lines of the abstract expresses a dogma – the **Triptych**[3] dogma. In those lines we used the term 'reasonable'. By 'reasonable' we mean that we can rationally reason about the domain – as do physicists and mathematicians. To us that means that domain descriptions are expressed in some notation that allows logical reasoning. Here we shall use RSL, the Raise[4] Specification Language [29, 28]. To express the analysis and description calculi of this paper we shall use an informal extension of RSL, one whose description functions yield RSL texts, RSL$^+$Text.

This paper thus serves to propagate the dogma that software development proceeds from the study, analysis and informal and formal *domain descriptions*, via the "derivation" of *requirements prescriptions* from domain descriptions, to *software design*, "derived" from requirements prescriptions.

The paper presents a capsule view of the monograph. For the reasoning behind the various concepts and the technical details of the domain engineering method, its principles, techniques and tools, we refer to [11].

By a **method** we shall understand a set of principles and procedures for selecting and applying a number of techniques and tools for constructing an artifact. By a **formal method** we shall understand a method whose techniques and tools are given a mathematical understanding. By a **formal software development method** – in the context of the triptych dogma –we shall understand a formal method which is "built upon", i.e., utilizes, one or more formal specification languages, i.e., languages with formal syntax, formal semantics and proof systems, that are the used to describe, prescribe and design domain descriptions, requirements prescriptions and software – allowing formal tests [34], formal model checks [20] and formal proofs in order to verify these specifications and their transformations.

1.1 What is a Domain ?

By a *domain* we shall understand a *rationally describable*[5] area of a *discrete dynamics* segment of a *human assisted reality*, i.e., of the world, its *solid or fluid entities*: *natural* ["God-given"] and *artefactual* ["man-made"] parts, and its *living species entities: plants* and *animals* including, notably, *humans* [11, Sect. 4.2, Defn. 27]. In this paper we shall not cover the 'living species' aspects.

[2] https://en.wikipedia.org/wiki/Philosophy_of_computer_science

[3] Triptych: a picture (such as an altarpiece) or carving in three panels side by side, or something composed or presented in three parts or sections especially, like a trilogy

[4] Raise: Rigorous approach to industrial software engineering

[5] By 'rationally describable' we mean that the specification, in this case the description, must allow for formal, i.e., logical reasoning.

1.2 Structure of Paper

There are four main sections of this paper. Section 2 discusses the problem of what must, unavoidably, be in any domain description. It does so on the background of the quest of philosophers – since antiquity – for understanding the world around us. Sections 3–4 summarise, respectively exemplify, a domain analysis & description method. The two sections go hand-in-hand. They have, sequentially, 'near-identical' subsections and paragraphs. Where some aspects of the method may be omitted in Sect. 3, Sect. 4 may exemplify also those aspects. Section 5 'speculates' on further perspectives of domain science & engineering. Its potential for application in aeronautics and space!

2 Philosophy: What Must be in any Domain Description?

Philosophy, since the ancient Greeks, have pondered over the question: *which are the absolutely necessary conditions for describing any world?*, that is: *what, if anything, is of such necessity, that it could under no circumstances be otherwise?*, or: *which are the necessary characteristics of any possible world?* We take these three as one-and-the-same question.

Philosophers, from *Aristotle* (384–322 BC) to *Immanuel Kant* (1724–1804), and onwards, have contributed to understanding this set of questions. We shall draw upon the works of the Danish Philosopher *Kai Sørlander* (1944) [66–69]. We shall therefore base our search for techniques and tools with which to analyse & describe domains in Sørlander's findings. This, in effect means, that we suggest a philosophy-basis for domain analysis & description! Next we shall therefore first summarise two thousand five hundred years of trying to answer the question with which we opened this section.

2.1 The Search

We shall focus only on one aspect of the philosophies of the very many philosophers that are mentioned below — namely their thinking wrt. *ontology*[6] and *epistemology*[7]; for many of these philosophers – from Plato onwards – this is, but a mere fraction of their great thinking.

This section borrows heavily from [69]. That book is only published in Danish. So the next three pages, till Sect. 3, is a terse summary of the first 130 pages of [69].

The Ancient Greeks. The quest for understanding the world around us appears to have started in ancient Greece. *Thales of Miletus* [52] (624/623–548/545 BC) claimed that everything originates from `water`. *Anaximander* [21] (610–546 BC)

[6] Ontology is the study of of concepts such as *existence, being, becoming,* and *reality.*

[7] Epistemology is the study of properties, origin and limits for human knowledge.

counter-claimed that *'apeiron'* (the 'un-differentiated', 'the unlimited') was the origin. *Anaximenes* [51] (586–526 BC) counter-counter-claimed that *air* was the basis for everything. *Heraklit of Efesos* [1] (540-480 BC) suggested that *fire* was the basis and that everything in nature *was in never-ending ''battle''*. *Empedokles* [76] (490–430 BC) synthesized the above into the claim that there are four base elements: *fire, water, air* and *soil*. *Parminedes* [32] (515–470 BC) meant that everything that exists is *eternal and immutable*. *Demokrit* [1] (460–370 BC) argues that all is built from *atoms*. These were [some of] the *natural philosophers*, the *pre-Socrates* philosophers, the *ontologists*, of Ancient Greece.

The Sofists. Then came a period of so-called *sofists*. They maintained that we cannot reach understanding of the world through common sense. For a time they thus broke philosophical tradition. It was not their task to reach an understanding of that which exists. Such an understanding, they claimed, was an illusion; in that they seem to agree with today's modernism and post-modernism.

Socrates, Plato and Aristotle. *Socrates* (470–399 BC) [2] broke rank with this. For him it was a fundamental error to give up on the obligation of common, universal sense. Socrates, instead of reflecting on the general aspects of ontology, put the human in centrum. *Plato* [3] (427–347 BC) established a *Theory of Ideas* of "universal concepts" as of highest reality, that, however, seems to raise more questions than answering some. *Aristotle* [4] (384–322 BC) turned Plato's thinking upside-down: "concrete things" have primary existence and the universal concepts are abstractions. Aristotle made precise relations between the **modalities** of the *necessary,* the *real* and the *possible*, and suggested a list, [4, *Categories*], of ten **categories:** `substance`, `quantity`, `quality`, `relation`, `place`, `time`, `position`, `possession`, `acting` and `suffering`.

The "Middle Ages". Philosophical thinking – in the European sphere – from about 300 BC till about 1600 AC was dominated by religious thought – till shortly after the time of Martin Luther. From ontological arguments philosophy turned in the direction of epistemological arguments.

From Descartes to Hume. Then a number of philosophical schools succeeded one another. Sørlander shows that the philosophies of *Descartes* [24] (1596–1650), *Spinoza* [70] (1632–1677), *Leibniz* [46] (1646–1716), *Locke* [48] (1632–1704), *Berkeley* [7] (1685–1753) and *Hume* [39] (1711–1776) are individually inconsistent, and must thus be rejected.

Historicism. Sørlander also rejects the *'historicism'* philosophies, after *Immanuel Kant*, i.e., those of *Fichte* [43] (1762–1814), *Schelling* [6] (1775–1854) and *Hegel* [31] (1770–1831) as likewise individually inconsistent.

From Aristotle to Kant and onwards .... Sørlander builds on the thinking of *Aristotle* [4] (384–322) and *Immanuel Kant* [44] (1724–1804). In doing so, Sørlander takes up a thread, lost for two hundred years of *"radical meaninglessness, loss of religion, the disappearance of [proper] philosophy – lost in the "historicism" of the 19th century and the "modernism" of the 20th century, in postmodernism's rejection of universal values, the possibility of objective knowledge, or solid foundation for human existence." ... "In this post-modern age nothing seems to be absolutely valid, there is no sharp boundary between fiction and science, everything is dissolved into uncertainty and individual interpretation"* No, says Sørlander, and builds a Philosophy based on rational reasoning. The current author, obviously, subscribes to the above!

Philosophies of Sciences. The science breakthroughs, in the late 1800s and the early-to-mid 1900s, in mathematics, physics and biology, brought with it, independent of the 'historicism' of philosophy, philosophical investigations of these sciences.

Peano (1858–1932) [45] showed that some of **mathematics** could be understood axiomatically, i.e., logically. *Frege* (1848–1925) [26] contributed significantly to attempts to build an axiomatic basis for all of mathematics. On the basis of similar axiom systems *non-Euclidean Geometries* were then put forward[8]. *Principia Mathematica* [72, *Whitehead & Russell*] "grandiosely" attempted to axiomatise all of mathematics. *Gödel*'s (1906–1978) [30] *first incompleteness theorem* states that in any formal system $\mathbb{F}$ within which a certain amount of arithmetic can be carried out, there are statements of the language of $\mathbb{F}$ which can neither be proved nor disproved in $\mathbb{F}$. According to the second incompleteness theorem, such a formal system cannot prove that the system itself is consistent (assuming it is indeed consistent). These results have had a great impact on the philosophy of mathematics and logic.

Within **physics**, *Maxwell* (1831–1879) [49] *Planck* (1858–1947) [54] originated *quantum mechanics*. *Einstein*[9] (1879–1955), in 1905–1916 changed the study of physics with his *special* and *general theories of relativity*. *Bohr*[10] (1885–1962) [25] contributed with his understanding of the structure of atoms and with *quantum theory*. *Heisenberg* (1901–1976) [33] contributed further to quantum theory and is known for the uncertainty principle.

Darwin (1809–1882) [22, Origin of Species], *Wallace* (1823–1913) [71], and *Mendel* (1822–1884) [50] – as did Planck, Einstein, Bohr, Heisenberg, et al. for physics – founded modern **life sciences.**

These advances in mathematics and the natural sciences spurred some philosophers on to renewed studies – *"as from Kant!"*

The 20th Century. The *phenomenology* of *Husserl* (1859–1938) [40], *is the study of structures of consciousness as experienced from the first-person point of view*

[8] https://en.wikipedia.org/wiki/Non-Euclidean_geometry
[9] https://en.wikipedia.org/wiki/Religious_and_philosophical_views_of_Albert_-Einstein#Philosophical_beliefs
[10] https://plato.stanford.edu/entries/qm-copenhagen/

[Wikipedia]. Our consciousness, claims Husserl, is characterised by *intentionality:* an elementary directedness. Husserl's phenomenology appears to be inconsistent in the way it requires a study of our consciousness from "within", for example in the a-priory requirements that these concepts are introduced, not as a result of the study, but "beforehand".

It appeared then that philosophical studies along the lines *"what must inevitably be in any description of any domain"* additionally required consideration of our use of language. *Wittgenstein* (1889–1951) [73, 74] and the *Logical Atomism* [53] of *Russell* (1872–1970) [60–63], made attempts in this direction, but failed. Wittgenstein realised that in his [74, *"Philosophisces Untersuchungen"*]. Logical atomism failed in not finding examples of propositions if they have to be logically independent of one another.

Logical Positivism, "coming out of" Vienna in the 1920s–1930s, rejected Russell's logical atomism and concentrated on the meaning of a sentence as being [the conditions for] its truth-value: one must be able to describe the circumstances under which the sentence can be verified. To them, in the early days, meaningful propositions, say, in any of the sciences, must have a common linguistic base. Eventually those theses also failed: Neither the verification-criteria, of, for example *Carnap* (1891–1970) [15–18], could be verified, nor could the falsification-criteria of *Popper* (1902–1994) [55, 57, 58] be falsified.

2.2 Sørlander's Findings

Three Cornerstones. We can claim that Sørlander bases his philosophical analyses on three "cornerstones": (A) an analysis and a conclusion of *"what it means to be rational"*; (B) an analysis and a conclusion of what it means to speak abut *"the meaning of a word"*; and (C) an analysis and a conclusion of the base point from which to start the philosophical inquiry into *"what must inevitably be in any domain description"*. We shall now review these three bases.

A: Rational Thinking. The following is adapted from [67, Chapter II, Sects. 4–5 *Common Sense and Motivation*]. Humans are physical entities. Thus we are characterisable by the causal conditions for moving around with purpose. To do so requires three conditions: We can *sense* our immediate situation. We have *feelings* that may result in incentives (encouragements). We have motoric apparatus that satisfy physical laws. These were the causal conditions for purposeful movements. Further: We possess languages by means of which to express propositions as to what we sense, our feelings and actions. We express propositions which reflect that *we know*, i.e., *have knowledge*, Finally we have *memory* from which we build *experience*. The above factors, after some further analysis, leads us to conclude that humans are *rational beings*.

B: The Implicit Meaning-Theory. The following is adapted from [68, Chapter III, especially Sect. 9 *The Meaning of a Word*, Pages 121–122]. On the basis of some simple considerations of what it means to express oneself by means of language, i.e., linguistically, Sørlander reaches the *interdependence* criterion. In saying, or

writing, something, a choice is made. The chosen statement may be inconsistent with something else that one could have chosen to state. That means, that possible statements stand in consistence relations. What determines such relations? We can, firstly, say that these relations are determined by the meaning, of the designations used in the statements. Secondly we can say that meaning of the designations used in several statements which (thus must) stand in mutual consistence-relations. It is thus that we arrive at the necessary condition, *interdependence* criterion, also referred to as the *implicit meaning theory*, that *there is a mutual dependence between the meaning of designations and the consistence relations between statements*.

For computer scientists, this interdependence criterion is quite familiar. When defining an *abstract data type* — that is, its values and operations, as is, for example typical in algebraic semantics [64] — one states a number of propositions. They constrain values and operations, and, together, express their meaning.

C: The Possibility of Truth. The following is adapted from [69, Part III, Chapter 2 *"Basis & Method for the Philosophy"*]. Where Kant built on *human self-awareness*, Sørlander builds on the *possibility of truth*. One cannot deny that a proposition may be false. And one cannot accept that a proposition is both true and false. Hence the possibility of truth.

Building a Foundation.

Logic, Relations, Transcendental Deduction, Space and Time. On the basis of *the principle of contradiction* and the *implicit meaning theory*. Kai Sørlander then motivates *the logical connectives* and, from these, *the associative, symmetry and transitive relations*, and, based on these, by *transcendental deduction*, reasons that *space* and *time* follows, not as, with Immanuel Kant, empirical facts, but as logical necessities.

Multiple, Uniquely Identifiable Entities and States. Again, in a rational manner, Sørlander, motivates that there must be an indefinite number of entities, that these are uniquely identifiable, and that they endure in possibly changing states.

Newton's Laws. Again, in a rational manner, Sørlander, motivates *movement* and *causality*, and, from these, again by *transcendental deductions*, *Newton's Laws*.

2.3 The Basis

The above, i.e., the rational deductions of what must be in any domain description, is then the foundation on which [11] and the present paper base their domain analysis & description approach.

3 Elements of Domain Science & Engineering

We embark on introducing a number of *domain analysis predicates*. These are not mathematical functions. They are informal in the sense of being applied by human *domain analysers cum describers*. They can not be formalised. That would require that we have a formal model of "the world" ! Our domain analysis & description endeavour seeks such models ! So the reader must bear with me: The delineations (cum definitions, characterisations) of the domain concepts that now follow must unavoidably be informal, yet sufficiently precise. Most are drawn from *The Shorter Oxford Dictionary of the English Language* [47, 2 vols., 1987].

3.1 Phenomena, Entities, Endurants and Perdurants

A *phenomenon*, ϕ, is an *entity*, `is_entity(`ϕ`)`, if it can be *observe*d, i.e., be seen or touched by humans, *or* that can be *conceive*d as an *abstraction* of an entity; alternatively, a phenomenon is an entity *if it exists, it is "being", it is that which makes a "thing" what it is: essence, essential nature* [47, Vol. I, pg. 665]. If a phenomenon cannot be so described it is not an entity.

There are an indefinite number of entities in any domain. This follows from philosophic-analytic reasoning outlined by the philosopher Kai Sørlander [66–69]. We refer to [11, Sect. 2.2.3] for a summary.

By an *endurant*, `is_endurant(`e`)`, we shall understand an entity, e, that can be observed, or conceived and described, as a "complete thing" at no matter which given snapshot of time; alternatively an entity is endurant if it is capable of *enduring*, that is *persist*, *"hold out"* [47, Vol. I, pg. 656]. Were we to "freeze" time we would still be able to observe the entire endurant.

By a *perdurant*, `is_perdurant(`e`)`, we shall understand an entity, e, for which only a fragment exists if we look at or touch them at any given snapshot in time. Were we to freeze time we would only see or touch a fragment of the perdurant [47, Vol. II, pg. 1552].

External qualities of endurants of a manifest domain are, in a simplifying sense, those we, for example with our eyes blinded, can touch, hence manifestly "observe", and hence speak about abstractly.

Internal qualities of endurants of a manifest domain are those we, with our eyes open and with instruments, can measure.

3.2 Endurants

Figure 1 presents a graphic structure of the domain concepts such as we have and shall unveil them.

External Qualities. Our treatment of endurants "follow" the upper ontology of Fig. 1 in a left-to-right, depth-first traversal of the endurant "tree" (of Fig. 1).

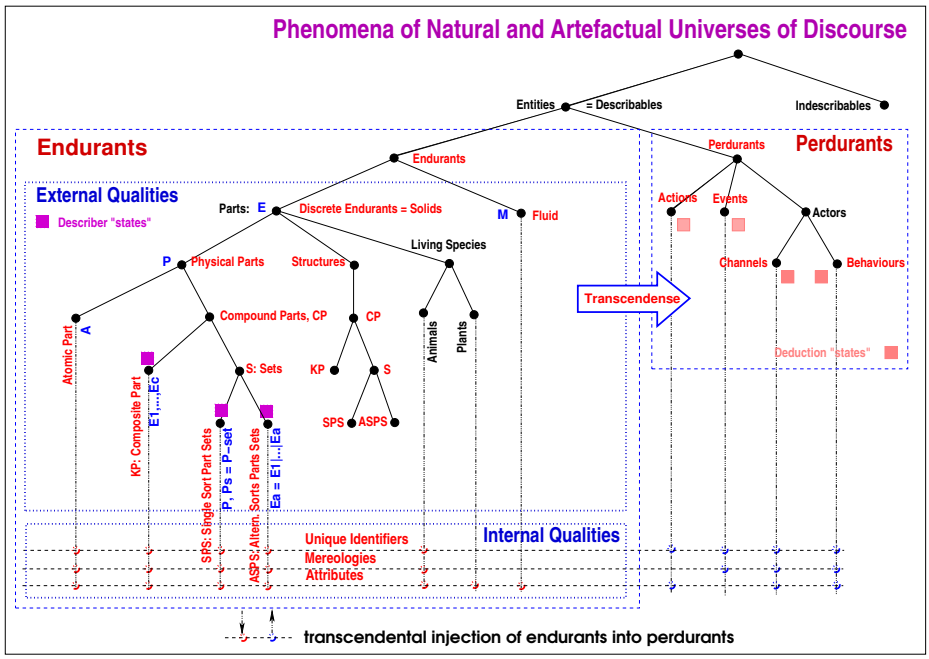

Fig. 1. A Domain Description Ontology

Analysis Predicates. *Endurants*, e `[is_endurant(`e`)]`, are either *solid* `[is_solid(`e`)]`; or *fluid* `[is_fluid(`e`)]` (such as liquids, gases and plasmas). Solid endurants appears to be the "work-horse" of the domains we shall be concerned with. *Fluid*s are presently further un-analysed. A *solid*, e, is either a *part* `[is_part(`e`)]`; or a *structure* `[is_structure(`e`)]`; or a *living species* `[is_living_species(`e`)]`. A *part*, p, is either an *atomic part* `[is_atomic_part(`p`)]`; or a *compound part* `[is_compound_part(`p`)]`. An *atomic part*, by definition, has no proper sub-parts. It is the domain analyser cum describer who decides which parts are atomic and which not. Atomic parts are further characterised by their internal qualities. A *compound part* is either a *composite part* `[is_composite(`p`)]`; or a *part set* `[is_part_set(`p`)]`, A *part set* is either a *single sort part set* of parts of the same sort `[is_single_sort_set(`p`)]`; or an *alternative sort part set* of parts of two or more distinct sorts `[is_altermative_sort_set(`p`)]` – with two or more parts possibly being of the same sort.

A *composite* part consists of two or more parts (and could be modeled as a *Cartesian* of these). A *structure* is like a compound part but we omit recording its internal qualities[11]. A *living species* is either an *animal* `[is_animal(`e`)]`; or a *plant* `[is_plant(`e`)]`. An *animal* is either a *human* `[is_human(`e`)]`; or other.

[11] We could omit the concept of structure altogether and just allow compounds that do not have internal qualities.

Observers. Given a compound part we can observe its sub-parts and their sorts. We formulate these observers in RSL$^+$Text.

• • •

We remind the reader that the analysis and description processes are necessarily informal. That is, that it is the decision of the domain analyser cum describer as to whether an entity is an endurant or other, a part or other, etc. Next, in outlining, ever so briefly, the observer (cum describer) "functions", the describer must, repeatedly, decide that endurants are of definite sorts, and must, likewise repeatedly, choose names for endurant sorts. [11, Sect. 4.14] discusses that process in some detail. In reality, to determine, distinctness and names of sorts require a depth-first analysis, that is, one that analyses the internal qualities of the sort under investigation, then the external followed by internal qualities of possible sub-parts, et cetera, till atomic parts or fluids have been analysed, etc. In this section we first analyse external qualities. Analysis of internal qualities follow subsequently.

• • •

The next three observer functions reflect analyses pre-requisite to the subsequent description functions. In the formulas below we introduce two notions: The *name* of a type, say type E, as ηE, and the RSL text, of a type name, "E". ηE is an identifier whose value is "E".

Observe Single Sort Part Sets.

Observing a part, $p : P$, which is a set of endurants of the same sort, yields a pair of a set of endurants and the name of the endurant type.

```
value
  obs_single_sort_set: P → E-set × ηE
```

where E is to be further analysed and described.

Observe Alternative Sorts Part Sets.

Observing a part, $p : P$, which is a set of endurants of the possibly different sorts, yields a Cartesian of representative pairs[12] of endurants and names of their type.

```
value
  obs_alternative_sort_set: P → (E1 ×ηE1)×(E2 ×ηE2)× ... ×(Em ×ηEm)
```

where each Ei is to be further analysed and described.

Observe Composite Part.

Observing a part, $p : P$, which is a composite of endurants of [it is assumed] different sorts, yields a pair Cartesians of endurants, respectively their type.

```
value
  obs_composite: P → (E_1×E_2×...×E_m) × (ηE_1×ηE_2×...×ηE_m)
```

where each E_i is to be further analysed and described.

[12] By 'representative Cartesian of pairs' we mean that there is a pair of any part (of the set) and its type for every possible part type in the Cartesian.

Description Functions. There are three compound-part description functions. These are summarised in the RSL$^+$Text form next.

We advocate first narrating all formal texts. The literal **type** prefix type and sort definitions. The literal **value** prefix predicate and function signatures and definitions. Proof obligations are required where sorts are expressed in terms of concrete types that may define something meaningless if not properly constrained.

Caveat: We remind the reader that the above *description functions*, really, are not mathematical functions: They are, in a sense, procedural guide-lines to be followed by *domain analysers cum describers:* they have to decide on which kind of parts they are dealing with, of which, already "discovered" or new sorts, hence sort names to ascribe these, etc.

The External Qualities Frames. The three frames next contain part descriptors for single sort sets, alternative sort sets, and composites.

describe_single_sort_set(p) **as**

```
let (_,ηE) = obs_single_sort_set(p) in
"Narration:
    ... on sorts ...
    ... on sort observers ...
    ... on axioms/proof obligations ...
Formalisation:
    type
        E
        Ps = P-set
    value
        obs_Ps: E → Ps  "
end
pre: is_single_sort_set(p)
```

describe_alternative_sorts_set(p) **as**

```
let ((_,ηE_1),...,(_,ηE_n))
      = obs_alternative_sorts_set(p) in
"Narration:
    ... on alternative sorts ...
    ... on sort observers ...
    ... on axioms/proof obligations ...
Formalisation:
    type
        Ea = E_1 | ... | E_n
        E_1 :: E1, ..., E_n :: En
        E1 == ..., ..., En == ...
    value
        obs_E_i: E → E_i [ i=1,...,n ]
    proof obligation
        [ disjointness of alt. sorts ]  "
end
pre: is_alternative_sorts_set(p)
```

describe_composite(p) **as**

```
let (_,({ηE1,...,ηEm})) =
      = obs_composite(p) in
"Narration:
    ... on sorts ...
    ... on sort observers ...
    ... on axiom/proof obligations ...
Formalisation:
    type
        E1, ..., Em
    value
        obs_Ei: E → Ei [ i:{1..m} ]
    proof obligation
        [ disjoint endurant sorts ]  "
end
pre: is_composite(p)
```

Initial Endurant State. An *endurant state* is any set of domain endurants.

Taxonomy. The taxonomy of a domain is given by the set of endurants sorts and their observers. A taxonomy can be given a graphic rendition such as shown in Fig. 2 on page 20.

Internal Qualities. *Internal qualities* of endurants of a manifest domain are, in a simplifying sense, those which we may not be able to see or "feel" when "touching" an endurant, but they can, as we now 'mandate' them, be reasoned

about, as for *unique identifiers* and *mereologies*, or be measured by some *physical/chemical* means, or be "spoken of" by *intentional deduction*, and be reasoned about, as we do when we *attribute* properties to endurants. We refer to [11, *Sects. 2.2.3–4, 3.8, and 5.2–5.3*] for a fuller discussion of the concepts and unique identification and mereology.

Unique Identifiers. With each part sort P we associate a further undefined unique identifier sort Π and a similarly further undefined unique identifier observer **uid**_P such that for all distinct parts $p, p', ..., p''$ of sort P, **uid**_P(p), **uid**_P(p'), ..., **uid**_P(p''), yield distinct unique identifiers $(\pi, \pi', ..., \pi'')$.

We refer to the leftmost of the three internal qualities frames on Page 13.

Mereology. "Mereology (from the Greek $\mu\epsilon\rho o\varsigma$ 'part') is a theory of part-hood *relations:* of the relations of part to whole and the relations of part to part within a whole"[13].

The mereology relations are here expressed in terms of the unique part identifiers. Let p:P (p of sort P) be a part with unique identifiers π. Let $\{p_1 : P_1, p_2 : P_2, ..., p_m : P_m\}$ be the set of parts (or respective sorts) to which p is [mereologically] related. We can express this by stating that mereo_P(p) = $\{\pi_1 : \Pi_1, \pi_2 : \Pi_2, ..., \pi_m : \Pi_m\}$, or **value** mereo_P: P→UI-**set** – i.e., as a set of unique identifiers. mereo_P is the mereology observer.

We shall deploy mereology practically. That is, we are not studying mereology. We are using the ideas of mereology for experimental research and engineering purposes.

For natural endurants, a typical relation is that of the topological *"next-to"*. For artefactual endurants typical relations, in addition to topological mereologies, make explicit how the designers of these artefacts *intended* their logical, not necessarily geographical relationship, to be: *"next-to"*, *"to-be-part-of"*, *"as-an-element-of-a-set"*, et cetera.

We refer to the middle of the three internal qualities frames on Page 13.

Attributes. Whereas unique identification and mereology are both of abstract, existential, logic nature, attributes are of concrete nature: physical, biological or historical nature. Attributes have values and attribute values are of types. *Two or more endurants that all have sets of attribute values of the same type, as well as the same unique identifier type and mereology types, are of the same sort. This is the endurant sort-determining mantra.*

From any part, p:P, we can thus identify a set of attribute type names, $\{A_{p_1}, A_{p_2}, ...A_{p_p}\}$, informally:

– *attrs*_P(a) **as** $\{\eta A_{p_1}, \eta A_{p_2}, ..., \eta A_{p_p}\}$.

Given a p:P, attr_A obtains the value of attribute A. The attr_A_{p_i}s are attribute observers of p_i:P_i.

[13] Achille Varzi: Mereology, http://plato.stanford.edu/entries/mereology/ 2009 and [19].

We refer to the rightmost of the three internal qualities frames on Page 13. Michael A. Jackson [42] has suggested a hierarchy of attribute categories.

- *Static attributes:* values do not change.
- *Dynamic attributes:* values can change.
  Within the dynamic attribute category there are sub-categories.
  - *Inert attributes:* values are not determined by the endurant, but by "an outside" (e.g., other endurants).
  - Or *reactive attributes:* values which, if they change, change in response to external stimuli.
  - Or *active attributes:* values which change of the "own volition" of the part.
    We can define sub-categories of dynamic attributes.
    - *Autonomous attributes:* values which change only on the "own volition" of the part.
    - *Biddable attributes:* values, values that may be prescribed[14], but may fail to attain the prescribed value.
    - And *programmable attributes:* values which are prescribed.

For our purposes we "reduce" these six categories to three, CAT = STA|MON|PRO:

- *static* [STA], (static values),
- *monitorable* [MON] (dynamic, except the programmable values), and the
- *programmable* (values) [PRO].

The Internal Qualities Frames. The three frames next contain part descriptors for unique identifiers, mereologies, and attributes.

unique_identifier_observer(p) **as**

"**Narration:**
  on unique identifier sort UI ...
  on unique identifier observer ...
  on uniqueness of identifiers ...
Formalisation:
type
  UI
value
  uid_P: P → UI
axiom
  [disjoint UIs wrt. all sorts] "

mereology_observer(p) **as**

"**Narration:**
on mereology type ...
on mereology observer ...
on mereology type constraints ...
Formalisation:
type
  MT = $\mathcal{M}$(UI$_i$,...,UI$_k$)
value
  mereo_P: P → MT
axiom [Well−formed Mereology]
  $\mathcal{A}$(MT): well−formed "

describe_attributes(p) **as**

let {ηA$_1$,...,ηA$_m$} = *attrs_*P(p) **in**
"**Narration:**
  on attribute sorts ...
  on attribute sort observers ...
  attribute sort proof obligations ...
Formalisation:
type
  A$_1$, ..., A$_m$
value
  attr_A$_1$: P→A$_1$,
  attr_A$_1$: P→A$_2$,
  ...,
  attr_A$_1$: P→A$_m$
proof obligation [Disjointness]
  let P be any part sort **in**
  let a:(A$_1$|...|A$_m$) **in**
  is_A$_i$(a)≠is_A$_j$(a) [i≠j, i,j:[1..m]]
  end end "
end

[14] – by the transcendent part behaviour

Intentional Pull. The concept of *intentional "pull"* is a concept which "parallels", we claim, the *gravitational pull* concept of physics.

For artefacts one can claim that certain parts p:P are created in order to "serve" other parts q:Q, and vice versa: *roads serve to convey transport*, and *automobiles serve to transport goods*.

Historical events *time-stamp* record interactions between such parts p and q. So a historical attribute of p records its interaction with q, and a historical attribute of q records its interaction with p, and *"one cannot have one without the other"*, and this is what we mean by *intentional "pull"*!

Since we can talk about such events we can also model them as attributes. So introducing historical attributes for a sort P usually entails also introducing historical attributes for another sort Q, et cetera. And this consequentially implies that the domain analyser cum describer must express a necessary *intentional "pull" axiom* that expresses that *"one cannot have one without the other"*.

A classical example of intentional pull is found in *double bookkeeping* which states that every financial transaction has equal and opposite effects in at least two different accounts. It is used to satisfy the accounting equation: $Assets = Liabilities + Equity$.

3.3 Transcendental Deduction

"A *transcendental argument* is an argument which elucidates the conditions for the possibility of some fundamental phenomenon, whose existence is unchallenged or uncontroversial in the philosophical context in which the argument is propounded" [5, Anthony Brueckner, page 808]. "Such an argument proceeds deductively, from a premise of asserting the existence of some basic phenomenon (such as a meaningful discourse, conceptualisation of objective states of affairs, or the practice of making promises), to a conclusion asserting the existence of some interesting, substantive enabling conditions for that phenomenon" [5, Anthony Brueckner, page 808].

An **example** of a transcendental deduction is that of "morphing", for example, *automobile endurants* into *automobile perdurants*. That is: There is the automobile as, for example, shown at the dealer. It represents a *part*, an *endurant*. And there is the automobile "speeding" down the road. It represents a *behaviour*, a *perdurant*. The automobile as listed in the manufacturer's and car dealer's catalogues represents an *attribute* of manufacturers and dealers.

3.4 Perdurants

The emphasis is now on the **transcendental** deduction of **parts** into **behaviours**.

To explain what we mean by behaviours we first introduce *actions* and *events*. *Channels* will be introduced as a consequence of *interacting*, that is, *communicating* behaviours.

This section is necessarily a mere capsule view of Chapter 7 of [11]. Section 4.2, of the main example of this paper, should rectify some lacunae.

Actions, Events and Behaviours.

Actions. By an *action* we shall understand something that occurs in time, lasting, however, no time, or, at least, we ignore time – considering actions as indivisible, taking place as the result of a "willed" [other] action, and usually changing the state ξ:Ξ[15].

The action may, or may not be based on some argument value.

value action: [VAL] $\rightarrow \Xi \xrightarrow{\sim} \Xi$

Events. By an *event* we shall understand something that occurs in time, lasting, however, no time, taking place spontaneously, not as the result of a "willed" action, but possibly as the result of another event, and usually changing the state ξ:Ξ.

The event is usually not based on any argument value. The literal **Unit** can here be understood as a no argument value.

value event: **Unit** $\rightarrow \Xi \xrightarrow{\sim} \Xi$

Behaviours. By a *behaviour* we shall then understand a set of sequences of actions, events and [other, sub-] behaviours, some of which relate to, i.e., *interact* with one another. Behaviours are uniquely identified, subject to the part mereology, and otherwise based on *static* (*constant*) attribute argument values, *dynamic monitorable* (*variable*) attribute argument values, *dynamic programmable* (*variable*) attribute argument values, and *channels* (for their interaction).

Behaviour Deduction, I: Signature

```
value behaviour: Uid × Mereo × Static_VAL* × Mon_Attr_Name*
                                → Prgr_VAL*  → in|out|in out ch... Unit
```

The literal **Unit** will here be understood as defining a never-ending behaviour. The signature, with **Unit**, expresses that if the process terminates no value is returned.

Channels. Interactions – between behaviours – are, as we model them, in `RSL` – as inspired by `CSP` [35, 36, 59, 65, 37], expressed in terms of `CSP-like` *channel* (ch) input/outputs: *ch[index] ?*, respectively *ch[index] ! value*, where *values* [based on internal qualities] are *communicated* over indexed channels.

A domain defines a number of *mereologies*, one for each part (of the *state*). These mereologies determine the *channels* to be *declared*. Given that any interesting, i.e., to us relevant, domain always consists of an indefinite, larger than 1,

[15] We shall forego explaining the state concept Ξ.

number of parts, the *common channel* for all behaviours is an index-able *channel array*[16]*:*

Channel Deduction

```
channel {ch[ {i,j} ]|i,j:UI•{i,j}⊆ [ mereologies of the domain ]}:M
```

where M is the type of the values communicated.

Part Behaviours. Parts exist in a context of several parts. (The taxonomy, for example graphically represented, as in Fig. 2 on page 20, reflects these parts.) Part behaviours can therefore be expected to interact, i.e., to synchronise and communicate. A part behaviour can, consequently, be expected to alternate between either (a) doing an internal non-deterministic choice (⨅) of 0, 1 or more "own work" behaviours, or (b) external non-deterministic choice (▯) offering [to accept] values from an alternative of 0, 1 or more other part behaviours. We can, schematically, summarise (a-b) as follows:

Behaviour Deduction, II: Part Behaviour Definition Structure

```
value
  part_behav(...)(...) ≡
    (a) ⨅ { own_behav_i(...)(...) | i ∈ {1..p} }
        ⨆
    (b) ▯ { ext_behav_j(...)(...) | j ∈ {1..q} }
    where: p+q > 0
```

The ⨅ and ▯ operators are the usual CSP operators on behaviours. The ⨆ operator is like an "or" operator on behaviours. The ⨅, ⨆ and ▯ operators are commutative. We shall refer to either of the alternatives of the part_behav definition body as a part_alternative.

From Internal Qualities to Behaviour Arguments. By arguments of transcendental nature we shall assign unique part identifiers as static arguments of behaviours, part mereologies as determining channel communication, and part attributes as either static or dynamic arguments of behaviours.

Behaviour Deduction, III: Signature, Part p:P

```
value
  behaviour_P: PI × mereo_P × Stat_Attr_Vals_P × Mon_Attr_Names
               → Prgr_Attr_Vals_P →
               → in|out|in out {ch[ {i,j} ] [ i,j ∈ mereology of P ]} Unit
```

[16] RSL does not have channel arrays. So this is a deviation from RSL.

Mon_Attr_Names makes use of *attrs*_P.

Part Alternative Behaviours. We shall express behaviours in terms of usually never-ending functions, behaviour !.[17] That is:

Behaviour Deduction, IV: Alternative Part Definition, Part p:P

```
value
    alt_behav(uid_P(p),mereo_P(p),Stat_Attr_Vals_P(p),Mon_Attr_Names(p))
                  (Prgr_Attr_Vals_P(p)) ≡
        let ui=uid_P(p), me=mereo_P(p), sta=Stat_Attr_Vals_P(p),
            mnl=Mon_Name_list(p), prgr=Prgr_Attr_Vals_P(p) in
        let prgr′ = alt_behav_body(ui,me,sta,mnl)(prgr) in
        part_behav(ui,mereo,sta,mnl)(prgr′) end end
```

Behaviour Clauses: Expressions and Statements. Further: alt_behav_body is a sequence of one or more action, event and sub-behaviour clauses – usually ending with an expression:

```
value
  behaviour_body(uid,mereo,sta_var)(prgr_var) ≡ clause_1 ; clause_2 ; ... ; clause_m
```

Clauses are either

- s, simple `statements`, or
- ch[...] ! expression, `output statement`, or
- **let** pattern[18] = expression **in** ... **end**, `value decompositions`, or
- e, `expressions`[19], or
- clause_a ⌈⌉ clause_b, internal non-deterministic clauses, or
- clause_a [] clause_b, external non-deterministic clauses, or
- clause_a ⌊⌋ clause_b, either/or non-deterministic clauses, or
- clause_a ‖ clause_b, parallel clauses, or
- **skip**, skip clause, or
- **stop**, abort function invocation.

Values of monitorable attributes, of name ηA[20], of parts p:P, are expressed as attr_val(uid)(σ) where attr_val is defined as:

```
value
  attr_val: PI → ηA → Σ → VAL
  attr_val(pi)(ηA)(σ) ≡ attr_A(retr_P(pi)(σ))
```

[17] *Parts* – being the bases for behaviours – persist, endure.
[18] where pattern – typically is a "grouping expression" over [free] identifiers
[19] ch[{ui,uj}] ? is an expression
[20] The type of attribute A names (a single element type) is ηA, and the value is "A". The type of all attribute names is $\eta\mathbb{A}$

where σ is the endurants state:

```
type
  Σ = (P|Q|...|R)-set
value
  retr_P: PI → Σ
  retr_P(pi)(σ) ≡ let p:P • p ∈ σ • uid_P(pi) in p end
```

Initial System. Given a[n endurant] state, cf. Page 11, one can then define [a corresponding perdurant] behaviour, namely the parallel (‖) composition of an invocation of all the corresponding behaviours. This is exemplified as from Item 69 on page 29.

3.5 The Domain Analysis & Description Process

1. There is the RSL$^+$Text to be developed.
2. There is the $\mathcal{D}$omain.
3. The analyse_and_describe_domain process applies to a $\mathcal{D}$omain and yields, line 12 an RSL$^+$Text. That process proceeds "sequentially":
4. first external qualities, then
5. unique identifiers,
6. mereologies,
7. attributes,
8. channels
9. behaviour signatures,
10. behaviour definitions, and
11. initial system – yielding
12. a complete RSL$^+$Text[21].

```
type
1. RSL+Text
2. D
value
3. analyse_and_describe_domain: D → > RSL+Text
3. analyse_and_describe_domain(d) ≡
4.   let es = analyse_and_describe_external_qualities(d) in
5.   let is = analyse_and_describe_unique_identifiers(es)(d) in
6.   let ms = analyse_and_describe_mereologies(es⊎is)(d) in
7.   let as = analyse_and_describe_attributes(es⊎is⊎ms)(d) in
8.   let cs = analyse_and_describe_channels(es⊎is⊎ms⊎as)(d) in
9.   let ss = analyse_and_describe_signatures(es⊎is⊎ms⊎as⊎cs)(d) in
10.  let bs = analyse_and_describe_behaviours(es⊎is⊎ms⊎as⊎cs⊎ss)(d) in
11.  let si = analyse_and_describe_initial_system(es⊎is⊎ms⊎as⊎cs⊎ss⊎bs)(d)(s)
12.  in es ⊎ is ⊎ ms ⊎ as ⊎ cs ⊎ ss ⊎ bs ⊎ si end end end end end end end end
```

[21] The ⊎ operator merges RSL$^+$Texts

4 An Example Domain Description

Initial Remark: *We shall illustrate core elements of a domain description of a road transport system. In doing so we really do not rely on the reader having already an idea as to what the terms of this road transport system mean – as we "slowly" unfold it. But at any stage, before the final, the informal meaning that You, the reader may ascribe to these terms, is not what the formulas express! At any stage, up to the point of the formal specification that we are unfolding, this specification denotes a space of meanings according to the RSL semantics [27]. Initially that space is very large. As we proceed the further formulas narrow down, restrict, the space. When, at the end, we think we have specified all that we need specify, the formulas define "exactly" what we mean by a road transport system. We shall continue this remark at the very end of this section, i.e., just before Sect. 5.*

• • •

The sectioning/paragraph structure of this section follows that of Sect. 3.

4.1 Endurants

External Qualities.

13. We start by identifying and naming the universe of discourse, here a road transport system.
14. In a road transport system we can observe a structure of a composite in which we observe an aggregate of a road net and an aggregate of automobiles.
15. Road nets are here seen as structures of composites of aggregates of road links[22] and road hubs[23].
16. Link and Hub aggregates are set structures of Links, respectively Hubs.
17. Links and Hubs are considered atomic.
18. Automobile aggregates are set structures of automobiles.
19. Automobiles are considered atomic.

```
type                                value
13. RTS                             14. obs_RN: RTS → RN
14. RN, AA                          14. obs_AA: RTS → AA
15. LS, HS                          15. obs_LS: RN → LS
16. Ls = L-set, Hs = H-set          15. obs_HS: RN → HS
17. L, H                            16. obs_Ls: LS → Ls
18. As = A-set                      16. obs_Hs: HS → Hs
19. A                               18. obs_As: AA → As
```

State.

20. The state, σ,
21. of a road transport system rts consists of

[22] A link is a street segment delineated by street intersections.
[23] A hub is a street intersection of one or more links.

(a) the road net aggregate,
(b) the automobile aggregate,
(c) the links,
(d) the hubs,
(e) the automobiles.

22. For later use we also define the union of all links and hubs.

value
21. rts:RTS
20. σ:$\Sigma = \{rn\} \cup \{aa\} \cup ls \cup hs \cup as$
21a. $rn = \mathsf{obs\_RN}(rts)$
21b. $aa = \mathsf{obs\_AA}(rts)$
21c. $ls = \mathsf{obs\_Ls}(\mathsf{obs\_LS}(rn))$
21d. $hs = \mathsf{obs\_Hs}(\mathsf{obs\_HS}(rn))$
21e. $as = \mathsf{obs\_As}(aa)$
22. us:(L|H)-**set** $= ls \cup hs$

Taxonomy. Figure 2 presents a graphic rendition of the taxonomy of road transport systems.

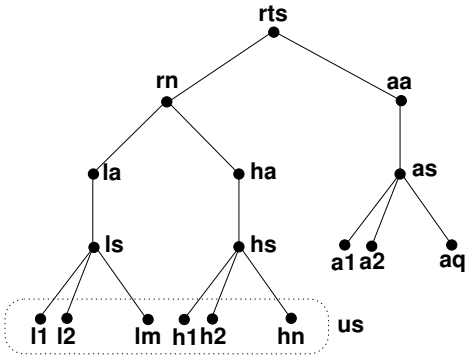

Fig. 2. Road Transport System Taxonomy

Internal Qualities.

Unique Identifiers. Road traffic systems, aggregates of links and hubs, and sets of links, hubs and automobiles are endurant ***structures***, hence have no internal qualities[24].

23. Road nets have unique identification,
24. automobile aggregates likewise,
25. links, hubs and automobiles also!

[24] – so we have decided!

type
23. RNI
24. AAI
25. LI, HU, AI

value
23. uid_RN: RN → RNI
24. uid_AA: AA → AAI
25. uid_L: L→LI
25. uid_H:H→HI
25. uid_A:A→AI

Uniqueness of Parts.

26. All parts (of the state σ) have unique identification. This means that the number of state components equal the number of [their] unique identifiers.

value
26. rni = uid_RN(rn), aai=uid_AA(aa),
26. lis = {uid_L(l)|l:L•l∈ ls},
26. his = {uid_H(h)|h:H•h∈ hs},
26. ais = {uid_A(a)|a:A•a∈ as},
26. σuis = $\{rni\}\cup\{aai\}\cup lis\cup his\cup ais$

axiom
26. **card** σ = **card** σuis

Retrieving Endurants.

27. Given any unique identifier, ui, in σuis, the "corresponding" endurant, e, can be retrieved from σ.

value
27. retr_E: UI → Σ → E
27. retr_E(ui)(σ) ≡ **let** e:E • e ∈ σ ∧ uid_E(e) = ui **in** e **end**

Mereology.

28. The mereology of a road net aggregate is a pair of the unique identifier *of the automobile aggregate of the road transport system of which the road net is an aggregate*, and a pair of sets of the unique identifiers *of the links and hubs of the road transport system of which the road net is an aggregate.*
29. The mereology of an automobile aggregate is a pair of the unique identifiers *of the road net aggregate of the road transport system of which the automobile aggregate net is a part*, and a set of unique identifiers of automobiles *of the automobile aggregate of the road transport system of which the automobile aggregate is a part.*
30. The mereology of a link is a pair of a two element set of hub identifiers and a set of identifiers of the automobiles that are allowed onto the link – *such that the hub and automobile identifiers are of the road transport system.*
31. The mereology of a hubs is a pair of a set of link identifiers and a set of identifiers of the automobiles that are allowed into the hub – *such that the link and automobile identifiers are of the road transport system.*

32. The mereology of an automobile is a pair of the identifier of *its* automobile aggregate and the set of identifiers of the links and hubs – *of the road net aggregate of the road transport system of which the automobile is a part* it is allowed to travel on.
33. The *slanted texts* above hint at axiomatic constraints.

```
type
28. RNM = AAI × (LI-set × HI-set)
29. AAI = RNI × AI-set
30. LM = HI-set × AI-set
31. HM = LI-set × AI-set
32. AM = AAI × (LI|RI)-set
value
28. mereo_RN: RN → RNM
29. mereo_AA: AA → AAM
30. mereo_L: L → LM
31. mereo_H: H → HM
32. mereo_A: A → AM
axiom
28. let (aai,(lis,his)) = mereo_RN(rn) in
28. aai = aai ∧ lis = lis ∧ his = his end
29. let (rni,ais) = mereo_AA(aa) in
29. rni = rni ∧ ais ⊆ ais end
30. ∀ l:L • l ∈ ls ⇒
30.     let (his,ais) = mereo_L(l) in
30.     his ⊆ his ∧ ais ⊆ ais end
31. ∀ h:H • h ∈ hs ⇒
31.     let (lis,ais) = mereo_H(h) in
31.     lis ⊆ lis ∧ ais ⊆ ais end
32. ∀ a:A • a ∈ as ⇒
32.     let (aai,ris) = mereo_H(a) in
32.     aai = aai ∧ ris ⊆ lis∪his end
```

Routes.

34. The observed road net defines a possibly infinite set of finite length routes:
    Basis Clauses:
35. The null sequence, $\langle\rangle$, of no links or hubs is a route.
36. Any one link or hub, u, of a road net forms a route, $\langle u\rangle$, of length one.
    Inductive Clauses:
37. Let $r_i\hat{}\langle u_i\rangle$ and $\langle u_j\rangle\hat{}r_j$ be two finite routes of a road net.
38. Let $u_{i_{ui}}$ and $u_{j_{ui}}$ be the unique identifiers for u_i, respectively u_j.
39. Let the road (hub or link) identifiers of mereology of u_i be uis and of u_j be ujs. If $u_{i_{ui}}$ is in uis and $u_{j_{ui}}$ is in ujs,
40. then $r_i\hat{}\langle u_i, u_j\rangle\hat{}r_j$ is a route of the road net.
    Extremal Clause:
41. Only such routes which can be formed by a finite number of applications of the clauses form a route.

```
type
34.  R = (L|H)*
value
34  routes: RN ~→ R-infset
34  routes(rn) ≡
35    let rs = {⟨⟩}
36          ∪ {⟨u⟩|u:(L|H)•u ∈ us} ∪
40          ∪ {ri^⟨ui⟩^⟨uj⟩^rj | ui,uj:(L|H) • {ui,uj} ⊆ us
37              ∧ ri^⟨ui⟩,⟨uj⟩^rj:R • {ri^⟨ui⟩,⟨uj⟩^rj} ⊆ rs
38,39           ∧ ui_ui = uid_U(ui) ∧ ui_ui ∈ xtr_UIs(ui)
```

38,41 $\wedge$ uj_ui = uid_U(uj) $\wedge$ uj_ui $\in$ xtr_UIs(uj)} **in**
35 rs **end**

xtr_UIs: (L|H) $\to$ UI-**set**, xtr_UIs(u) $\equiv$ **let** (uis,_)=mereo_(L|H)(u) **in** uis **end**

rs is the smallest [fixed point] set of finite routes that satisfy the equation 35.

42. We can also model routes, as identifier routes, IR, in terms of link and hub identifiers.
43. Given a road net we can examine whether it is strongly connected, i.e., whether any link or hub can be reached from any other link or hub.
44. Et cetera!

type
42. IR = (LI|HI)*
value
42. i_routes: RN $\to$ IR-**infset**
42. i_routes(rn) $\equiv$
42. **let** rs = routes(rn) **in**
42. { $\langle$ uid_(L|H)(r[i]) | i:**Nat** • 1$\leq$i$\leq$**len** r $\rangle$ | r:R • r $\in$ rs } **end**

43. is_connected_RN: RN $\to$ **Bool**
43. is_connected_RN(rn) $\equiv$
43. **let** rs = routes(rn) **in**
43. $\forall$ u,u′:(L|H) • {u,u′}$\subseteq$ $ls \cup hs \Rightarrow \exists$ r:R • r $\in$ rs and {u,u′} $\subseteq$ **elems** r
43. **end**

Attributes. We treat attributes only for atomic sorts. And we show only a very few attribute examples.

Links.

45. Links have lengths.
46. Links have states – *sets of zero, one or two* pairs of hub identifiers – *of their hub mereology*[25].
47. Links have state spaces: a set of all relevant link states – *the link state must at any time be in its link state space.*

type
45. LEN
46. LΣ = (HI$\times$HI)-**set**
47. LΩ = LΣ-**set**
value

[25] – zero expresses that the link is [currently] closed for traffic, one if it is [currently] a one way link, in one or the other direction as indicated by the connecting hub identifiers, or two if it is [currently] a two way link

45. attr_LEN: L → LEN
46. attr_LΣ: L → LΣ
47. attr_LΩ: L → LΩ

axiom

46. ∀ l:L • l ∈ *ls* ⇒
46. **let** (lσ,lω)=(attr_LΣ,attr_LΩ)(l) **in** lσ ∈ lω ∧
46. ∀ (hi′,hi″):(HI×HI) • (hi′,hi″)∈lσ ⇒ {hi′,hi″}⊆*his* **end**

Hubs.

48. Hubs have states: a set of pairs of link identifiers – *of its mereology*.[26]
49. Hubs have state spaces: the set of all relevant hub states – *the current hub state must at any time be in its hub state space.*

type

48. HΣ = (LI×LI)**-set**
49. HΩ = HΣ**-set**

value

48. attr_HΣ: H → HΣ
49. attr_HΩ: H → HΩ

axiom

48. ∀ h:H • h ∈ *hs* ⇒
49. **let** (hσ,hω)=(attr_HΣ,attr_HΩ)(h) **in** hσ ∈ hω ∧
49. ∀ (li′,li″):(LI×LI) • (li′,li″)∈hσ ⇒ {li′,li″}⊆*lis* **end**

Automobiles.

50. Automobiles have positions on links or in hubs (programmable attributes).
    (a) An automobile on a link position is a triplet of (1) a link identifier *of the road net*, (2) and ordered pair of two hub identifiers *of the link mereology*, and (3) a real number properly between 0 and 1.[27]
    (b) An automobile at a hub position is a pair of (1) a hub identifier hi *of the road net*, and (2) an ordered pair of two link identifiers *li′ and li″ of the hub mereology*.[28]
51. Automobiles have a (programmable attribute) history of appearing, at times, at hubs or on links[29].

53c Automobiles have (monitorable attribute) speed and acceleration (plus or minus).

[26] – each pair, (li_j, li_k) expressing that automobiles may [currently] enter the hub from the links identified by li_j and leave the hub to the links identified by li_k

[27] – expressing the fraction along the designated link between the two designated hubs. The type constructor :: is "borrowed" from VDM [23].

[28] – expressing that the automobile at hub hi is on its way between links designated by li′ and li″

[29] We shall define that attribute in items 53c on the facing page

52. Etc.

```
type
50.  APos = onL | atH
50a. onL :: LI × (HI×HI) × F
50a. F = Real, invariant: ∀ f:F • 0<f<1
50b. atH :: HI × (LI×LI)
53c. AHist
51.  Vel, Acc
52.  ...
value
50.  attr_APos: A → APos
51.  attr_Vel: A → Vel, attr_Acc: A → Acc
52.  ...
axiom
50.  ∀ a:A • a ∈ as ⇒
50.    let apos = attr_APos(a) in
50.    case apos:
50a.      onL(li,(fhi,thi),_) →
50a.        li ∈ lis ∧ let (his,_) = mereo_L(retr_L(li)(σ)) in {fhi,thi}⊆his end
50b.      atH(hi,(fli,tli)) →
50b.        hi ∈ his ∧ let (lis,_) = mereo_H(retr_H(hi)(σ)) in {fli,tli}⊆lis end
51.,52.   ...
50.    end end
```

Intentional Pull. We simplify the link, hub and automobile histories – aiming at just showing an essence of the intentional pull concept.

53. With links, hubs and automobiles we can associate history attributes.
    (a) Link history attributes time-stamp record, as an ordered list, the presence of automobiles.
    (b) Hub history attributes time-stamp record, as an ordered list, the presence of automobiles.
    (c) Automobile history attributes time-stamp record, as an ordered list, their visits to links and hubs.

```
type                                    value
53a. LHist = AI ~m~> TIME*              53a. attr_LHist: L → LHist
53b. HHist = AI ~m~> TIME*              53b. attr_HHist: H → HHist
53c. AHist = (LI|HI) ~m~> TIME*         53c. attr_AHist: A → AHist
```

Wellformedness of Event Histories.
Some observations must be made with respect to the above modelling of time-stamped event histories.

54. Each $\tau_\ell : \mathbb{TIME}^*$ is an indefinite list. We have not expressed any criteria for the recording of events: *all the time, continuously* ! (?)
55. Each list of times, $\tau_\ell : \mathbb{TIME}^*$, is here to be in decreasing, *continuous* order of times.
56. Time intervals from when an automobile enters a link (a hub) till it first time leaves that link (hub) must not overlap with other such time intervals for that automobile.
57. If an automobile leaves a link (a hub), at time τ, then it may enter a hub (resp. a link) and then that must be at time τ' where τ' is some infinitesimal, sampling time interval, quantity larger that τ. Again we refrain here from speculating on the issue of sampling !
58. Altogether, ensembles of link and hub event histories for any given automobile define routes that automobiles travel across the road net. Such routes must be in the set of routes defined by the road net.

As You can see, there is enough of interesting modelling issues to tackle !

Formulation of an Intentional Pull.

59. An *intentional pull* of any road transport system, rts, is then if:
    (a) for any automobile, a, of rts, on a link, ℓ (hub, h), at time τ,
    (b) then that link, ℓ, (hub h) "records" automobile a at that time.
60. and:
    (c) for any link, ℓ (hub, h) being visited by an automobile, a, at time τ,
    (d) then that automobile, a, is visiting that link, ℓ (hub, h), at that time.

axiom

```
59a.  ∀ a:A • a ∈ as ⇒
59a.    let ahist = attr_AHist(a) in
59a.    ∀ ui:(LI|HI) • ui ∈ dom ahist ⇒
59b.      ∀ τ:TIME • τ ∈ elems ahist(ui) ⇒
59b.        let hist = is_LI(ui) → attr_LHist(retr_L(ui))(σ),
59b.                   _ → attr_HHist(retr_H(ui))(σ) in
59b.        τ ∈ elems hist(uid_A(a)) end end
60.   ∧
60c.  ∀ u:(L|H) • u ∈ ls∪hs ⇒
60c.    let uhist = attr(L|H)Hist(u) in
60d.    ∀ ai:AI • ai ∈ dom uhist ⇒
60d.      ∀ τ:TIME • τ ∈ elems uhist(ai) ⇒
60d.        let ahist = attr_AHist(retr_A(ai))(σ) in
60d.        τ ∈ elems uhist(ai) end end
```

4.2 Perdurants

Behaviours. We show only the signature and definition of one aspect of one behaviour. That of an automobile at a hub. We refer to [11, Examples 82–83, pages 183–184] for the full set of signatures and definitions for link, hub and automobile behaviours.

Signatures.

61. automobile:
    (a) there is the usual "triplet" of arguments: unique identifier, mereology and static attributes;
    (b) then there are two programmable attributes: the automobile position (cf. Item 50 on page 24), and the automobile history (cf. Item 53c on page 25);
    (c) and finally there are the input/output channel references allowing communication between the automobile and the hub and link behaviours.

    We deviate from RSL in expression these signatures. The deviation amounts to a form of dependent types [38].
62. Similar signatures can be given for
    (a) link and
    (b) hub behaviours.

We omit the modelling of monitorable attributes.

value
61,61a automobile: ai:AI×(_,uis):AM×...
61b → (apos:APos × ahist:AHist)
61c → **in out** {ch[{ai,ui}]|ai:AI,ui:(HI|LI) • ai∈ais ∧ ui ∈ uis} **Unit**
62a link: li:LI×(his,ais):LM×LΩ
62a → LΣ
62a → **in out** {ch[{li,ui}]|li:LI,ui:(AI|HI)**-set** • ai∈ais ∧ li ∈lis∪his} **Unit**
62b hub: hi:HI×(_,ais):HM×HΩ
62b → HΣ
62b → **in out** {ch[{ai,ui}]|hi:HI,ai:AI • ai∈ais ∧ hi ∈ uis} **Unit**

We omit the pre-conditions.

Definitions: Automobile at a Hub.

63. We abstract automobile behaviour at a Hub (hi).
    (a) Either the automobile remains in the hub,
    (b) or, internally non-deterministically,
    (c) leaves the hub entering a link,
    (d) or, internally non-deterministically,
    (e) stops.
    (f) or, internally non-deterministically,
    (g) decides to communicate with the department of vehicles,
    (h) or, externally non-deterministically,
    (i) is contacted by department of vehicles,

We omit the definition of department_of_vehicle (i.e., automobile aggregate) behaviour.

```
63 automobile(ai,(aai,uis),...)(apos:atH(fli,hi,tli),ahist) ≡
63a    (automobile_remains_in_hub(ai,(aai,uis),...)(apos:atH(fli,hi,tli),ahist)
63b    ⌈⌉
63c     automobile_leaving_hub(ai,(aai,uis),...)(apos:atH(fli,hi,tli),ahist)
63d    ⌈⌉
63e     automobile_stop(ai,(aai,uis),...)(apos:atH(fli,hi,tli),ahist)
63f    ⌊⌋
63g     automobile_contacts_dv(ai,(aai,uis),...)(apos:atH(fli,hi,tli),ahist))
63h    []
63i     dv_contacts_automobile(ai,(aai,uis),...)(apos:atH(fli,hi,tli),ahist)
```

64. [63a] The automobile remains in the hub:
    (a) the automobile remains at that hub, "idling",
    (b) informing ("first") the hub behaviour.

```
64 automobile_remains_in_hub(ai,(aai,uis),...)(apos:atH(fli,hi,tli),ahist) ≡
64     let τ = record_𝕋𝕀𝕄𝔼() in
64b    ch[ ai,hi ] ! τ ;
64a    automobile(ai,(aai,uis),...)(apos,upd_hist(τ,hi)(ahist))
64     end

64a  upd_hist: (𝕋𝕀𝕄𝔼×I) → (AHist|LHist|HHist) → (AHist|LHist|HHist)
64a  upd_hist(τ,i)(hist) ≡ hist † [ i ↦ ⟨τ⟩^hist(i) ]
```

65. [63c] The automobile leaves the hub entering a link:
    (a) tli, whose "next" hub, identified by thi, is obtained from the mereology of the link identified by tli;
    (b) informs the hub it is leaving and the link 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.

```
65 automobile_leaving_hub(ai,(aai,uis),...)(apos:atH(fli,hi,tli),ahist) ≡
65a    (let ({fhi,thi},ais) = mereo_L(retr_L(tli)(σ)) in assert: fhi=hi
65b     ( ch[ ai,hi ] ! τ ‖ ch[ ai,tli ] ! τ ) ;
65c     automobile(ai,(aai,uis),...)
65c             (onL(tli,(hi,thi),0),upd_hist(τ,tli)(upd_hist(τ,hi)(ahist))) end)
```

66. [63e] Or the automobile "disappears — off the radar" !

```
66 automobile_stop(ai,(aai,uis),...)(apos:atH(fli,hi,tli),ahist) ≡ stop
```

Similar behaviour definitions can be given for *automobiles on a link*, for *link*s and for *hub*s. Together they must reflect, amongst other things: the time continuity of automobile flow, that automobiles follow routes, that automobiles, links and hubs together adhere to the intentional pull expressed earlier, et cetera. A specification of these aspects must be proved to adhere to these properties.

Initial System. The initial system is the parallel composition of

67. the road net aggregate behaviour,
68. the automobile aggregate behaviour,
69. all automobile behaviours,
70. all link behaviours, and
71. all hub behaviours.

```
value
67.     dept_of_roads(uid_RN(rn),mereo_RN(rn),...)(...)
68.   ‖ dept_of_vehicles(uid_AA(aa),mereo_AA(aa),...)(...)
69.   ‖ {automobile(uid_A(a),mereo_A(a),...)(attr_Apos(a),attr_AHist(a))|a:A•a∈as}
70.   ‖ {link(uid_L(l),mereo_L(l),(attr_LEN(l),attr_LΩ(l)))(attr_LΣ(l),attr_LHist(l))|l:L•l∈ls}
71.   ‖ {hub(uid_H(h),mereo_H(h),attr_HΩ(h))(attr_HΣ(h),attr_HHist(h))|h:H•h∈hs}
```

That's all folks! Neat!?

• • •

Initial Remark Reviewed: *Initially the narratives of the domain description were scant and their counterpart formalisations left many possible interpretations as to what these formal types and function signatures really meant. As the domain description proceeded – now with perdurants: channels and action, event and behaviour signatures and definitions – these meanings were narrowed down, considerably – focusing, finally, on yielding the properties that are deemed necessary and sufficient.*

5 Relevance to Aeronautics and Space

The specific relevance of domain engineering to aeronautics and space will be the subject of this section.

5.1 But First

As a preamble for briefly discussing the relevance of domain engineering to aeronautics and space, we 'complete' our treatment of domain engineering with three small notes.

Domain Modelling Experiments. It is appropriate to mention that the method, i.e., the principles, techniques and tools of domain analysis & description, has been "tuned & honed" by extensive "laboratory work". That is, there has been experimentally researched and developed a number of less-or-more "complete" domain models. In reverse chronological order we mention some:

– 2021: **Assembly Lines**, September, 2021. Techn. Univ. of Denmark
www.imm.dtu.dk/~dibj/2021/assembly/assembly-line.pdf

- 2021: **Shipping**, April 2021. Techn. Univ. of Denmark
  www.imm.dtu.dk/~dibj/2021/ral/ral.pdf
- 2021: **Rivers and Canals**, March 2021. Techn. Univ. of Denmark
  www.imm.dtu.dk/~dibj/2021/Graphs/Rivers-and-Canals.pdf
- 2021: **A Retailer Market**, January 2021. Techn. Univ. of Denmark
  www.imm.dtu.dk/~dibj/2021/Retailer/BjornerHeraklit27January2021.pdf
- 2019: **Container Terminals**, ECNU, Shanghai, China
  www.imm.dtu.dk/~dibj/2018/yangshan/maersk-pa.pdf
- 2018: **Documents**, TongJi Univ., Shanghai, China
  www.imm.dtu.dk/~dibj/2017/docs/docs.pdf
- 2017: **Urban Planning**, TongJi Univ., Shanghai, China
  www.imm.dtu.dk/~dibj/2018/BjornerUrbanPlanning24Jan2018.pdf
- 2017: **Swarms of Drones**, Inst. of Softw., CAS, Peking, China
  www.imm.dtu.dk/~dibj/2017/swarms/swarm-paper.pdf
- 2013: **Road Transport**, Techn. Univ. of Denmark
  www.imm.dtu.dk/~dibj/road-p.pdf
- 2012: **Credit Cards**, Uppsala, Sweden
  www.imm.dtu.dk/~dibj/2016/credit/accs.pdf
- 2012: **Weather Information**, Bergen, Norway
  www.imm.dtu.dk/~dibj/2016/wis/wis-p.pdf
- 2010: **Web-based Transaction Processing**, Techn. Univ. of Vienna, Austria
  www.imm.dtu.dk/~dibj/wfdftp.pdf
- 2010: **The Tokyo Stock Exchange**, Tokyo Univ., Japan
  www.imm.dtu.dk/~db/todai/tse-1.pdf, www.imm.dtu.dk/~db/todai/tse-2.pdf
- 2009: **Pipelines**, Techn. Univ. of Graz, Austria
  www.imm.dtu.dk/~dibj/pipe-p.pdf
- 2007: **A Container Line Industry Domain**, Techn. Univ. of Denmark
  www.imm.dtu.dk/~dibj/container-paper.pdf
- 2002: **The Market**, Techn. Univ. of Denmark
  www.imm.dtu.dk/~dibj/themarket.pdf
- 1995–2004: **Railways**, Techn. Univ. of Denmark - a compendium
  www.imm.dtu.dk/~dibj/train-book.pdf

Requirements Engineering. If our objective for having a domain description is that it serves as a basis for software development, then a [next] phase of development is that of requirements engineering. Chapter 9 of [11] shows how to systematically develop requirements from a domain description.

As we did for domain analysis & description, Sect. 3.5 on page 18, we can do for requirements development: present an informal, but precise specification of the *requirements analysis & description process*.

The "formalisation" below reveals an essence of [11, Chapter 9]. Namely that the requirements development consists of three major stages: domain requirements, DR – which in turn consists of five steps, interface requirements, IR, and machine requirements, MR. The stages of domain and interface requirements development can be further 'decomposed' into steps. The pseudo procedure names these steps. For details we refer to [11, Chapter 9]

```
value
    requirements_analysis_description: RSL⁺Text→ 𝒟 →
                                        (𝒟×𝒟×...×𝒟) → RSL⁺Text
    requirements_analysis_description(rsl_txt)(d)(d1,...,dm) ≡
DR:     let dr=(let drp = domain_projection(rsl_txt)(d) in
                let dri = domain_requirements_instantiation(drp)(d) in
                let drd = domain_requirements_determination(dri)(d) in
                let dre = domain_requirements_extension(drd)(d) in
                let drf = domain_requirements_fitting(dre)((d1,...,dm),d)
                in drf end end end end end) in
IR:     let irp = interface_requirements(drf)(d) in
MR:     let mrp = machine_requirementsn(irp)(d)
        in mrp end end end
```

Here (d1,...,dm) are the "other" requirements with which ((dre),(...,d)) is to be fitted; mrp then represents the full set of requirements from which to develop, in a next phase, the software.

Software Design. The three monographs cum textbooks [8–10] show how to develop software from requirements prescriptions.

5.2 Air Traffic Control, ATC

On the background of the domain to requirements transformation, [11, Chapter 9], and a similar requirements to software design transformation [10], we now claim to have a rigorous path of development from domains to trustworthy software.

An domain, "close", informally speaking, to that of NASA's concerns, is *air traffic control, ATC.*

Future ATCs. Today's ATC is primarily radar-based and human-operated. Tomorrow's ATC appears headed for satellite-orientation and automation.[30]

We suggest, in this paper, that major US and European efforts for formulating the next generation ATCs be supported by *pre-domain* modeling experiments.

Models of proposed ATCs are neither domain models nor requirements models. They are models of virtual ATCs, as [13] formulates a family of models of automobile assembly lines. Such a family can be used to determine values

[30] We refer to:

- ICAO: https://www.icao.int/airnavigation/documents/ganp-2016-interactive.pdf
- US: https://www.faa.gov/nextgen/
- Europe: https://www.easa.europa.eu/domains/air-traffic-management

of future ATC "parameters": which ATC components should undertake which tasks, etc. Their modelling process can also, and this is something new, help experiment with alternative ATC-component or procedure proposals, as a form of "sounding boards".

A Model for Current ATC. In order to develop models for families of ATCs we suggest to first develop a model of the existing, worldwide ATC. A basis for such a model is illustrated in Fig. 3.

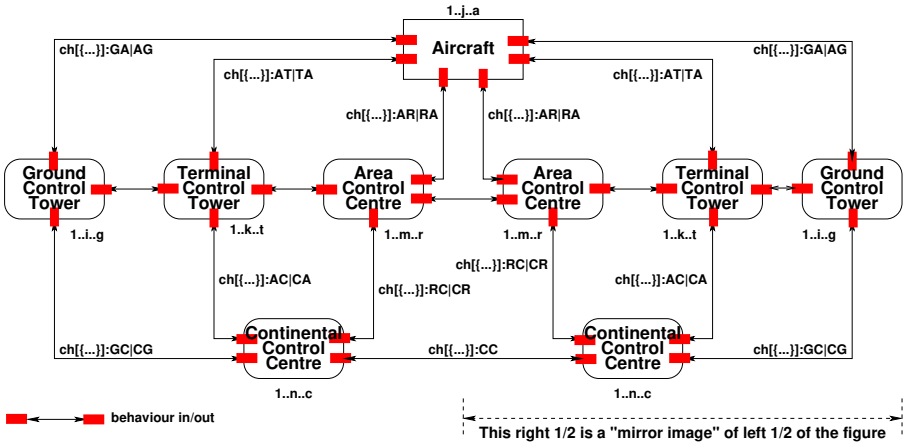

Fig. 3. Conventional Air Traffic Control

Thus we challenge the reader to analyse & describe external qualities (as basically shown in Fig. 3), and states, then internal qualities, first unique identifiers, mereologies, and attributes; then external qualities, first channels, then behaviour signatures and definitions, and finally an initial state.

Now the modellers are well prepared for modelling future ATCs.

• • •

The above suggests that *domain modelling* problems related to aeronautics and space might also be a good idea!

5.3 An Aeronautics & Space Domain

To properly understand *the domain of aeronautics &*[31] *space* we must first analyse various facets of the domain as we see it today. Aeronautics & space, as an

[31] We shall use the ampersand, &, instead of 'and', to emphasize that we speak of one, consolidated topic, not two!

endeavour, is pursued in order to explore space, with space exploration missions "divided" into stages, deploying a variety of technologies, and satellites.[32]

[I] Types of Space Exploration. There are many kinds of space exploration: *earth observation satellites*, *spy satellites*, *communications satellites*, *military satellites*, *satellite navigation*, *space telescopes*, *space exploration* and *space tourism*.

[II] Stages of Space Exploration. There are common stages of missions: the *launch phase* (assembly, test, and launch operations), the *cruise phase*, the *encounter phase* and depending on the state of spacecraft health and mission funding, the *extended operations phase*.

[III] Space Technologies. There are different kinds of space technologies: *spacecraft*, *satellites*, *space stations* and *orbital launch*.

[IV] Types of Satellites and Applications. And there are many types of satellites and applications: *remote sensing satellites*, *navigation satellites*, *geocentric orbit type satellites*, *global positioning systems*, *geostationary satellites*, *drone satellites*, *ground satellites* and *polar satellites*.

• • •

A[n Aeronautics &] Space Control, ASC, Sketch.

An Analysis. *Air traffic control*, ATC, hinted at in Sect. 5.2, can, in contrast to a perceived *aeronautics & space monitoring, communication and control*, i.e., an *air space control*, ASC, it seems, be primarily characterised as follows: (a) ATC is concerned with only one kind of moveable entities: passenger and cargo aircraft whereas an ASC would have to deal with quite a variety of moveable entities; (b) ATC is independent of the multitude of national and international air carriers, whereas, it seems, today's national aeronautics & space efforts and their monitoring, communications and controls are fragmented into national agencies who are also the [main] stakeholders in the monitored, etc., space efforts; (c) ATC can, today, be partly identified in terms of *aircraft* (one, unifying concept), *ground control towers*, *terminal controls*, *area controls and continental controls*; and (d) ATC responsibility is shared by many (overflown) nations.

• • •

There is today an estimated 3.500 man-made space objects "up there", right now! Each such space "mission" lasting for up to many years. In contrast there is, today, an estimated 10.000 aircraft in flight at any moment. Each such flight lasting between 1/2 hour and 14+ hours. *We proceed, therefore, on the assumption that a global, multi-nation co-ordinated ASC is required.*

• • •

[32] The following text is adapted from various NASA Web pages found under: https://www.nasa.gov.

The As Yet Unknowns. The above rather terse and simplified analysis left open a number of issues: (i) Can a perceived, "single", ASC be devised to handle all facets of space exploration, applications and technologies ? (ii) Can a perceived ASC, of a next future, be "pinned down" to two or more separate physical, stationary parts (and behaviours) such as the *aircraft, ground control towers, terminal controls, area controls and continental controls* ? (iii) Is it too early to consolidate matters ? That is, do political concerns and technological advances stand in the way of consolidation ?

A Suggestion. It is therefore suggested that the concepts of *domain science & engineering* be applied to the issues of whether (α) a national and/or an international, or a global, ASC; (β) one or several distinct ASCs, one per type of space exploration ([I]) or satellite ([IV]) or application ([IV]); and (γ) in case (β) recommends several, typed, ASCs, how to coordinate these.

In doing so *domain science & engineering* is being used not to model an existing, but a contemplated domain ! Thus the modelling may involve modelling a variety of choices. In [13], the authors show how domain modelling can be formulated such that optimisation of assembly line production can be investigated. Similar possibilities could be investigated in connection with modelling proposed aeronautic & space control. Domain science & engineering may cast a new kind of light on these issues.

Thus it is suggested that the US Government FAA and NASA, and, in Europe, the EUROCONTROL and ESA, separately or jointly, and these in cooperation with many other space agencies[33], co-operate on researching and experimentally developing domain models for aeronautics & space.

6 Conclusion

The title of this paper had the prefix *'An Essence of'*. The *'An'*, rather than a *'The'*, shall indicate that there are many essential aspects of *domain engineering*. Some essences of *domain science & engineering* are (i) a basis in *philosophy*; (ii) an interpretation of *transcendental deduction*; (iii) *intentional pull*, an interpretation of *"gravitational pull"* being a core property of domains; and (iv) that domain analysis & description 'wavers' between *science* and *engineering*, being conducted in a context of more-or-less following *formal method principles, techniques* and *tools* – yet searching and deciding *informally* for the entities to analyse & describe.

There may be other 'essences' ![34] We refer to [11] for other aspects.

[33] ICAO (UN), Roscosmos (Russia), CNSA (China), ISRO (India), JAXA (Japan), AEB (Brazil), CSA (Canada), ASA (Australia) and others

[34] It appears to have become fashionable to include the idea of 'essence' in the title of methods or books:

– https://essence.ivarjacobson.com/services/what-essence: *The Essence of Software Engineering. The SEMAT kernel.* Ivar Jacobson, Pan-Wei

The proposed domain modelling method of this paper, and hence [11], raises a great many research issues:

- The issue of *intentional pull* is also only briefly sketched, paragraph *Intentional Pull* Sect., 3.2 on page 14.
- There is the issue of *the modelling of continuity*, illustrated in paragraph *Intentional Pull* of Sect. 4.1 on page 25. In modelling aeronautics & space there is a more general need for modelling continuity. 'Formal Methods', so far, has yet to "deliver" on this: the ability to freely alternate between discrete, logical models and continuous, say differential and integral calculus-based models.
- There is a carefully thought out and apparently complete analysis & description calculus for endurants, but there is no analysis & description calculus for perdurants!?

6.1 Acknowledgments

The front matter preface of [11] ends with an extensive list of acknowledgments. For this paper I repeat acknowledging three persons: *Kai Sørlander* from whose philosophical works and from our personal interaction I have benefited; my editor at Springer, *Ronan Nugent*, whose steadfast and tireless work also lies behind [11]; and *Klaus Havelund* for being a great discussion partner over now many years. I also thank the NASA Formal Methods Symposium for the invitation which has afforded me the possibility to correct, clarify and simplify a number of issues wrt. RSL, RSL$^+$Text, and domain analysis and description methodology: its principles, techniques and tools.

References

1. Aaronson, S.: Quantum Computing since Democritus. Cambridge University Press (2013)
2. Ahbel-Rappe, S.: Socrates: A Guide for the Perplexed. A&C Black (Bloomsbury), ISBN 978-0-8264-3325-1 (2011)
3. et al., W.D.R.: Plato's Theory of Ideas. Oxford University Press (1963)
4. Aristotle: Categories. On Interpretation. Prior Analytics. Harvad University Press [Loebb Classical Library, translated by H.P. Cooke and Hugh Tredenick] (1938)
5. Audi, R.: The Cambridge Dictionary of Philosophy. Cambridge University Press, The Pitt Building, Trumpington Street, Cambridge CB2 1RP, England (1995)
6. Berger, B., Whistler, D.: The Schelling Reader. Bloomsbury Publishing PLC (2020)

Ng, Paul E. McMahon, Ian Spence, and Svante Lidman. ACM Queue, October 24, 2012, Volume 10, issue 10.

- https://press.princeton.edu/books/hardcover/9780691225388/-the-essence-of-software: *The Essence of Software: Why Concepts Matter for Great Design.* Daniel Jackson, Nov.16, 2021.

7. Berkeley, G.: Philosophical Works, Including the Works on Vision. Everyman edition, London (1975 (1713))
8. Bjørner, D.: Software Engineering, Vol. 1: Abstraction and Modelling. Texts in Theoretical Computer Science, the EATCS Series, Springer (2006)
9. Bjørner, D.: Software Engineering, Vol. 2: Specification of Systems and Languages. Texts in Theoretical Computer Science, the EATCS Series, Springer (2006), Chapters 12–14 are primarily authored by Christian Krog Madsen.
10. Bjørner, D.: Software Engineering, Vol. 3: Domains, Requirements and Software Design. Texts in Theoretical Computer Science, the EATCS Series, Springer (2006)
11. Bjørner, D.: Domain Science & Engineering – A Foundation for Software Development. EATCS Monographs in Theoretical Computer Science, Springer (2021)
12. Bjørner, D.: A Domain Science & Engineering Interpretation of Sørlander's Philosophy, http://www.imm.dtu.dk/~dibj/2022/sorlander/Sorlander.pdf. Tech. rep., Technical University of Denmark, Fredsvej 11, DK-2840 Holte, Denmark (September 2022), currently incomplete.
13. Bjørner, N., Levatich, M., Lopes, N.P., Rybalchenko, A., Vuppalapati, C.: Supercharging plant configurations using Z3. In: Stuckey, P.J. (ed.) Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 18th International Conference, CPAIOR 2021, Vienna, Austria, July 5-8, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12735, pp. 1–25. Springer (2021). https://doi.org/10.1007/978-3-030-78230-6_1, https://doi.org/10.1007/978-3-030-78230-6_1
14. Butterfield, J., Earmann, J. (eds.): Philosophy of Physics. Elsevier (2006), Handbook of The Philosophy of Science
15. Carnap, R.: Der Logische Aufbau der Welt. Weltkreis, Berlin (1928)
16. Carnap, R.: The Logical Syntax of Language. Harcourt Brace and Co., N.Y. (1937)
17. Carnap, R.: Introduction to Semantics. Harvard Univ. Press, Cambridge, Mass. (1942)
18. Carnap, R.: Meaning and Necessity, A Study in Semantics and Modal Logic. University of Chicago Press (1947, 1956)
19. Casati, R., Varzi, A.C.: Parts and Places: the structures of spatial representation. MIT Press (1999)
20. Clarke, E.M., Henzinger, T.A., Veith, H., Bloem, R.: Handbook of Model Checking. Springer (2018)
21. Couprie, D.L., Kocandrle, R.: Anaximander: Anaximander on Generation and Destruction. x (Springer (Briefs in Philosophy Series))
22. Darwin, C.: Origin of Species. Penguin Putnam (2003), introduction by Sir Julian Huxley
23. Dawes, J.: The VDM-SL reference guide, vol. 18. Pitman London (1991)
24. Descartes, R.: Discours de la méthode. Texte et commentaire par Étienne Gilson. Paris: Vrin (1987)
25. andHenry Folse, J.F. (ed.): Niels Bohr and the Philosophy of Physics: Twenty-First-Century Perspectives. Bloomsbury Academic (2019)
26. Frege, G. (ed.): Begriffsschrift – *"a formula language, modelled on that of arithmetic, for pure thought."*. Verlag von Louis Nebert, Halle (1879)
27. George, C., Haxthausen, A.E.: The logic of the RAISE specification language. Comput. Artif. Intell. **22**(3-4), 323–350 (2003), http://www.sav.sk/index.php?lang=en&charset=ascii&doc=journal&part=list_articles&journal_issue_no=882#abstract_2729
28. George, C.W., Haff, P., Havelund, K., Haxthausen, A.E., Milne, R., Nielsen, C.B., Prehn, S., Wagner, K.R.: The RAISE Specification Language. The BCS Practitioner Series, Prentice-Hall, Hemel Hampstead, England (1992)

29. George, C.W., Haxthausen, A.E., Hughes, S., Milne, R., Prehn, S., Pedersen, J.S.: The RAISE Development Method. The BCS Practitioner Series, Prentice-Hall, Hemel Hampstead, England (1995)
30. Gödel, K.: Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. Monatshefte für Mathematik Physik **38**, 173–198 (1931), [English translation in van Heijenoort 1967, 596–616, and in Gödel, 1986, 144195]
31. Hegel, G.W.F.: Wissenschaft der Logik. Hofenberg (2016 (1812–1816))
32. Heidegger, M.: Parminedes. Indiana University Press (1998)
33. Heisenberg, W.: Physics and Philosophy: The Revolution in Modern Science. Harper Perennial Modern Classics (2007)
34. Hierons, R.M., Bowen, J.P., Harmann, M. (eds.): Formal Methods and Testing: An Outcome of the Fortest Network. Springer LNCS 4949 (2008)
35. Hoare, C.A.R.: Communicating Sequential Processes. Communications of the ACM **21**(8) (Aug 1978)
36. Hoare, C.A.R.: Communicating Sequential Processes. C.A.R. Hoare Series in Computer Science, Prentice-Hall International (1985)
37. Hoare, C.A.R.: Communicating Sequential Processes. C.A.R. Hoare Series in Computer Science, Prentice-Hall International (1985), published electronically: usingcsp.com/-cspbook.pdf (2004)
38. Hofmann, M.: Syntax and semantics of dependent types. In: Extensional Constructs in Intensional Type Theory, pp. 13–54. Springer (1997)
39. Hume, D.: Enquiry Concerning Human Understanding. Squashed Editions (2020 (1758))
40. Husserl, E.: Ideas. General Introduction to Pure Phenomenology. Routledge (2012)
41. Irvine, A.D. (ed.): Philosophy of Mathematics. Elsevier Science & Technology (2006)
42. Jackson, M.A.: Software Requirements & Specifications: a lexicon of practice, principles and prejudices. ACM Press, Addison-Wesley, Reading, England (1995)
43. James, D., Zoller, G.: Cambridge Companion to Fichte. Cambridge University Press (2016)
44. Kant, I.: Critique of Pure Reason. Penguin Books Ltd (2007 (1787))
45. Kennedy, H.C. (ed.): Selected works of Giuseppe Peano, with a biographical sketch and bibliography. London: Allen & Unwin (1973)
46. Leibniz, G.W.: The Philosophical Writings of Leibniz. Hassell Street Press (2021)
47. Little, W., Fowler, H., Coulson, J., Onions, C.: The Shorter Oxford English Dictionary on Historical Principles. Clarendon Press, Oxford, England (1973, 1987), Two vols.
48. Locke, J.: An Essay Concerning Human Understanding. Penguin Classics (1998 (1689))
49. Maxwell, J.C.: A Treatise on Electricity and Magnetism. Dover reprint (1954 (1892)), 3rd edition, vols. 1–2
50. Mendel, G., Bateson, W. (eds.): Mendel's Principles of Heredity. Franklin Classics Trade Press (2018)
51. Mercer, J.E.: The Mysticism Of Anaximenes And The Air. Kessinger Publishing, LLC (2010)
52. O'Grady, P.: Thales of Miletus. Routledge (Western Philosophy Series) (2002)
53. Pears, D.: Russell's Logical Atomism. Fontana Collins (1972)
54. Planck, M.: Eight Lectures on Theoretical Physics. Dover Publications (2003 (1915))
55. Popper, K.R.: Logik der Forschung. Julius Springer Verlag, Vienna, Austria (1934 (1935)), english version [56]
56. Popper, K.R.: The Logic of Scientific Dicovery. Hutchinson of London, 3 Fitzroy Square, London W1, England (1959,. . . ,1979), translated from [55]

57. Popper, K.R.: Conjectures and Refutations. The Growth of Scientific Knowledge. Routledge and Kegan Paul Ltd. (Basic Books, Inc.), 39 Store Street, WC1E 7DD, London, England (New York, NY, USA) (1963,. . . ,1981)
58. Popper, K.R.: A Pocket Popper. Fontana Pocket Readers, Fontana Press, England (1983), an edited collection, Ed. David Miller
59. Roscoe, A.W.: Theory and Practice of Concurrency. C.A.R. Hoare Series in Computer Science, Prentice-Hall (1997), http://www.comlab.ox.ac.uk/people/bill.-roscoe/publications/68b.pdf
60. Russell, B.: On Denoting. Mind **14**, 479–493 (1905)
61. Russell, B.: The Problems of Philosophy. Home University Library, London (1912), oxford University Press paperback, 1959 Reprinted, 1971-2
62. Russell, B.: Introduction to Mathematical Philosophy. George Allen and Unwin, London (1919)
63. Russell, B.: "Preface," Our Knowledge of the External World. G. Allen & Unwin, Ltd., London (1952)
64. Sannella, D., Tarlecki, A.: Foundations of Algebraic Semantcs and Formal Software Development. Monographs in Theoretical Computer Science, Springer, Heidelberg (2012)
65. Schneider, S.: Concurrent and Real-time Systems — The CSP Approach. Worldwide Series in Computer Science, John Wiley & Sons, Ltd., Baffins Lane, Chichester, West Sussex PO19 1UD, England (January 2000)
66. Sørlander, K.: Det Uomgængelige – Filosofiske Deduktioner [The Inevitable – Philosophical Deductions, with a foreword by Georg Henrik von Wright]. Munksgaard · Rosinante (1994), 168 pages
67. Sørlander, K.: Under Evighedens Synsvinkel [Under the viewpoint of eternity]. Munksgaard · Rosinante (1997), 200 pages
68. Sørlander, K.: Den Endegyldige Sandhed [The Final Truth]. Rosinante (2002), 187 pages
69. Sørlander, K.: Indføring i Filosofien [Introduction to The Philosophy]. Informations Forlag (2016), 233 pages
70. Spinoza, B.: Ethics, Demonstrated in Geometrical Order. The Netherlands (1677)
71. Wallace, A.R.: The Annotated Malaysian Archipelago. National Uniersity of Singapore Press (2014), edited by John Van Wyhe
72. Whitehead, A.N., Russell, B.: Principia Mathematica, 3 vols. Cambridge University Press (1910, 1912, and 1913), second edition, 1925 (Vol. 1), 1927 (Vols 2, 3), also Cambridge University Press, 1962
73. Wittgenstein, L.J.J.: Tractatus Logico–Philosophicus. Oxford Univ. Press, London ((1921) 1961)
74. Wittgenstein, L.J.J.: Philosophical Investigations. Oxford Univ. Press (1958)
75. Wolfe, C.T., Huneman, P., Reydon, T.A. (eds.): History, Philosophy and Theory of the Life Sciences. Springer (2013)
76. Wright, M.: Empedokles: The Extant Fragments. Hackett Publishing Company, Inc. (1995)