Domain Analysis: Endurants An Analysis & Description Process Model

Dines Bjørner

Fredsvej 11, DK-2840 Holte, Danmark DTU^{*} Compute[†], DK-2800 Kgs. Lyngby, Denmark E–Mail: bjorner@gmail.com, URL: www.imm.dtu.dk/~dibj

May 30, 2013: 19:35

Abstract

We present a summary, Sect. 2, of a rather complex structure of domain analysis $\&^1$ description concepts: techniques and tools. And we link, in Sect. 3, these concepts, embodied in *domain analysis prompts* and *domain description prompts*, in a model of how a diligent domain analyser cum describer would use them. We claim that both sections, Sects. 2–3 are contributions to a methodology of software engineering. This paper is based on that of [7].

Contents

1	Intro	ntroduction						
	1.1	A Preamble						
	1.2	Proper Introduction						
2	A Si	A Summary of The TripTychTripTych@TripTych Domain Analysis Approach						
	2.1	Hierarchical versus Compositional Analysis & Description						
	2.2	Domains						
	2.3	Sorts, Types and Domain Analysis	ļ					
	2.4	Entities and Qualities	ļ					
		2.4.1 Entities	ļ					
		242 Qualities						
	25	Endurants and Perdurants						
	2.0	251 Endurants						
		2.5.2 Perdurants						
	2.6	Discrete and Continuous Endurants						
	2.0	2.6.1 Discrete Endurants						
		2.6.1 Districte Endurants						
	07	2.0.2 Continuous Endurants						
	2.1	Discrete and Continuous Perdurants						
	2.8	Atomic and Composite Discrete Endurants						
		2.8.1 Atomic Endurants	,					
		2.8.2 Composite Endurants						
	2.9	Part Observers	,					
		2.9.1 Composite Sorts						
		2.9.2 Sort Models						

*DTU: Technical University of Denmark

[†]The former **DTU Informatics** was renamed, by Jan.1, 2013, into **DTU Compute**. It now consists of the former Departments of Mathematics, Computer Science & Engineering and Mathematical Modeling (i.a., Applied Math.). Its English long name is: **Department of Mathematics and Computer Science**.

¹We use the empersand '&' between two terms a and b to emphasize that the term a&b is one.

Domain Analysis: Endurants

	2.10	Material Observers	8
	2.11	2.11.1. External and Internal Qualities	9
	2 1 2		9
	2.13	Mereology	9
	2.14	Attributes	10
		2.14.1 Shared Attributes	10
	2.15	Discussion	10
-			
3	A Pr	rompt & Description 'Method'	10
	3.1 2.2	A Summary of Prompts	10
	3.2 3.3	Initialising the Domain Analysis & Description Process	11
	3.4	A Domain Analysis & Description Process	11
	3.5	Analysis & Description of Endurants	12
		3.5.1 Analysis & Description of Part Sorts	13
		3.5.2 Analysis & Description of Part Materials	14
		3.5.3 Analysis & Description of Material Parts	15
		3.5.4 Analysis & Description of Composite Endurants	15
		3.5.5 Analysis & Description of Concrete Sort Types	15
		3.5.6 Analysis & Description of Abstract Sorts	17
		3.5.7 Analysis & Description of Unique Identifiers	17
		3.5.8 Analysis & Description of Mereologies	17
	2.0	3.5.9 Analysis & Description of Part Attributes	18
	3.0	Discussion of The Model	18
		3.0.1 Termination	18
		3.6.3 Order of Analysis & Description: A Meaning of '@'	19
		364 Laws of Description Prompts	19
			10
4	Cond	clusion	19
	4.1	Comparison to Other Work	19
		4.1.1 Domain Analysis	19
	4.0	4.1.2 Methodology	20
	4.2	What Have We Achieved	20
		4.2.1 Domain Analysis	20
	43	Future Work	20
	4.4	Acknowledgements	21
5	Bibli	iography	21
	5.1	Bibliographical Notes	21
	5.2	References	21
^	Dino	line Endurants	22
A		Parts	22
	A.2	Part Identification and Mereology	24
		A.2.1 Unique Identification	24
		A.2.2 Unique Identifiers	24
		A.2.3 Mereology	24
	A.3	Part Concepts	25
		A.3.1 Pipe Routes	25
		A.3.2 Well-formed Routes	26
		A.3.3 Embedded Routes	27
	A 4	A.3.4 A Theorem	27
	А.4 Л Б	Naterials	28 29
	A.5	Δ 5.1 Part Attributes	2ð 28
		A.5.2 Flow Laws. 1	20 29
			20
В	Inde	xes	30
	B.1	Index of Definitions	30
	B.2	Index of Concepts	33

© Dines Bjørner 2013, DTU Informatics, Techn.Univ.of Denmark – May 30, 2013: 19:35 2Domain Analysis: Endurants — An Analysis & Description Process Model

 $\mathbf{2}$

1 Introduction

1.1 A Preamble

We bring the introductory lines from a recent four page position statement of Don Batory²: Why (Meta-)Theories of Automated Software Design Are Essential: A Personal Perspective, for the SEMAT workshop on a General Theory of Software Engineering, San Francisco, May 26, 2013, http://semat.org/?page_id=632:

"Consider the first two definitions of 'science' from dictionary.com:

- 1. a branch of knowledge or study dealing with a body of facts or truths systematically arranged and showing the operation of general laws: the mathematical sciences.
- 2. systematic knowledge of the physical or material world gained through observation and experimentation.

The dominant paradigm today in Software Engineering (SE) is for referees to insist on a rigorous hypothesis evaluation of a proposed technique. A set of tests (observations) must be conducted by an author and a careful analysis of one or more hypotheses must be presented. This is the scientific method. It closely matches Definition 2 and the intended use of experimental methods in SE. To me, these are 'pre-theory' activities.

To put this into perspective, a colleague once told me: "Empirical studies helped design spacecraft, but it was the theory of gravity that took us to the moon". Theories are the big ideas in science, not empirical studies. Empirical studies help shape and determine the validity of laws. There are examples of Definition 1 in SE, although most software engineers would never recognize them as such."

I wish to take Don Batory's lines as also mine!

Please keep that in mind should you read this paper further.

1.2 Proper Introduction

Domain Analysis: Endurants — Domain Analysis: Endurants

Before software can be designed we must have a reasonably good grasp of its requirements. Before requirements can be prescribed we must have a reasonably good grasp of the domain in which the software is to reside. So we turn to domain analysis & description as a means to obtain and record that 'grasp'. In this paper we summarise an approach to domain analysis & description recorded in more detail in [7]. Thus this paper is based on [7]. This paper is one in a series of papers on domain science & engineering. In [3] we present techniques related to the analysis and description of domain facets. In [1] we investigate some research issues of domain science. And in [2] we show how to systematically "transform" domain descriptions into requirements prescriptions. The paper [9] examines possible contributions of domain science & engineering th computation for the humanities. It is expected that the present paper may be followed by respective ("spin-off") papers on Perdurants [8], A Formal Model of Prompts [5], Domain Facets (cf. [3]) [6], and On Deriving Requirements From Domain Descriptions (cf. [2]) [10]. The structure of this paper is as follows: First, Sect. 2 we present a terse summary

 $^{^2 \}rm Department$ of Computer Science, University of Texas at Austin, Austin, Texas, USA, batory@cs.utexas.edu

Domain Analysis: Endurants

of a system of domain analysis & description concepts focused on endurants. This summary is rather terse, and is a "tour de force". Section 2 is one of the two main sections of this paper. Section 3 suggest a formal-looking model of the structure of *domain analysis prompts* and *domain description prompts* introduced in Sect. 2. It is not a formalisation of domains, but of the domain analysis & description process. Domains are usually not computationally tractable. Less so is the domain analysis & description processes. Finally, Sect. 4 concludes this paper. An appendix, Appendix A, presents a domain description of a [class of] pipeline systems. Some seminars over the underlying paper may start by a brief presentation of this model. The reader is invited to browse this pipeline system model before, during and/or after reading Sects. 2–3.

2 A Summary of The TripTych Domain Analysis Approach

2.1 Hierarchical versus Compositional Analysis & Description

In this paper we choose, what we shall call, a **'hierarchical analysis'** approach which is was based on decomposing an understanding of a domain from the "overall domain" into its components, and these, if not atomic, into their subcomponents \bullet In contrast we could have chosen a **'compositional analysis'** approach which starts with an understanding of a domain from its atomic endurants and composes these into composite ones, finally ending up with an "overall domain" description \bullet

2.2 **Domains**

A 'domain' is characterised by its observable, i.e., manifest *entities* and their *qualities* \bullet ³ **Example 1.** Domains: *a road net, a container line, a pipeline, a hospital* \bullet ⁴

2.3 Sorts, Types and Domain Analysis

By a 'sort' (or 'type' which we take to be the same) we shall understand the largest set of entities all of which have the same qualities⁵ • Example 2. Sorts: Links of any road net form a sort. So does hubs. The largest set of (well-formed) collections of links form a sort. So does similar collections of hubs. The largest set of road nets (containing well-formed collections of hubs and links) form a sort •

By **'domain analysis'** we shall understand a process whereby a **domain analyser** groups entities of a domain into sorts (and types) • The rest of this paper will outline a class of domain analysis principles, techniques and tools.

2.4 Entities and Qualities

2.4.1 Entities

By an **'entity'** we shall understand a **phenomenon** that can be **observed**, i.e., be seen or touched by humans, or that can be **conceived** as an **abstraction** of an entity • The method can thus be

© Dines Bjørner 2013, DTU Informatics, Techn.Univ.of Denmark – May 30, 2013: 19:35 4 Domain Analysis: Endurants — An Analysis & Description Process Model

 $^{^{3}\}textbf{Definitions}$ start with a single quoted 'term' and conclude with a \blacklozenge

⁴Examples conclude with a

⁵Taking a sort (type) to be the largest set of entities all of which have the same qualities reflects Ganter & Wille's notion of a 'formal concept' [11].

Science & Engineering

said to provide the *domain analysis prompt*: is_entity where is_entity(θ) holds if θ is an entity. **Example 3.** Entities: (a) a road net, (b) a link⁶ of a road net, (c) a hub⁷ of a road net; and (d) insertion of a link in a road net, (e) disappearance of a link of a road net, and (f) the movement of a vehicle on a road net

2.4.2Qualities

By a 'quality' of an entity we shall understand a property that can be given a name and precisely measured by physical instruments or otherwise identified • Example 4. Quality cadestral location of a hub, hub state⁸, hub state space⁹, etcetera **Example 5**. Names: Quality Values: the name of a road net, the ownership of a road net, the length of a link, the *location of a hub*, etcetera

Endurants and Perdurants 2.5

Entities are either endurants or are perdurants.

2.5.1Endurants

By an 'endurant entity' (or just, an endurant) we shall understand that can be observed or conceived, as a "complete thing", at no matter which given snapshot of time. Were we to "freeze" time we would still be able to observe the entire endurant • Thus the method provides a *domain analysis prompt*: $is_endurant$ where $is_endurant(e)$ holds if entity e is an endurant. **Example 6.** Endurants: Items (a–b–c) of Example 2.4.1 are endurants; so are the pipes, valves, and pumps of a pipeline.

2.5.2**Perdurants**

By a 'perdurant entity' (or just, an perdurant) we shall understand for which only a fragment exists if we look at or touch them at any given snapshot in time, that is, where we to freeze time we would only see or touch a fragment of the perdurant • Thus the method provides a domain analysis prompt: is_perdurant where is_perdurant(e) holds if entity e is a perdurant. **Example 7.** Perdurants: Items (d–e–f) of Example 2.4.1 are perdurants; so are the insertion of a hub, removal of a link, etcetera

2.6**Discrete and Continuous Endurants**

Entities are either discrete or are continuous.

2.6.1**Discrete Endurants**

By a 'discrete endurant' we shall understand something which is separate or distinct in form or concept, consisting of distinct or separate parts • We use the term 'part' for discrete endurants, that is: $is_part(p) \equiv is_endurant(p) \land is_discrete(p) \bullet$ Thus the method provides a domain analysis prompt: is_discrete where is_discrete(e) holds if entity e is discrete.

Domain Analysis: Endurants — Domain Analysis: Endurants

⁶A link: a street segment between two adjacent hubs

 $^{^7\}mathrm{A}$ hub: an intersection of street segments

⁸From which links can one reach which links at a given time.

 $^{^9\}mathrm{Set}$ of all hub states over time.

Example 8. Discrete Endurants: The examples of Example 2.5.1 on the facing page are all discrete endurants

2.6.2 Continuous Endurants

By a 'continuous endurant' we shall understand something which is prolonged without interruption, in an unbroken series or pattern • We use the term 'material' for continuous endurants • Thus the method provides a *domain analysis prompt*: is_continuous where is_continuous(e) holds if entity e is continuous. Example 9. Continuous Endurants: The pipes, valves, pumps, etc., of Example 2.5.1 on the preceding page may contain oil; water of a hydro electric power plant is also a material (i.e., a continuous endurant).

2.7 Discrete and Continuous Perdurants

We are not covering perdurants in this paper.

2.8 Atomic and Composite Discrete Endurants

Discrete endurants are either atomic or are composite.

2.8.1 Atomic Endurants

By an 'atomic endurant' we shall understand a discrete endurant which in a given context, is deemed to *not* consist of meaningful, separately observable proper sub-parts \bullet The method can thus be said to provide the *domain analysis prompt*: is_atomic where is_atomic(p) holds if p is an atomic part. Example 10. Atomic Parts: Examples of atomic parts of the above mentioned domains are: aircraft (of air traffic), demand/deposit accounts (of banks), containers (of container lines), documents (of document systems), hubs, links and vehicles (of road traffic), patients, medical staff and beds (of hospitals), pipes, valves and pumps (of pipeline systems), and rail units and locomotives (of railway systems).

2.8.2 Composite Endurants

By a 'composite endurant' we shall understand a discrete endurant which in a given context, is deemed to *indeed* consist of meaningful, separately observable proper sub-parts • The method can thus be said to provide the *domain analysis prompt*: is_composite where is_-composite(*p*) holds if *p* is an a composite part. Example 11. Composite Parts: Examples of atomic parts of the above mentioned domains are: airports and air lanes (of air traffic), banks (of a financial service industry), container vessels (of container lines), dossiers of documents (of document systems), routes (of road nets), medical wards (of hospitals), pipelines (of pipeline systems), and trains, rail lines and train stations (of railway systems).

2.9 Part Observers

From atomic parts we cannot observe any sub-parts. But from composite parts we can.

Science & Engineering

2.9.1 Composite Sorts

For composite parts, p, the domain description prompt observe_parts(p) yields some formal description text according to the followig schema: type P_1 , P_2 , ..., P_n ; value obs_ P_1 : $P \rightarrow P_1$, obs_ P_2 : $P \rightarrow P_2$,...,obs_ P_n : $P \rightarrow P_n$; where sorts P_1 , P_2 , ..., P_n must be disjoint. A proof obligation may need be discharged to secure disjointness.

2.9.2 Sort Models

A part sort is an abstract type. Some part sorts, Ps, may have a concrete type model. Here we consider only two such models: one model is as sets of parts of sort Ps. the other model has parts being of either of two or more alternative, disjoint sorts. The *domain analysis prompt*: has_concrete_type(*p*) holds if part *p* has a concrete type. In this case the *domain description prompt* observe_concrete_type(*p*) yields some *formal description text* according to the followig *schema*, either type A, B, ..., C, T = $\mathcal{E}(A,B,...,C)$; value obs_T: P \rightarrow T; where A,B,...,C are either (new) part sorts or are auxiliary (abstract or concrete) types¹⁰; or: type T = P1|P2|...|PN, P₁,P₂,...,P_n, P1::mkP1(s_p:P_1),P2::mkP1(s_p:P_2),...,PN::mkP1(s_p:P_n); value obs_T: P \rightarrow T;

2.10 Material Observers

Some parts p of sort P may contain material. The domain analysis prompt has_material(p) holds if composite part p contains one or more materials. The domain description prompt observe_material_sorts(p) yields some formal description text according to the followig schema: type M_1 , M_2 , ..., M_m ; value obs_ M_1 : $P \rightarrow M_1$, obs_ M_2 : $P \rightarrow M_2$, ..., obs_ M_m : $P \rightarrow M_m$; where values, m_i , of type M_i satisfy is_material(m) for all i; and where M_1 , M_2 , ..., M_m must be disjoint sorts. Example 12. Part Materials: The pipeline parts p pipes, valves, pumps, etc., contains some either liquid material, say crude oil. or gaseous material, say natural gas

Some material m of sort M may contain parts. The *domain analysis prompt* has_parts(m)holds if material m contains one or more parts. The domain description prompt observe_ $part_sorts(m)$ yields some formal description text according to the followig schema: type P_1 , P_2 , ..., P_n ; value obs_ P_1 : $M \rightarrow P_1$, obs_ P_2 : $M \rightarrow P_2$,...,obs_ P_m : $M \rightarrow P_m$; where values, p_i , of type P_i satisfy $is_part(p_i)$ for all i; and where P_1 , P_2 , ..., P_n must be disjoint sorts. **Example 13.** Material and Part Relations: A global transport system can, for example, be described as primarily containing navigable waters, land areas and air — as three major collections of parts. Navigable waters contain a number of "neighbouring" oceans, channels, canals, rivers and lakes reachable by canals or rivers from other navigable waters (all of which are parts). The part sorts of navigable waters has water materials. All water materials has (zero or more) parts such as vessels and sea-ports. Land areas contain continents, some of which are neighbouring (parts), while some are isolated (that is, being islands not "border-"connected to other continents). Some land areas contain harbour. Harbours and seaports are overlapping parts sharing many attributes. And harbours and seaports are connected to road and rail nets. Etcetera, etcetera The above example, Example 2.10, help motivate the concept of mereology (see below).

¹⁰The *domain analysis prompt*: **sorts_of**(t) yields a subset of {A,B,...,C}.

Domain Analysis: Endurants

2.11 Endurant Properties

2.11.1 External and Internal Qualities

We have already, above, treated the following properties of endurants: is_discrete, is_continuous, is_atomic, is_composite and has_material. We may think of those properties as external qualities. In contrast we may consider the following internal qualities: has_unique_identifier (parts), has_mereology (parts) and has_attributes (parts and materials).

2.12 Unique Identifiers

Without loss of generality we can assume that every part has a unique identifier¹¹. A **'unique part identifier'** (or just unique identifier) is a further undefined, abstract quantity. If two parts are claimed to have the same unique identifier then they are identical, that is, their possible mereology and attributes are (also) identical \bullet The *domain description prompt*: observe_unique_identifier(p) yields some *formal description text* according to the followig *schema*: type PI; value uid_P: P \rightarrow PI; Example 14. Unique Identifiers: A road net consists of a set of hubs and a set of links. Hubs and links have unique identifiers. That is: type HI, LI; value uid_H: H \rightarrow HI, uid_L: L \rightarrow LI;

2.13 Mereology

By **'mereology'** [13] we shall understand the study, knowledge and practice of parts, their relations to other parts and "the whole" \bullet

Part relations are such as: two or more parts being connected, one part being embedded within another part, and two or more parts sharing (other) attributes. Example 15. Mereology: The mereology of a link of a road net is the set of the two unique identifiers of exactly two hubs to which the link is connected. The mereology of a hub of a road net is the set of zero or more unique identifiers of the links to which the hub is connected. The domain analysis prompt: has mereology (p) holds if the part p is related to some others parts (p_a, p_b, \ldots, p_c) . The domain description prompt: observe_mereology(p) can then be invoked and yields some *formal description text* according to the followig *schema*: type MT $= \mathcal{E}(\mathsf{Pl}_A, \mathsf{Pl}_B, ..., \mathsf{Pl}_C)$; value mereo_P: $\mathsf{P} \to \mathsf{MT}$; where $\mathcal{E}(...)$ is some type expression over unique identifier types of one or more part sorts. Mereologies are expressed in terms of structures of unique part identifiers. Usually mereologies are constrained. Constraints express that a mereology's unique part identifiers must indeed reference existing parts, but also that these mereology identifiers "define" a proper structuring of parts. **Example 16.** Mereology Constraints: We continue our line of examples of road net endurants, cf. Example 2.4.1 on Page 6 but now a bit more systematically: A road net, n:N, contains a pair, (HS,LS), of sets Hs of hubs h:H and sets Ls of links. The mereology of links must identify exactly two hubs of the road net, the mereology of hubs must identify links of the road net, so connected hubs and links must have commensurate mereologies Two parts, $p_i:P_i$ and $p_j:P_j$, of possibly the same sort (i.e., $P_i \equiv P_j$) are said to **'refer one to another'** if the mereology of p_i contains the unique identifier of p_j and vice-versa • The parts p_i and p_j are then said to enjoy 'part **overlap'** • We refer to the concept of shared attributes covered at the very end of this section.

© Dines Bjørner 2013, DTU Informatics, Techn.Univ.of Denmark – May 30, 2013: 19:35 8Domain Analysis: Endurants — An Analysis & Description Process Model

¹¹That is, **has_unique_identifier(**p**)** for all parts p.

Science & Engineering

2.14**Attributes**

Attributes are what really endows parts with qualities. The external properties¹² are far from enough to distinguish one sort of parts from another. Similarly with unique identifiers and the mereology of parts. We therefore assume, without loss of generality, that every part, whether discrete or continuous, whether, when discrete, atomic or composite, has at least one attributes.

By a 'part attribute', or just an 'attribute', we shall understand a property that is associated with a part p of sort P, and if removed from part p, that part would no longer be part p but may be a part of some other sort P'; and where that property itself has no physical extent (i.e., volume), as the part may have, but may be measurable by physical means • Example 17. Attributes: Some attributes of road net hubs are location, hub state¹³, hub state space¹⁴, and of road net links are location, length, link state¹⁵, link state space¹⁶, etcetera The *domain description prompt* observe_attributes(*p*) yields some *formal description text* according to the followig schema: type A_1 , A_2 , ..., A_n , ATTR; value attr_ A_1 : $P \rightarrow A_1$, attr_ A_2 : $P \rightarrow A_2$, ..., attr_A_n:P \rightarrow A_n, attr_ATTR:P \rightarrow ATTR; where for \forall p:P, attr_A_i(attr_ATTR(p)) \equiv attr_A_i(p).

Shared Attributes 2.14.1

A final quality of endurant entities is that they may share attributes. Two parts, $p_i:P_i, p_j:P_j$, of different sorts are said to enjoy 'shared attributes' if P_i and P_j have at least one attribute name in common \bullet In such cases the mereologies of p_i and p_j are expected to refer to one another, i.e., be 'commensurable'.

2.15Discussion

We have left out any coverage of perdurant entities. For the time we refer to Sect. 5 of [7], hoping to further develop that section's understanding into a forthcoming study [8].

3 A Prompt & Description 'Method'

3.1**A Summary of Prompts**

In the previous section we outlined two classes of prompts: the *domain analysis prompts*:

attribute_ names [XII], 16 has_ concrete_ type [XI], 10 has_ materials [XIV], 25 has_ mereology [XIII], 22 is_ atomic [VIII], 7 is_ composite [IX], 7 is_ continuous [V], 6

is_ discrete [IV], 6 is_ endurant [II], 6 is_ entity [I], 5 is_ material [VII], 7 is_ part [VI], 7 is_ perdurant [III], 6 observe_ parts [X], 8

and the *domain description prompts*:

Domain Analysis: Endurants — Domain Analysis: Endurants

¹²is_discrete, is_continuous, is_atomic, is_composite and has_material.

¹³Hub state: a set of pairs of unique identifiers of actually connected links.

 $^{^{14}\}mathrm{Hub}$ state space: a set of hub states that a hub states may range over.

 $^{^{15}\}mathrm{Link}$ state: a set of pairs of unique identifiers of actually connected hubs.

¹⁶Link state space: a set of link states that a link state may range over.

observe_ attributes [4], 16 observe_ material_ sorts [6], 25 observe_ mereology [5], 22 observe_ part_ sorts [1], 8 observe_ part_ type [2], 10 observe_ unique_ identifier [3], 14

These prompts are imposed upon the domain analyser cum describer. They are "figuratively" applied to the domain. Their orderly, sequenced application follows the method hinted at in the previous section and expressed in a pseudo-formal notation in this section. The notation looks formal but since we have not formalised these prompts it is only pseudo-formal. In [5] we shall formalise these prompts.

3.2 **Preliminaries**

Let P be a sort, that is, a collection of endurants. By ηP we shall understand a syntactic quantity: the name of P. By ι_{P} :P we shall understand the semantic quantity: an (arbitrarily selected) endurant in P. And by $\eta^{-1}\eta P$ we shall understand P. To guide our analysis & description process we decompose it into steps. Each step "handles" a sort p:P or a material m:M. Steps handling discovery of composite sorts generates a set of sort names ηP_1 , ηP_2 , ..., ηP_n and ηM_1 , ηM_2 , ..., ηM_n . These are put in a reservoir for sorts to be inspected. The handled sort ηP or ηM is removed from that reservoir. Handling of material sorts concerns only their attributes. Each domain description prompt results in domain specification text (here we show only the formal texts) being deposited in the domain description reservoir, a global variable τ . The clause: domain_description_prompt(p): $\tau := \tau \oplus [$ "text;"] means that the formal text "text;" is joined to the global variable τ where that "text;" is prompted by domain_description_prompt(p). The meaning of \oplus will be discussed at the end of this section.

3.3 Initialising the Domain Analysis & Description Process

We remind the reader that we are dealing only with endurant domain entities. The domain analysis approach covered in Sect. 2 was based on decomposing an understanding of a domain from the "overall domain" into its components, and these, if not atomic, into their subcomponents. So we need to initialise the domain analysis & description by selecting (or choosing) the domain Δ .

Here is how we think of that "initialisation" process. The domain analyser[s] & describer[s] spends some time focusing on the domain, maybe at the "white board"¹⁷, rambling, perhaps in an un-structured manner, across its domain, Δ , and its sub domains. Informally jotting down more-or-less final sort names, building, in the domain analysers' & describers' mind an image of that domain. After some time doing this the domain analyser[s] & describer[s] is/are ready. An image of the domain in the form of "a domain" endurant, $\delta:\Delta$. Those are the quantities, $\eta\Delta$ (name of Δ) [Item 1] and $\iota p:P$ (for $(\delta:\Delta)$) [Item 8 on the following page], referred to below.

Thus this initialisation process is truly a creative one.

3.4 A Domain Analysis & Description State

1. A global variable αps will accumulate all the sort names being discovered.

© Dines Bjørner 2013, DTU Informatics, Techn.Univ.of Denmark – May 30, 2013: 19:35 10Domain Analysis: Endurants — An Analysis & Description Process Model

10

 $^{^{17}{\}rm Here}$ 'white board' is a conceptual notion. It could be physical, it could be yellow "post-it" stickers, or it could be an electronic conference "gadget".

Science & Engineering

- 2. A global variable νps will hold names of sorts yet to be analysed and described.
- 3. A global variable τ will hold the (so far) generated (in this case only) formal domain description text.

variable

- 1. $\alpha ps := [\eta \Delta] \eta P$ -set or ηP^*
- 2. $\nu ps := [\eta \Delta] (\eta P | \eta M)$ -set or $(\eta P | \eta M)^*$
- 3. $\tau := []$ Text-set or Text^{*}

We shall explain the use of [...]s and the operations of \setminus and \oplus on the above variables in Sect. 3.6.3 on Page 19.

3.5 Analysis & Description of Endurants

- 4. To analyse and describe endurants means to first
- 5. examine those endurant which have yet to be so analysed and described
- 6. by selecting (and removing from νps) a yet unexamined sort (by name);
- 7. then analyse and describe an endurant entity ($\iota p:P$) of that sort this analysis, when applied to composite parts, leads to the insertion of zero¹⁸ or more sort names¹⁹;
- 8. then to analyse and describe the mereology of each part sort,
- 9. and finally to analyse and describe the attributes of each sort.

value

- 4. analyse_and_describe_endurants: $Unit \rightarrow Unit$
- 4. analyse_and_describe_endurants() \equiv
- 5. while \sim is_empty(ν ps) do
- 6. **let** $\eta S = select_and_remove_\eta S()$ in
- 7. analyse_and_describe_endurant_sort(ι s:S) end end ;
- 8. for all $\eta P \cdot \eta P \in \alpha ps$ do analyse_and_describe_mereology($\iota p:P$) end
- 9. for all $\eta P \cdot \eta P \in \alpha ps$ do analyse_and_describe_attributes($\iota p:P$) end

The ι of Items 7, 8 and 9 are crucial. The domain analyser is focused on sort S (and P) and is "directed" (by those items) to choose (select) an endurant ι s (ι p) of that sort. The ability of the domain analyser to find such an entity is a measure of that person's professional creativity.

As was indicated in Sect. 2, the mereology of a part may involve unique identifiers of any part sort, hence must be done after all such part sort unique identifiers have been identified. Similarly for attributes which also may involve unique identifiers Each iteration of analyse_and_describe_endurant_sort($\iota p:P$) involves the selection of a sort (by name) (which is that of either a part sort or a material sort) with this sort name then being removed.

 $^{^{18}}$ If the sub-parts of p are all either atomic or already analysed, then no new sort names are added to the repository $\nu \mathsf{ps}).$

¹⁹These new sort names are then "picked-up" for sort analysis &c. in a next iteration of the while loop.

Domain Analysis: Endurants

- 10. The selection occurs from the global state (hence: ()) and changes that (hence **Unit**).
- 11. The affected global state component is that of the reservoir, νps .

value

- 10. select_and_remove_ $\eta S: Unit \rightarrow \eta P$
- 10. select_and_remove_ $\eta S() \equiv$
- 11. let $\eta S \bullet \eta S \in \nu ps$ in $\nu ps := \nu ps \setminus {\eta S} ; \eta S$ end

The analysis and description of all sorts also performs an analysis and description of their possible unique identifiers (if part sorts) and attributes. The analysis and description of sort mereologies potentially requires the unique identifiers of any set of sorts. Therefore the analysis and description of sort mereologies follows that of analysis and description of all sorts.

- 12. To analyse and describe an endurant
- 13. is to find out whether it is a part.
- 14. If so then it is to analyse and describe it as a part,
- 15. else it is to analyse and describe it as a material.
- 12. analyse_and_describe_endurant_sort: $(P|M) \rightarrow Unit$
- 12. analyse_and_describe_endurant_sort(e:(P|M)) \equiv

```
13. if is_part(e)
```

- 13. **assert:** $is_part(e) \equiv is_endurant(e) \land is_discrete(e)$
- 14. **then** analyse_and_describe_part_sort(e:P)
- 15. **else** analyse_and_describe_material_parts(e:M)
- 12. end

3.5.1 Analysis & Description of Part Sorts

- 16. The analysis and description of a part sort
- 17. is based on there being a set, ps, of parts²⁰ to analyse —
- 18. of which an archetypal one, p', is arbitrarily selected.
- 19. analyse and describe part p'
- 16. analyse_and_describe_part_sort: $P \rightarrow Unit$
- 16. analyse_and_describe_part_sort(p:P) \equiv
- 17. **let** $ps = observe_parts(p)$ in
- 18. let $p': P \bullet p' \in ps$ in
- 19. $analyse_and_describe_part(p')$
- 16. **end end**

© Dines Bjørner 2013, DTU Informatics, Techn.Univ.of Denmark – May 30, 2013: 19:35 12Domain Analysis: Endurants — An Analysis & Description Process Model

²⁰We can assume that there is at least one element of that set. For the case that the sort being analysed is a domain Δ , say "The Transport Domain", \mathbf{p}' is some representative "transport domain" δ . Similarly for any other sort for which \mathbf{ps} is now one of the sorts of δ .

21. first analyses and describes its unique identifiers.

Science & Engineering

22. If atomic

23. and

20. The analysis (&c.) of a part

24. if the part embodies materials,

```
25. we analyse and describe these.
 26. If not atomic then the part is composite
 27. and is analysed and described as such.
20.
      analyse_and_describe_part: P \rightarrow Unit
      analyse_and_describe_part(p) \equiv
20.
21.
        analyse_and_describe_unique_identifier(p);
22.
        if is_atomic(p)
          then
23.
24.
               if has_materials(p)
25.
                  then analyse_and_describe_part_materials(p) end
26.
          else assert: is_composite(p)
27.
               analyse_and_describe_composite_endurant(p) end
        pre: is_discrete(p)
20.
```

We do not associate materials with composite parts.

3.5.2 Analysis & Description of Part Materials

- 28. The analysis and description of the material part sorts, one or more, of atomic parts **p** of sort **P** containing such materials,
- 29. simply observes the material sorts of p,
- 30. that is generates the one or more continuous endurants
- 31. and the corresponding observer function text.
- 32. The reservoir of sorts to be inspected is augmented by the material sorts except if already previously entered (the $\setminus \alpha ps$ clause).
- 28. analyse_and_describe_part_materials: $P \rightarrow Unit$
- 28. analyse_and_describe_part_materials(p) \equiv
- 29. **observe_material_sorts**(p) :
- 30. $\tau := \tau \oplus [$ "type $M_1, M_2, \dots, M_m;$
- 31. value obs_M₁:P \rightarrow M₁,obs_M₂:P \rightarrow M₂,...,obs_M_m:P \rightarrow M_m;"
- 32. $\nu \mathsf{ps} := \nu \mathsf{ps} \oplus ([M_1, M_2, ..., M_m] \setminus \alpha \mathsf{ps})$
- 28. **pre**: has_materials(p)

Domain Analysis: Endurants

3.5.3 Analysis & Description of Material Parts

33. To analyse and describe materials, m, i.e., continuous endurants,

- 34. is only necessary if m has parts.
- 35. Then we observe the sorts of these parts.
- 36. The identified part sort names update both name reservoirs.

```
analyse_and_describe_material_parts: M \rightarrow Unit
33.
33.
       analyse_and_describe_material_parts(m:M) \equiv
34.
          if has_parts(m)
35.
             then observe_part_sorts(m):
                        \tau := \tau \oplus [" type P1,P2,...,PN ;
35.
35.
                                           value obs_Pi: M \rightarrow Pi i: \{1..N\};"]
36.
                \parallel \nu \mathbf{ps} := \nu \mathbf{ps} \oplus ([\eta P1, \eta P2, ..., \eta PN] \setminus \alpha \mathbf{ps})
                \parallel \alpha ps := \alpha ps \oplus [\eta P1, \eta P2, ..., \eta PN]
36.
33.
             end
          assert: is_continuous(m)
33.
```

3.5.4 Analysis & Description of Composite Endurants

- 37. To analyse and describe a composite endurant of sort P
- 38. is to (we choose first) to analyse and describe the unique identifier of that composite endurant,
- 39. then to analyse and describe the sort. If the sort has a concrete type
- 40. then we analyse and describe that concrete sort type
- 41. else we analyse and describe the abstract sort.
- 37. analyse_and_describe_composite_endurant: $P \rightarrow Unit$
- 37. analyse_and_describe_composite_endurant(p) \equiv
- 38. analyse_and_describe_unique_identifier(p);
- 39. **if has_concrete_type**(p)
- 40. **then** analyse_and_describe_concrete_sort(p)
- 41. **else** analyse_and_describe_abstract_sort(p)
- 39. end

3.5.5 Analysis & Description of Concrete Sort Types

- 42. The concrete sort type being analysed and described is
- 43. either
- 44. expressible by some compound type expression

Science & Engineering

15

43. or is

45. expressible by some alternative type expression.

- 42. analyse_and_describe_concrete_sort: $P \rightarrow Unit$
- 42. analyse_and_describe_concrete_sort(p:P) \equiv
- 44. analyse_and_describe_concrete_compound_type(p)
- 43.

- 45. analyse_and_describe_concrete_alternative_type(p)
- 42. **pre**: **has_concrete_type**(p)
 - 46. The concrete compound sort type
 - 47. is expressible by some simple type expression, $T = \mathcal{E}(Q, R, ..., S)$ over either concrete types or existing or new sorts Q, R, ..., S.
 - 48. The emerging sort types are identified
 - 49. and assigned to both νps
 - 50. and αps .
- 44. analyse_and_describe_concrete_compound_type: $P \rightarrow Unit$

```
analyse_and_describe_concrete_compound_type(p:P) \equiv
44.
46.
            observe_part_type(p):
                \tau := \tau \oplus ["type Q,R,...,S, T = \mathcal{E}(Q,R,...,S);
46.
                                    value obs_T: P \rightarrow T;"];
46.
47.
            let \{P_a, P_b, \dots, P_c\} = \text{sorts_of}(\{Q, R, \dots, S\})
                        assert: \{P_a, P_b, \dots, P_c\} \subseteq \{Q, R, \dots, S\} in
48.
49.
            \nu \mathbf{ps} := \nu \mathbf{ps} \oplus [\eta \mathbf{P}_a, \eta \mathbf{P}_b, ..., \eta \mathbf{P}_c] \parallel
50.
            \alpha ps := \alpha ps \oplus ([\eta P_a, \eta P_b, ..., \eta P_c] \setminus \alpha ps) end
44.
            pre: has_concrete_type(p)
```

- 52. is expressible by an alternative type expression T=P1|P2|...|PN where each of the alternative types is made disjoint wrt. existing types by means of the description language Pi::mkPi(s_u:P_i) construction.
- 53. The emerging sort types are identified and assigned
- 54. to both νps

55. and αps .

- 45. analyse_and_describe_concrete_alternative_type: $P \rightarrow Unit$
- 45. analyse_and_describe_concrete_alternative_type(p:P) \equiv
- 51. **observe_part_type**(p):
- 52. $\tau := \tau \oplus [$ "type T=P1 | P2 | ... | PN, Pi::mkPi(s_u:P_i) (1 \le i \le N);

^{51.} The concrete alternative sort type expression

Domain Analysis: Endurants

```
52. value obs_T: P \rightarrow T;"];

53. let {P_a, P_b, ..., P_c} = sorts_of({P_i | 1 \le i \le n})

53. assert: {P_a, P_b, ..., P_c} \subseteq {P_i | 1 \le i \le n} in

54. \nu ps := \nu ps \oplus ([\eta P_a, \eta P_b, ..., \eta P_c] \setminus \alpha ps) \parallel

55. \alpha ps := \alpha ps \oplus [\eta P_a, \eta P_b, ..., \eta P_c] end

42. pre: has_concrete_type(p)
```

3.5.6 Analysis & Description of Abstract Sorts

- 56. To analyse and describe an abstract sort
- 57. amounts to observe part sorts and to
- 58. update the sort name repositories.

```
56. analyse_and_describe_abstract_sort: P \rightarrow Unit
```

```
56. analyse_and_describe_abstract_sort(p:P) \equiv
```

```
57. observe_part_sorts(p):
57. \tau := \tau \oplus ["type P<sub>1</sub>, P<sub>2</sub>, ..., P<sub>n</sub>;
```

```
57. value obs_P_i: P \rightarrow P_i \ (0 \le i \le n);"]
```

- 58. $\| \nu \mathbf{ps} := \nu \mathbf{ps} \oplus ([\eta \mathbf{P}_1, \eta \mathbf{P}_2, ..., \eta \mathbf{P}_n] \setminus \alpha \mathbf{ps})$
- 58. $\| \alpha \mathbf{ps} := \alpha \mathbf{ps} \oplus [\eta \mathbf{P}_1, \eta \mathbf{P}_2, ..., \eta \mathbf{P}_n]$

3.5.7 Analysis & Description of Unique Identifiers

59. To analyse and describe the unique identifier of parts of sort ${\sf P}$ is

- 60. to observe the unique identifier of parts of sort ${\sf P}$
- 61. where we assume that all parts have unique identifiers.
- 59. analyse_and_describe_unique_identifier: $P \rightarrow Unit$
- 59. analyse_and_describe_unique_identifier(p) \equiv

```
60. observe_unique_identifier(p):
```

- 60. $\tau := \tau \oplus [$ "type PI; value uid_P:P \rightarrow PI;"]
- 61. **assert:** has_unique_identifier(p)

3.5.8 Analysis & Description of Mereologies

- 62. To analyse and describe a part mereology
- 63. if it has one
- 64. amounts to observe that mereology
- 65. and otherwise do nothing.
- 66. The analysed quantity must be a part.

Science & Engineering

3.5.9 Analysis & Description of Part Attributes

```
67. To analyse and describe the attributes of parts of sort P is
```

- 68. to observe the attributes of parts of sort P
- 69. where we assume that all parts have attributes.

```
67. analyse_and_describe_part_attributes: P \rightarrow Unit
```

- 67. analyse_and_describe_part_attributes(p) \equiv
- 68. **observe_attributes**(p):
- 68. $\tau := \tau \oplus [$ "**type** A₁, A₂,..., A_m;
- 68. **value attr_** $A_1:P \rightarrow A_1$, **attr_** $A_2:P \rightarrow A_2$,..., **attr_** $A_m:P \rightarrow A_m;$ "]
- 69. **assert:** has_attributes(p)

3.6 Discussion of The Model

The above model lacks a formal understanding of the individual prompts as listed in Sect. 3.1; such an understanding is attempted in [5].

3.6.1 **Termination**

The sort name reservoir νps is "reduced" by one name in each iteration of the **while** loop of the analyse_and_describe_endurants, cf. Item 6 on Page 12, and is augmented by new part and material sort names in some iterations of that loop, cf. formula Items 32 on Page 14, 36 on Page 15, 49 on Page 16, 54 on Page 16 and 49 on Page 16. It remains to prove that the analysis & description process terminates.

3.6.2 Axioms and Proof Obligations

We have omitted from the above (and also in Sect. 2) treatment of axioms concerning wellformedness of parts, materials and attributes and proof obligations concerning disjointness of observed part and material sorts and attribute types. A more proper treatment would entail adding a line of proof obligation text right after Item lines 65 on the preceding page and 68. and of axiom text right after Item lines 31 (Page 14), 35 (Page 15), 46 (Page 16), 48 (Page 16), 60 (Page 17) and 68 (Page 18). No axiom is needed in connection with Item line 52 on Page 16.

[7] covers axioms and proof obligations in some detail.

Domain Analysis: Endurants

3.6.3 Order of Analysis & Description: A Meaning of '⊕'

The variables αps , νps and τ are defined to hold either sets or lists. The operator \oplus can be thought of as either set union $(\cup \text{ and } [,] \equiv \{,\})$ — in which case the domain description text in τ is a set of domain description texts or as list concatenation (^ and $[,] \equiv \langle,\rangle$) of domain description texts. The operator $\ell_1 \oplus \ell_2$ now has at least two interpretations: either $\ell_1 \cap \ell_2$ or $\ell_2 \cap \ell_1$. In the case of lists the \oplus (i.e., ^) does not (suffix or prefix) append ℓ_2 elements already in ℓ_1 . The select_and_remove_ ηP function on Page 13 applies to the set interpretation. A list interpretation is:

value

- 6. select_and_remove_ ηP : Unit $\rightarrow \eta P$
- 6. select_and_remove_ $\eta P() \equiv$
- 6. let $\eta P = hd \nu ps$ in $\nu ps := tl \nu ps; \eta P$ end

In the first case $(\ell_1 \ \ell_2)$ the analysis and description process proceeds from the root, breadth first, In the second case $(\ell_2 \ \ell_1)$ the analysis and description process proceeds from the root, depth first.

3.6.4 Laws of Description Prompts

The domain 'method' outlined in the previous section suggests that many different orders of analysis & description may be possible. But are they? That is, will they all result in "similar" descriptions? That is, if \mathcal{D}_a and \mathcal{D}_b are two domain description prompts where \mathcal{D}_a and \mathcal{D}_b can be pursued in any order will that yield the same description? And what do we mean by 'can be pursued in any order', and 'same description'? Let us assume that sort P decomposes into sorts P_a and P_b (etcetera). Let us assume that the domain description prompt \mathcal{D}_a is related to the description of P_a and \mathcal{D}_b to P_b . Here we would expect \mathcal{D}_a and \mathcal{D}_b to commute, that is $\mathcal{D}_a; \mathcal{D}_b$ yields same result as does $\mathcal{D}_b; \mathcal{D}_a$. In [4] we made an early exploration of such laws of domain description prompts.

To answer these questions we need a reasonably precise model of domain prompts. We attempt such a model in [5].

4 Conclusion

It is time to conclude.

4.1 Comparison to Other Work

4.1.1 **Domain Analysis**

In [7] we give, in its Sect. 6.3, a 3+ page comparison to the broader concept of 'domain analysis' as covered in almost 30 literature references. We claim, on this background, that our concept of domain analysis, as treated in Sect. 2, is sufficiently different (i.e., novel) as to warrant your attention!

© Dines Bjørner 2013, DTU Informatics, Techn.Univ.of Denmark – May 30, 2013: 19:35 18Domain Analysis: Endurants — An Analysis & Description Process Model

Science & Engineering

4.1.2Methodology

We are not aware of any publications (other other than [4]) that attempt for "formalise" core concepts of the notion of 'method'. See below.

4.2What Have We Achieved

4.2.1**Domain Analysis**

In Sect. 2 we have presented a terse, five+ page, summary of a novel approach to domain analysis. That this approach is different from other 'domain analysis' approaches is argued in [7, Sect. 6.2]. The new aspects are: the distinction between parts and materials, the distinction between external and internal properties (Sect. 2.11.1), the introduction of the concept of mereologies and the therefrom separate treatment of attributes. It seems to us that "conventional" domain analysis treated all endurant qualities as attributes. The many concepts, endurants and perdurants, discrete and continuous, hence parts and materials, atomic and composite, uniqueness of parts, mereology, and shared attributes, we claim, are forced upon the analysis by the nature of domains: existing in some not necessarily computable reality. In this way the proposed domain analysis & description approach is new.

4.2.2Methodology

By a 'method' we shall understand a set of principles for selecting and applying techniques and tools in order to analyse and construct an artifact. Clearly Sect. 3 presents a partially instantiated framework for a formal model of a 'method': Some principles are abstraction (sorts in preference for concrete types), separation of concerns (tackling endurants before perdurants), commensurate narratives and formalisations, tackling domain analysis either "top-down", hierarchically from composite endurants, or "bottom-up", compositionally, from atomic endurants, or in some orderly combination of these; Etcetera. Some techniques are expressing axioms concerning well-formedness of mereologies and attribute values; stating (and discharging) proof obligations securing disjointness of sorts; etcetera. And some tools are the *domain analysis prompt*s, the *domain description prompt*s and the description language (here RSL [12]). We claim that we have sketched a formalisation of a method for domain analysis and description.

What is really new here is, as for domain analysis, that the analysis & description process is applied to a domain, that is, to our image of that domain, something not necessarily computable, and that our description therefore must not reduce the described domain to a computable artefact.

4.3**Future Work**

There remains to conclude studies of, to document and publish treatments of the following related topics: (i) domain analysis of perdurants (actions, events and behaviours [7, Sect. 5]) — including related *domain analysis prompts* and *domain description prompts*²¹, (ii) model(s) of prompts²², (iii) domain facets, cf. $[3]^{23}$, and (iv) derivation of requirements from domain

Domain Analysis: Endurants — Domain Analysis: Endurants

²¹See forthcoming [8]

 $^{^{22}}$ See forthcoming [5]

 $^{^{23}}$ See forthcoming [6]

Domain Analysis: Endurants

descriptions, cf. $[2]^{24}.$.

4.4 Acknowledgements

5 **Bibliography**

5.1 **Bibliographical Notes**

The citations of this paper are seriously skewed. Concerning Sect. 2, A Summary of TripTych Domain Analysis Approach: the 30+ citations, given in [7], of papers relating to domain analysis have here been left out. Concerning Sect. 3, A Prompt and Description 'Method': and we could not find — and were therefore not influenced or inspired by — publications of formalised process models for domain analysis & description.

5.2 References

- D. Bjørner. Domain Theory: Practice and Theories, Discussion of Possible Research Topics. In ICTAC'2007, volume 4701 of Lecture Notes in Computer Science (eds. J.C.P. Woodcock et al.), pages 1–17, Heidelberg, September 2007. Springer.
- [2] D. Bjørner. From Domains to Requirements. In Montanari Festschrift, volume 5065 of Lecture Notes in Computer Science (eds. Pierpaolo Degano, Rocco De Nicola and José Meseguer), pages 1–30, Heidelberg, May 2008. Springer.
- [3] D. Bjørner. Domain Engineering. In P. Boca and J. Bowen, editors, *Formal Methods: State of the Art and New Directions*, Eds. Paul Boca and Jonathan Bowen, pages 1–42, London, UK, 2010. Springer.
- [4] D. Bjørner. Domain Science & Engineering From Computer Science to The Sciences of Informatics Part II of II: The Science Part. Kibernetika i sistemny analiz, (2):100–120, May 2011.
- [5] D. Bjørner. Domain Analysis: A Model of Prompts (paper²⁵, slides²⁶). Research Report 2013-6, DTU Compute, Summer 2013. A first draft of this document will be written over the summer of 2013.
- [6] D. Bjørner. Domain Analysis: Facets (paper²⁷, slides²⁸). Research Report 2013-7, DTU Compute, Summer 2013. A first draft of this document might be written late summer of 2013.
- [7] D. Bjørner. Domain analysis (paper²⁹ slides³⁰). Research Report 2013-1, DTU Compute, April 2013.

20

²⁴See forthcoming [10]

²⁵http://www.imm.dtu.dk/~dibj/da-mod-p.pdf

²⁶http://www.imm.dtu.dk/~dibj/da-mod-s.pdf

²⁷http://www.imm.dtu.dk/~dibj/da-facets-p.pdf

²⁸http://www.imm.dtu.dk/~dibj/da-facets-s.pdf

²⁹http://www.imm.dtu.dk/~dibj/da-p.pdf

 $^{^{30} \}rm http://www.imm.dtu.dk/~dibj/da-s.pdf$

[©] Dines Bjørner 2013, DTU Informatics, Techn.Univ.of Denmark – May 30, 2013: 19:35 20Domain Analysis: Endurants — An Analysis & Description Process Model

Science & Engineering

- [8] D. Bjørner. Domain Analysis: Perdurants (paper³¹, slides³²). Research Report 2013-5, DTU Compute, Spring 2013. A first draft of this document will be written over the summer of 2013.
- [9] D. Bjørner. Domain Science and Engineering as a Foundation for Computation for Humanity, chapter 7, pages 159–177. Computational Analysis, Synthesis, and Design of Dynamic Systems. CRC [Francis & Taylor], 2013. (eds.: Justyna Zander and Pieter J. Mosterman).
- [10] D. Bjørner. On Deriving Requirements from Domain Specifications (paper³³, slides³⁴). Research Report 2013-8, DTU Compute, Summer 2013. A first draft of this document might be written late summer of 2013.
- [11] B. Ganter and R. Wille. Formal Concept Analysis Mathematical Foundations. Springer-Verlag, January 1999. ISBN: 3540627715, 300 pages, Amazon price: US \$ 44.95.
- [12] C. W. George, P. Haff, K. Havelund, A. E. Haxthausen, R. Milne, C. B. Nielsen, S. Prehn, and K. R. Wagner. *The RAISE Specification Language*. The BCS Practitioner Series. Prentice-Hall, Hemel Hampstead, England, 1992.
- [13] E. Luschei. The Logical Systems of Leśniewksi. North Holland, Amsterdam, The Netherlands, 1962.

A Pipeline Endurants

Our example is an abstraction of pipeline system endurants. The presentation of the example reflects a rigorous use of the domain analysis & description method outlined in Sect. 2, but is relaxed with respect to not showing all — one could say — intermediate analysis steps and description texts, but following stoichiometry ideas from chemistry makes a few short-cuts here and there. The use of the "stoichiometrical" reductions, usually skipping intermediate endurant sorts, ought properly be justified in each step — and such is adviced in proper, industry-scale analyses & descriptions.

To guide your intuition with respect to what a pipeline system might be we suggest some diagrams and some pictures. See Figs. 1 on the next page and 2 on the facing page.

The description only covers a few aspects of endurants.

A.1 Parts

- 70. A pipeline system contains a set of pipeline units and a pipeline system monitor.
- 71. The well-formedness of a pipeline system depends on its mereology (cf. Sect. A.2.3) and the routing of its pipes (cf. Sect. A.3.2).
- 72. A pipeline unit is either a well, a pipe, a pump, a valve, a fork, a join, or a sink unit.
- 73. We consider all these units to be distinguishable, i.e., the set of wells, the set pipe, etc., the set of sinks, to be disjoint.

³¹http://www.imm.dtu.dk/~dibj/da-perd-p.pdf

³²http://www.imm.dtu.dk/~dibj/da-perd-s.pdf

 $^{^{33}} http://www.imm.dtu.dk/~dibj/da-fac-p.pdf$

³⁴http://www.imm.dtu.dk/~dibj/da-fac-s.pdf

Domain Analysis: Endurants



Figure 1: Oil or gas pipelines



Figure 2: Some oil pipeline system units: pump, pipe, valve

\mathbf{type}

- 70. PLS', U, M
- 71. $PLS = \{ | pls: PLS' \bullet wf_PLS(pls) | \}$

value

- 71. wf_PLS: $PLS \rightarrow Bool$
- 71. wf_PLS(pls) \equiv wf_Mereology(pls) \land wf_Routes(pls)
- 70. $obs_Us: PLS \rightarrow U$ -set
- 70. $obs_M: PLS \to M$

type

- 72. U = We | Pi | Pu | Va | Fo | Jo | Si
- 73. We :: Well
- 73. Pi :: Pipe
- 73. Va :: Valv
- 73. Fo :: Fork
- 73. Jo :: Join
- 73. Si :: Sink

Science & Engineering

A.2 Part Identification and Mereology

A.2.1 Unique Identification

74. Each pipeline unit is uniquely distinguished by its unique unit identifier.

```
type

74. UI

value

74. uid_UI: U \rightarrow UI

axiom

74. \forall pls:PLS,u,u':U•{u,u'}⊆obs_Us(pls)\Rightarrowu\nequ'\Rightarrowuid_UI(u)\nequid_UI(u')
```

A.2.2 Unique Identifiers

75. From a pipeline system one can observe the set of all unique unit identifiers.

value

- 75. xtr_UIs: $PLS \rightarrow UI$ -set
- 75. $xtr_UIs(pls) \equiv {uid_UI(u)|u:U \bullet u \in obs_Us(pls)}$
 - 76. We can prove that the number of unique unit identifiers of a pipeline system equals that of the units of that system.

theorem:

```
76. \forall pls:PLS•card obs_Us(pl)=card xtr_UIs(pls)
```

A.2.3 Mereology

- 77. Each unit is connected to zero, one or two other existing input units and zero, one or two other existing output units as follows:
 - a. A well unit is connected to exactly one output unit (and, hence, has no "input").
 - b. A pipe unit is connected to exactly one input unit and one output unit.
 - c. A pump unit is connected to exactly one input unit and one output unit.
 - d. A valve is connected to exactly one input unit and one output unit.
 - e. A fork is connected to exactly one input unit and two distinct output units.
 - f. A join is connected to exactly two distinct input units and one output unit.
 - g. A sink is connected to exactly one input unit (and, hence, has no "output").

```
type
```

77. MER = UI-set \times UI-set

Domain Analysis: Endurants — Domain Analysis: Endurants

- value
- 77. mereo_U: $U \rightarrow MER$ axiom

Domain Analysis: Endurants

77.	wf_Mereology: $PLS \rightarrow Bool$
77.	wf_Mereology(pls) \equiv
77.	$\forall u:U \bullet u \in obs_Us(pls) \Rightarrow$
77.	let (iuis,ouis) = mereo_U(u) in iuis \cup ouis \subseteq xtr_UIs(pls) \land
77.	case (u, (card uius, card ouis)) of
77a	$(mk_We(we),(0,1)) \rightarrow \mathbf{true},$
77b	$(mk_Pi(pi),(1,1)) \rightarrow \mathbf{true},$
77c	$(\mathrm{mk_Pu}(\mathrm{pu}),(1,1)) \rightarrow \mathbf{true},$
77d	$(mk_Va(va),(1,1)) \rightarrow \mathbf{true},$
77e	$(\mathrm{mk_Fo}(\mathrm{fo}),(1,1)) \rightarrow \mathbf{true},$
77f	$(mk_Jo(jo),(1,1)) \rightarrow true,$
77g	$(mk_Si(si),(1,1)) \rightarrow true,$
77.	$_ \rightarrow$ false end end

A.3 Part Concepts

An aspect of domain analysis & description that was not covered in Sect. 2 was that of derived concepts. Example pipeline concepts are routes, acyclic or cyclic, circular, etcetera. In expressing well-formedness of pipeline systems one often has to develop subsidiary concepts such as these by means of which well-formedness is then expressed.

A.3.1 Pipe Routes

78. A route (of a pipeline system) is a sequence of connected units (of the pipeline system).

79. A route descriptor is a sequence of unit identifiers and the connected units of a route (of a pipeline system).

```
type

78. \mathbf{R}' = \mathbf{U}^{\omega}

78. \mathbf{R} = \{ | \mathbf{r}: \mathbf{Route'} \cdot \mathbf{wf}_{\mathbf{R}} \mathbf{Route}(\mathbf{r}) | \}

79. \mathbf{RD} = \mathbf{UI}^{\omega}

axiom

79. \forall \mathbf{rd}: \mathbf{RD} \cdot \exists \mathbf{r}: \mathbf{R} \cdot \mathbf{rd} = \mathrm{descriptor}(\mathbf{r})

value

79. \mathrm{descriptor}: \mathbf{R} \to \mathbf{RD}

79. \mathrm{descriptor}(\mathbf{r}) \equiv \langle \mathrm{uid}_{\mathbf{UI}}(\mathbf{r}[\mathbf{i}]) | \mathbf{i}: \mathbf{Nat} \cdot \mathbf{1} \leq \mathbf{i} \leq \mathbf{len} \mathbf{r} \rangle
```

80. Two units are adjacent if the output unit identifiers of one shares a unique unit identifier with the input identifiers of the other.

value

```
80. adjacent: U \times U \rightarrow \mathbf{Bool}

80. adjacent(u,u') \equiv

80. let (,ouis)=mereo_U(u),(iuis,)=mereo_U(u') in

80. ouis \cap iuis \neq {} end
```

Science & Engineering

- 81. Given a pipeline system, pls, one can identify the (possibly infinite) set of (possibly infinite) routes of that pipeline system.
 - a. The empty sequence, $\langle \rangle$, is a route of *pls*.
 - b. Let u, u' be any units of *pls*, such that an output unit identifier of u is the same as an input unit identifier of u' then $\langle u, u' \rangle$ is a route of *pls*.
 - c. If r and r' are routes of pls such that the last element of r is the same as the first element of r', then $r^{t}r'$ is a route of pls.
 - d. No sequence of units is a route unless it follows from a finite (or an infinite) number of applications of the basis and induction clauses of Items 81a.–81c..

value

A.3.2 Well-formed Routes

82. A route is acyclic if no two route positions reveal the same unique unit identifier.

value

- 82. acyclic_Route: $R \rightarrow Bool$
- 82. $acyclic_Route(r) \equiv \sim \exists i,j: Nat \{i,j\} \subseteq inds r \land i \neq j \land r[i] = r[j]$
- 83. A pipeline system is well-formed if none of its routes are circular (and all of its routes embedded in well-to-sink routes).

value

- 83. wf_Routes: $PLS \rightarrow Bool$
- 83. wf_Routes(pls) \equiv
- 83. non_circular(pls) \land are_embedded_in_well_to_sink_Routes(pls)
- 83. non_circular_PLS: $PLS \rightarrow Bool$
- 83. non_circular_PLS(pls) \equiv
- 83. $\forall r: \mathbb{R} \bullet r \in routes(p) \land acyclic_Route(r)$
- 84. We define well-formedness in terms of well-to-sink routes, i.e., routes which start with a well unit and end with a sink unit.

Domain Analysis: Endurants

value

- 84. well_to_sink_Routes: $PLS \rightarrow R$ -set
- 84. well_to_sink_Routes(pls) \equiv
- 84. **let** rs = Routes(pls) in
- 84. $\{r|r:R\bullet r \in rs \land is_We(r[1]) \land is_Si(r[len r])\}$ end

85. A pipeline system is well-formed if all of its routes are embedded in well-to-sink routes.

```
85. are_embedded_in_well_to_sink_Routes: PLS \rightarrow Bool
```

- 85. are_embedded_in_well_to_sink_Routes(pls) \equiv
- 85. **let** wsrs = well_to_sink_Routes(pls) **in**
- 85. \forall r:R r \in Routes(pls) \Rightarrow
- 85. $\exists r':R,i,j:Nat \bullet$
- 85. $r' \in wsrs$
- 85. $\wedge \{i,j\} \subseteq \mathbf{inds} r' \land i \leq j$
- 85. $\wedge \mathbf{r} = \langle \mathbf{r}'[\mathbf{k}] | \mathbf{k} : \mathbf{Nat} \cdot \mathbf{i} \leq \mathbf{k} \leq \mathbf{j} \rangle$ end

A.3.3 Embedded Routes

86. For every route we can define the set of all its embedded routes.

value

- 86. embedded_Routes: $R \rightarrow R$ -set
- 86. embedded_Routes(r) \equiv
- 86. $\{\langle \mathbf{r}[\mathbf{k}] | \mathbf{k}: \mathbf{Nat} \bullet \mathbf{i} \leq \mathbf{k} \leq \mathbf{j} \rangle \mid \mathbf{i}, \mathbf{j}: \mathbf{Nat} \bullet \mathbf{i} \{\mathbf{i}, \mathbf{j}\} \subseteq \mathbf{inds}(\mathbf{r}) \land \mathbf{i} \leq \mathbf{j} \}$

A.3.4 A Theorem

87. The following theorem is conjectured:

- a. the set of all routes (of the pipeline system)
- b. is the set of all well-to-sink routes (of a pipeline system) and
- c. all their embedded routes

theorem:

```
87. \forall pls:PLS •

87. let rs = Routes(pls),

87. wsrs = well_to_sink_Routes(pls) in

87a.. rs =

87b.. wsrs \cup

87c.. \cup \{\{r'|r': \mathbb{R} \cdot r' \in \text{embedded}_\text{Routes}(r'')\} \mid r'': \mathbb{R} \cdot r'' \in \text{wsrs}\}

86. end
```

26

Science & Engineering

A.4 Materials

88. The only material of concern to pipelines is the gas³⁵ or liquid³⁶ which the pipes transport³⁷.

type

- 88. GoL
 - value
- 88. $obs_GoL: U \rightarrow GoL$

A.5 Attributes

A.5.1 Part Attributes

- 89. These are some attribute types:
 - a. estimated current well capacity (barrels of oil, etc.),
 - b. pipe length,
 - c. current pump height,
 - d. current valve open/close status and
 - e. flow (e.g., volume/second).

type

89a	WellCap
89b	LEN
89c	Height
89d	ValSta == open close
89e	Flow

- 90. Flows can be added (also distributively) and subtracted, and
- 91. flows can be compared.

value

- 90. \oplus, \ominus : Flow×Flow \rightarrow Flow
- 90. \oplus : Flow-set \rightarrow Flow
- 91. $<,\leq,=,\neq,\geq,>$: Flow \times Flow \rightarrow **Bool**
 - 92. Properties of pipeline units include
 - a. estimated current well capacity (barrels of oil, etc.),
 - b. pipe length,

 $^{^{35}\}mathrm{Gaseous}$ materials include: air, gas, etc.

 $^{^{36}\}mathrm{Liquid}$ materials include water, oil, etc.

 $^{^{37}\}mathrm{The}$ description of this document is relevant only to gas or oil pipelines.

Domain Analysis: Endurants

- c. current pump height,
- d. current valve open/close status,
- e. current \mathcal{L} aminar in-flow at unit input,
- f. current \mathcal{L} aminar in-flow leak at unit input,
- g. maximum \mathcal{L} aminar guaranteed in-flow leak at unit input,
- h. current Laminar leak unit interior,
- i. current \mathcal{L} aminar flow in unit interior,
- j. maximum \mathcal{L} aminar guaranteed flow in unit interior,
- k. current \mathcal{L} aminar out-flow at unit output,
- l. current \mathcal{L} aminar out-flow leak at unit output,
- m. maximum guaranteed Laminar out-flow leak at unit output.

value

- 92a.. attr_WellCap: We \rightarrow WellCap
- 92b.. attr_LEN: $Pi \rightarrow LEN$
- 92c.. attr_Height: $Pu \rightarrow Height$
- 92d.. attr_ValSta: Va \rightarrow VaSta
- 92e.. attr_In_Flow \mathcal{L} : U \rightarrow UI \rightarrow Flow
- 92f.. attr_In_Leak_ \mathcal{L} : U \rightarrow UI \rightarrow Flow
- 92g.. attr_Max_In_Leak_ \mathcal{L} : U \rightarrow UI \rightarrow Flow
- 92h.. attr_body_ $Flow_{\mathcal{L}}$: U \rightarrow Flow
- 92i.. attr_body_Leak_{\mathcal{L}}: U \rightarrow Flow
- 92j.. $\operatorname{attr}_{\operatorname{Max}}\operatorname{Flow}_{\mathcal{L}}: U \to \operatorname{Flow}_{\mathcal{L}}$
- 92k.. $\operatorname{attr_Out_Flow}_{\mathcal{L}}: U \to UI \to Flow$
- 921.. attr_Out_Leak_{\mathcal{L}}: U \rightarrow UI \rightarrow Flow
- 92m.. attr_Max_Out_Leak_ \mathcal{L} : U \rightarrow UI \rightarrow Flow

A.5.2 Flow Laws, I

93. "What flows in, flows out !". For Laminar flows: for any non-well and non-sink unit the sums of input leaks and in-flows equals the sums of unit and output leaks and out-flows.

Law:

- 93. \forall u:U\We\Si •
- 93. $sum_in_leaks(u) \oplus sum_in_flows(u) =$
- 93. attr_body_Leak $_{\mathcal{L}}(\mathbf{u}) \oplus$
- 93. $sum_out_leaks(u) \oplus sum_out_flows(u)$

value

 $\begin{array}{l} \text{sum_in_leaks: } U \to Flow \\ \text{sum_in_leaks}(u) \equiv \\ & \quad \textbf{let (iuis,)} = \text{mereo_U}(u) \textbf{ in} \\ \oplus \{ attr_In_Leak_{\mathcal{L}}(u)(ui) | ui: UI \bullet ui \in iuis \} \textbf{ end} \end{array}$

Science & Engineering

```
sum_in_flows: U \rightarrow Flow
sum_in_flows(u) \equiv
       let (iuis,) = mereo_U(u) in
       \oplus {attr_In_Flow<sub>L</sub>(u)(ui)|ui:UI•ui \in iuis} end
sum_out_leaks: U \rightarrow Flow
sum_out\_leaks(u) \equiv
       let (,ouis) = mereo_U(u) in
       \oplus  {attr_Out_Leak<sub>L</sub>(u)(ui)|ui:UI•ui \in ouis} end
sum_out_flows: U \rightarrow Flow
sum_out_flows(u) \equiv
       let (,ouis) = mereo_U(u) in
       \oplus  {attr_Out_Leak<sub>L</sub>(u)(ui)|ui:UI•ui \in ouis} end
```

94. "What flows out, flows in !". For \mathcal{L} aminar flows: for any adjacent pairs of units the output flow at one unit connection equals the sum of adjacent unit leak and in-flow at that connection.

Law:

- 94. \forall u,u':U•adjacent(u,u') \Rightarrow
- let (,ouis)=mereo_U(u), (iuis',)=mereo_U(u') in 94.
- 94. assert: uid_U(u') \in ouis \land uid_U(u) \in iuis '
- 94. attr_Out_Flow_{\mathcal{L}}(u)(uid_U(u')) =
- attr_In_Leak_{\mathcal{L}}(u)(uid_U(u)) \oplus attr_In_Flow_{\mathcal{L}}(u')(uid_U(u)) end 94.

Β Indexes

B.1 Index of Definitions

abstract	autonomous, 18
type, 8	biddable, 18
action	continuous, 17
discrete, 38	discrete, 17
active	dynamic, 17
attribute, 18	inert, 17
actor, 37	programmable, 18
analysis	reactive, 18
domain	shared, 19
prompt, 3	static, 17
prompt	autonomous
domain, 3	attribute, 18
Atomic	
part, 7	behaviour
attribute	continuous, 40
active, 18	discrete, 38

Domain Analysis: Endurants — Domain Analysis: Endurants

30

Domain Analysis: Endurants

biddable attribute, 18 Composite part, 7 concept formal, 5 concrete type, 8 confusion, 28 context formal, 4 continuous attribute, 17 behaviour, 40 endurant, 6 derived, 11 description domain, 2 prompt, 3, 22 prompt domain, 3, 22 discrete action, 38 attribute, 17 behaviour, 38 endurant, 6 domain, 2 analysis prompt, 3 description, 2 prompt, 3, 22 facet, 49prompt analysis, 3 description, 3, 22 dynamic attribute, 17 endurant, 6 continuous, 6 discrete, 6 external quality, 13 internal quality, 13 quality

external, 13 internal, 13 entity, 5 event, 38 expression function type, 40 type function, 40 extent, 5 external endurant quality, 13 quality endurant, 13 facet domain, 49 human behaviour, 50 intrinsics, 49 organisation & management, 49 script, 50 support technology, 49 support technology behaviour, 50 formal concept, 5 context, 4 function expression type, 40 signature, 40 type expression, 40 has_ concrete_ type prerequisite prompt, 10 prompt prerequisite, 10 has_ mereology prerequisite prompt, 23 prompt prerequisite, 23 has_ unique_ identifier prerequisite prompt, 14

Science & Engineering

prompt prerequisite, 14 head pump, 40 human behaviour facet, 50 inert attribute, 17 intent, 5 internal endurant quality, 13 quality endurant, 13 intrinsics, 49 facet, 49 is_ composite prerequisite prompt, 8 prompt prerequisite, 8 is_ discrete prerequisite prompt, 7 prompt prerequisite, 7 is_ entity prerequisite prompt, 5, 6 prompt prerequisite, 5, 6 junk, 28 material, 6, 7, 25 mereology, 21 type, 22 observe_part_type prerequisite prompt, 10 prompt prerequisite, 10 ontological engineering, 46 organisation & management facet, 49

part, 6, 7 Atomic, 7 Composite, 7 qualities, 13 perdurant, 6, 37 prerequisite has_ concrete_ type prompt, 10 has_mereology prompt, 23 has_ unique_ identifier prompt, 14 is_ composite prompt, 8 is_ discrete prompt, 7 is_ entity prompt, 5, 6 observe_part_type prompt, 10 prompt has_ concrete_ type, 10 has_ mereology, 23 has_ unique_ identifier, 14 is_ composite, 8 is_ discrete, 7 is_ entity, 5, 6observe_part_type, 10 programmable attribute, 18 prompt analysis domain, 3 description domain, 3, 22 domain analysis, 3 description, 3, 22 has_concrete_type prerequisite, 10 has_ mereology prerequisite, 23 has_ unique_ identifier prerequisite, 14 is_ composite prerequisite, 8 is_ discrete

Domain Analysis: Endurants

prerequisite, 7 is_ entity prerequisite, 5, 6 observe_part_type prerequisite, 10 prerequisite has_ concrete_ type, 10 has_ mereology, 23 has_unique_identifier, 14 is_ composite, 8 is_ discrete, 7 is_ entity, 5, 6observe_part_type, 10 pump head, 40 qualities part, 13 semantic, 13 syntactic, 13 quality endurant external, 13 internal. 13 external endurant, 13 internal endurant, 13 reactive attribute, 18

B.2 Index of Concepts

abstract type, 4 value, 14 abstraction, 5 action, 2, 37 algorithmic engineering, 46 analyser domain, 2 analysis development stage domain, 49 regulation, 50 rule, 50 script facet, 50 semantic qualities, 13 share, 19 shared attribute, 19 signature function, 40 sort, 8 state, 37 static attribute, 17 support technology facet, 49 support technology behaviour facet, 50 syntactic qualities, 13 type, 8 abstract, 8 concrete, 8 expression function, 40

function, 40 function expression, 40 mereology, 22

domain, 2, 5, 47–49 development stage, 49 prompt, 3, 5–8, 10, 15, 22, 25, 36, 37 problem world, 48 product line, 47 prompt domain, 3, 5–8, 10, 15, 22, 25, 36, 37 world problem, 48 architecture software, 48

Science & Engineering

atomic, 2 atomicity, 4 attribute, 4 attributes, 2 axiom sort well-formedness, 28 well-formedness sort, 28 bases knowledge, 46 behaviour, 2, 37 change state, 40 class diagram, 49 component reusable software, 47 software, 48 reusable, 47 composite, 2 part, 4 compositionality, 4 conceive, 5 concept formal, 5 concrete part type, 4 type part, 4 confusion, 29 constructor function type, 40 type function, 40 context, 5 continuous, 2 endurant. 6 time, 40criminal human behaviour, 50 delinquent human behaviour, 50 demo

domain. 2 describer domain, 2 description development domain, 48 domain, 2, 47-49 development, 48 facet, 49 prompt, 3, 10, 14, 16, 22, 25, 36, 37 facet domain, 49 prompt domain, 3, 10, 14, 16, 22, 25, 36, 37 descriptions domain, 48, 49 design software, 2, 48 development description domain, 48 domain description, 48 model-oriented software, 47 requirements, 48 software, 2 model-oriented, 47 development stage analysis domain, 49 domain analysis, 49 diagram class, 49 diligent human behaviour, 50 discrete, 2 endurant, 2, 6, 7 domain, 48, 49 analyser, 2 analysis, 2, 5, 47–49 development stage, 49 prompt, 3, 5-8, 10, 15, 22, 25, 36, 37 demo, 2 describer, 2 description, 2, 47–49 development, 48

Domain Analysis: Endurants — Domain Analysis: Endurants

Domain Analysis: Endurants

facet, 49 prompt, 3, 10, 14, 16, 22, 25, 36, 37 descriptions, 48, 49 development description, 48 development stage analysis, 49 engineer, 2, 48 engineering, 2, 47, 48 facet description, 49 language specific, 47 modelling, 27, 47, 48 prompt analysis, 3, 5-8, 10, 15, 22, 25, 36, 37 description, 3, 10, 14, 16, 22, 25, 36, 37 regulation, 50 rule, 50 science, 2 scientist, 2 simulator, 2software specific, 2, 48 specific language, 47 software, 2, 48 stake-holder, 45 endurant, 2 continuous, 6 discrete, 2, 6, 7 entities, 4 external quality, 13 internal quality, 13 quality external, 13 internal, 13 engineer domain, 2, 48 requirements, 48 software, 2, 48 engineering algorithmic, 46 domain, 2, 47, 48

knowledge, 46 ontological, 46 product line software, 47 requirements, 2, 47 software product line, 47 entities, 2 endurant, 4 entity, 2 event, 2, 37 expression function type, 40 type, 40 function, 40 external endurant quality, 13 quality endurant, 13 facet description domain, 49 domain description, 49 formal concept, 5 formal concept analysis, 5 frame problem, 48 frames problem, 48 function constructor type, 40 expression type, 40 name, 40 type constructor, 40 expression, 40hardware, 48 has_concrete_type prerequisite

© Dines Bjørner 2013, DTU Informatics, Techn.Univ.of Denmark – May 30, 2013: 19:35 34 Domain Analysis: Endurants — An Analysis & Description Process Model

34

Science & Engineering

prompt, 10 prompt prerequisite, 10 has_ mereology prerequisite prompt, 23 prompt prerequisite, 23 has_ unique_ identifier prerequisite prompt, 14 prompt prerequisite, 14 head, 40 human behaviour criminal, 50 delinquent, 50 diligent, 50 sloppy, 50 identification unique, 2 identifier unique, 14 identity, 4 imperative language programming, 46 programming language, 46 internal endurant quality, 13 quality endurant, 13 interval time, 38is_ composite prerequisite prompt, 8 prompt prerequisite, 8 is_ discrete prerequisite prompt, 7 prompt prerequisite, 7

is_ entity prerequisite prompt, 5, 6 prompt prerequisite, 5, 6 join lattice, 13 junk, 28 knowledge bases, 46 engineering, 46 representation, 46 language domain specific, 47 imperative programming, 46 programming imperative, 46 specific domain, 47 lattice join, 13 machine, 48 manifest phenomena, 1 material, 2, 4 mereology, 2, 4 observer, 22 type, 22 method, 2methodology, 2 model-oriented development software, 47 software development, 47 modelling domain, 27, 47, 48 requirements, 27 name function, 40 non-manifest

Domain Analysis: Endurants — Domain Analysis: Endurants

© Dines Bjørner 2013, DTU Informatics, Techn.Univ.of Denmark – May 30, 2013: 19:35

36

Domain Analysis: Endurants

qualities, 1 obligation proof, 29 observable phenomena, 2 phenomenon, 2 observe, 5 observe_part_type prerequisite prompt, 10 prompt prerequisite, 10 observer mereology, 22 ontological engineering, 46 ontology upper, 46 part, 2, 4, 7 composite, 4 concrete type, 4 sort, 8 type concrete, 4 perdurant, 2 phenomena manifest, 1 observable, 2 phenomenon observable, 2 prerequisite has_ concrete_ type prompt, 10 has_ mereology prompt, 23 has_ unique_ identifier prompt, 14 is_ composite prompt, 8 is_ discrete prompt, 7 is_ entity prompt, 5, 6 observe_part_type

prompt, 10 prompt has_concrete_type, 10 has_ mereology, 23 has_ unique_ identifier, 14 is_ composite, 8 is_ discrete, 7 is_ entity, 5, 6observe_part_type, 10 prescription requirements, 2, 47–49 problem analysis world. 48 frame. 48 frames, 48 world, 48 analysis, 48 process schema, 49 product line analysis, 47 engineering software, 47 software, 47 engineering, 47 programming imperative language, 46 language imperative, 46 prompt, 3 analysis domain, 3, 5-8, 10, 15, 22, 25, 36, 37 description domain, 3, 10, 14, 16, 22, 25, 36, 37 domain analysis, 3, 5-8, 10, 15, 22, 25, 36, 37 description, 3, 10, 14, 16, 22, 25, 36, 37 has_concrete_type prerequisite, 10 has_ mereology prerequisite, 23 has_ unique_ identifier prerequisite, 14 is_ composite prerequisite, 8

© Dines Bjørner 2013, DTU Informatics, Techn.Univ.of Denmark – May 30, 2013: 19:35 36 Domain Analysis: Endurants — An Analysis & Description Process Model

Science & Engineering

is_ discrete prerequisite, 7 is_ entity prerequisite, 5, 6 observe_part_type prerequisite, 10 prerequisite has_ concrete_ type, 10 has_mereology, 23 has_ unique_ identifier, 14 is_ composite, 8 is_ discrete, 7 is_ entity, 5, 6 observe_part_type, 10 proof obligation, 29 qualities, 2, 4non-manifest, 1 spatiotemporal, 15 quality, 2 endurant external, 13 internal, 13 external endurant, 13 internal endurant, 13 regulation domain, 50 representation knowledge, 46 requirements, 48, 49 development, 48 engineer, 48 engineering, 2, 47 modelling, 27 prescription, 2, 47–49 reusable component software, 47 software component, 47 reuse, 47 rule domain, 50

schema process, 49 science domain, 2scientist domain, 2 sharing, 13, 14 simulator domain, 2 sloppy human behaviour, 50 software, 48 architecture, 48 component, 48 reusable, 47 design, 2, 48 development, 2 model-oriented, 47 domain specific, 2, 48 engineer, 2, 48 engineering product line, 47 model-oriented development, 47 product line, 47 engineering, 47 reusable component, 47 specific domain, 2, 48 sort, 4 axiom well-formedness, 28 part, 8 well-formedness axiom, 28 spatiotemporal qualities, 15 specific domain language, 47 software, 2, 48 language domain, 47 software domain, 2, 48 stake-holder

Domain Analysis: Endurants — Domain Analysis: Endurants

© Dines Bjørner 2013, DTU Informatics, Techn.Univ.of Denmark – May 30, 2013: 19:35

38

Domain Analysis: Endurants

domain, 45 state, 37 change, 40 sub-part, 7 subpart, 2, 4 time, 37, 38 continuous, 40 interval, 38 TripTych, 2, 5, 46-49 type, 5 abstract, 4 $\operatorname{concrete}$ part, 4 constructor function, 40 expression, 40 function, 40 function constructor, 40 expression, 40 mereology, 22 part

concrete, 4 Unified Modelling Language UML, 48, 49unique identification, 2 identifier, 14 upper ontology, 46 value abstract, 14 well-formedness axiom sort, 28 sort axiom, 28 world analysis problem, 48 problem, 48 analysis, 48